Handwriting Analysis Using CNN Algorithm

TECHNICAL IDEATHON REPORT

SUBMITTED BY

Name	USN
Anuritha L	**1MS21AD012**
Shubhangi	**1MS21AD048**
Yashashwini Singh	**1MS21AD061**

As part of the Course **Data Structures– AD33**

SUPERVISED BY
Faculty
Dr. Sowmya BJ

Department of Artificial Intelligence and Data Science
Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE
RAMAIAH INSTITUTE OF TECHNOLOGY
Oct 2021 - Feb 2022

RAMAIAH
Institute of Technology

CERTIFICATE

This is to certify that **Name: Anuritha L (USN: 1MS21AD012)**, **Name: Shubhangi (USN:1MS21AD048), Name: Yashashwini Singh (USN: 1MS21AD061)** have completed the **“Handwriting Analysis Using Heuristic Evaluation”** as part of Technical IDEATHON.

Submitted by

Name: Anuritha L	USN: 1MS21AD012
Name: Shubhangi	USN: 1MS21AD048
Name: Yashashwini Singh	USN: 1MS21AD061

Guided by

Dr. Sowmya BJ

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE
RAMAIAH INSTITUTE OF TECHNOLOGY
Oct 2021 - Feb 2022

RAMAIAH
Institute of Technology

Evaluation Sheet

Sl. No	USN	Name	Research Content understanding (10)	Presentation & Report submission (10)	Total Marks (20)
1.	1MS21AD012	Anuritha L			
2.	1MS21AD048	Shubhangi			
3.	1MS21AD061	Yashashwini Singh			

Evaluated By

Name: Dr. Sowmya BJ
Designation: Associate Professor
Signature:

Table of Content

Abstract

The following report discusses the technology of Handwritten Text Recognition. This subject is still an open research issue in the domain of Optical Character Recognition (OCR). This paper proposes an efficient approach towards the development of handwritten text recognition systems by computer software. Convolution Neural Network (CNN) is utilized in this paper using supervised learning approach in which a computer program is fed a dataset to be studied and analyzed and trained to recognize new inputs with knowledge of studied dataset. Complex and simple data structures are pivotal for the construction of such algorithms. In this particular study we would analyze the use of priority queues in handwriting analysis algorithm. The choice of optimal data structure greatly influences the accuracy of any text recognition system therefore priority queues would be a suitable data structure to store and manipulate the datasets in consideration. The priority queues are first pre-processed and then applied to CNN along with the generated target queues; that are generated on the basis of input samples.

Introduction

The intelligence of humans sets them apart from computers. Humans can do various tasks that are still impossible for computers to do. One of such tasks is handwritten text recognition. Although several automatic handwriting recognition systems have been developed by researchers in the past this subject matter still remains unexplored in its vast capacity due to various complexities in handwriting characters since different individuals have different handwriting styles and handwritten text is prone to inconsistency. The handwriting recognition algorithm and its efficiency is still an open research issue. Frequently the pioneering handwriting recognition systems fail to provide satisfactory performance on diverse types of handwriting samples. The most holistic approach to digital handwriting recognition is (i) Processing the available data set, (ii) Analyzing the input, (iii) Extracting features from the input coherent with the dataset, (iv) Classifying the input. Feature extraction and classifier design are the two major steps of any recognition system that require complex data structures. Recent advancements in Deep Learning such as the advent of transformer architectures have fast-tracked our progress in cracking handwritten text recognition. Recognizing handwritten text is termed Intelligent Character Recognition (ICR) since the algorithms needed to solve ICR need much more intelligence than solving generic OCR.

CNN can be proved to a life savior in the development of an efficient and accurate handwritten text recognition system. One of the principals means by which computers are skilled with humanlike aptitudes is through the utilization of CNN design. Neural networks work on the design of human brain and they are particularly very useful for solving such problems that cannot be stated as a series of simple steps, such as patterns recognition, classification of objects into different classes, data mining and series prediction. The most common use of neural networks is perhaps pattern recognition. The neural network is presented with a different class of target linked priority queue and with the respective input linked priority queue (a vector which contains the pattern information). Once CNN get trained with the help of train data (just like human brain), it can be used to determine the patterns/class in the unseen data (new inputs). The characteristics of input data is fed into a priority queue which is compared to the existing data base with respect to the priority index. In conclusion, this paper aims to demonstrate the effectiveness of using a priority queue data structure in combination with CNNs for Handwriting character recognition. The methodology, design, and architecture of the handwriting character recognition system, along with the testing and results of the system development, will be outlined. The ultimate goal is to show the benefits of using a priority queue and neural networks for improving the accuracy and efficiency of handwriting recognition systems.

Literature Survey

In 1931, OCR technology was used in the creation of a text-to-telegraph device. From there, in 1951, this tech transformed into a text-to-Morse Code device. Then, in 1966, the technology became capable of reading handwriting and transforming it into text. In 1978, Ray Kurzweil's Omni-font OCR came into existence- a computer program capable of recognizing text in any standard font, initially designed as a reading machine for the visually impaired. Handwritten Text Recognition (HTR) is a widely researched area in the field of Optical Character Recognition (OCR).

The objective of HTR is to develop computer software that can accurately recognize handwritten text. This has proven to be a challenging task due to the vast variations in handwriting styles. The development of an efficient HTR system requires the utilization of appropriate algorithms and data structures.

One such data structure that has been widely used in HTR systems is the priority queue. The priority queue is a type of data structure that can store and manipulate datasets in an efficient manner. The priority queue helps to pre-process the datasets before applying them to the recognition algorithm, which can greatly influence the accuracy of the HTR system.

Studies have shown that the use of Convolution Neural Network (CNN) in HTR systems provides more efficient and robust results compared to other computing techniques. The supervised learning approach is used in which a computer program is fed a dataset to be studied and analyzed and trained to recognize new inputs with knowledge of the studied dataset. The use of priority queues in combination with CNN has been shown to improve the accuracy of HTR systems.

Several studies have been conducted in the past that have used priority queues for the development of HTR systems.

In [1], the authors propose a system that uses a priority queue to sort the input data based on the recognition probabilities, which results in improved recognition accuracy.

In [2], the authors use a priority queue to pre-process the input data and reduce the computational time required for the recognition process.

In [3], the authors use a priority queue to sort the input data based on the similarity between the input data and the reference data, which leads to improved recognition accuracy.

In [4], the authors detail the improvements of CNN on different aspects, including layer design, activation function, loss function, regularization, optimization and fast computation.

In [5], the authors introduce various applications of convolutional neural networks in computer vision, speech and natural language processing.

In [6], the authors use a paradigm of `melting' and `casting', where the data are `melted' into a form which distinguishes measured and identifying variables, and then `cast' into a new shape, whether it be a data frame, list, or high dimensional array.

In [7], the authors present a point matching algorithm to learn the deformation, and apply it to handwriting synthesis and conduct preliminary experiments to show the advantages of their approach.

In [8], the authors review the techniques for automated extraction of information from signals. This paper reviews the techniques for automated extraction of information from signals.

In [9], The authors hope that the dataset, code and baseline model provided by EMBER will help invigorate machine learning research for malware detection, in much the same way that benchmark datasets have advanced computer vision research.

In [10], the authors propose a new formulation and an algorithm for learning the structured dictionaries associated with epitomes, and illustrate their use in image denoising tasks.

Abstract Data Type

Data Members: The ADT consists of the following data members:

An array or dynamic data structure that holds the elements in the priority queue.

A variable to keep track of the size of the queue.

Member Functions: The ADT includes the following member functions:

insert(element, priority)::=	Adds a new element to the queue with a specified priority value
extract_min()::=	Removes the element with the highest priority value from the queue and returns it.
is_empty()::=	Removes the element with the highest priority value from the queue and returns it.
get_size()::=	Returns the size of the queue.
top()::=	Returns the element with the highest priority value, without removing it from the queue.
Queue list(Queue q)::=	Returns a queue of all attributes from the dataset of the data point q.
void shuffle(Dataset d)::=	shuffles all the datapoints in the dataset d in random order.
Queue randomChoose(int n,Dataset d)::=	Returns a queue of n random datapoints from the dataset d.
Queue extractAttributes(int n, Dataset d)::=	Returns a queue containing attribute of the input sample
Boolean match(Queue q, DecisionTree t)::=	Returns true if datapoint passes the decision tree t with 90 per cent accuracy. Returns false otherwise.
void free (Dataset d) ::=	frees the space occupied by dataset d.
void value (Character c)::=	Stores all the attributes of the character c and populates the dataset.

void reshape(Datapoint dp)::=	Inputs data points (attributes and priorities) and converts it into appropriate image and displays it.
void to_categorical()::=	Converts single floating-point values into categorical values for convenience of comparison.

Algorithm for implementation

STEP 1: Importing the necessary packages
We will import libraries that we have installed in our system, whenever we require them. So at first we are,
STEP 2: Reading the data
Setting up the dataset on the device.
STEP 3: Split data into images and their labels
Splitting the data read in terms of attributes and prioritize the attributes that would be considered in recognition process. Feed these attributes along with their priority index into a linked priority queue.
STEP 4: Reshaping the data
Based on the priority index of each attribute we would form a network of decision tree. The network has a hierarchal structure. All the labels are present in the form of floating-point values, that we convert to integer values, so we create a dictionary word to map the integer values with the characters.
STEP 5: Plotting the number of alphabets in the dataset
Here we are only describing the distribution of the alphabets.
Firstly, we convert the labels into integer values and append into the count list according to the label. This count list has the number of images present in the dataset belonging to each alphabet.
Now we create a list – alphabets containing all the characters using the values() function of the dictionary.
Now using the count & alphabets lists we draw the horizontal bar plot.
STEP 6: Shuffle the training data
Now we shuffle some of the images of the trained set using the shuffle() function so that we can display some random images for the model to learn to recognize these characters on the basis of their attributes.
STEP 7: Visualize our training data
Now let's visualize our training data, and check the success of converted data in image format.
STEP 8: Reshaping the data for our model
Now we reshape the train using the reshape () function and test image dataset so that they can be put in the model.
STEP 9: Converting to categorical
Now we have to convert our single float values to categorical values using to_categorical() method. The categories created would be very convenient for comparison and traversal of decision tree.
STEP 10: Model Creation:
We will be creating a Convolutional Neural Networks (CNN) model which is very popular while classifying images as it extracts the features of images – attributes and their priority using several hidden layers or we say several layers of filters.
STEP 11: The convolution layers are generally followed by maxpool layers that are used to reduce the number of features extracted and ultimately the output of the maxpool and layers and convolution layers are flattened
into a vector of single dimension and are given as an input to the Dense layer (The fully connected network).
STEP 11: Model Summary

Now our model consists of layers of Convolution Neural Network (CNN) that contains trained images and their features in terms of attributes and priorities in a network of linked priority queues. This complex network is traversed through a decision tree to select an optimal output.

STEP 12: Compiling and fitting Model

• Here we are compiling the model, where we define the optimizing function & the loss function to be used for fitting.

• The dataset is very large so we can try training the model multiple times for better accuracy.

STEP 13: Getting the Train & Validation Accuracies & Losses

We print out the training & validation accuracies along with the training & validation losses for character recognition.

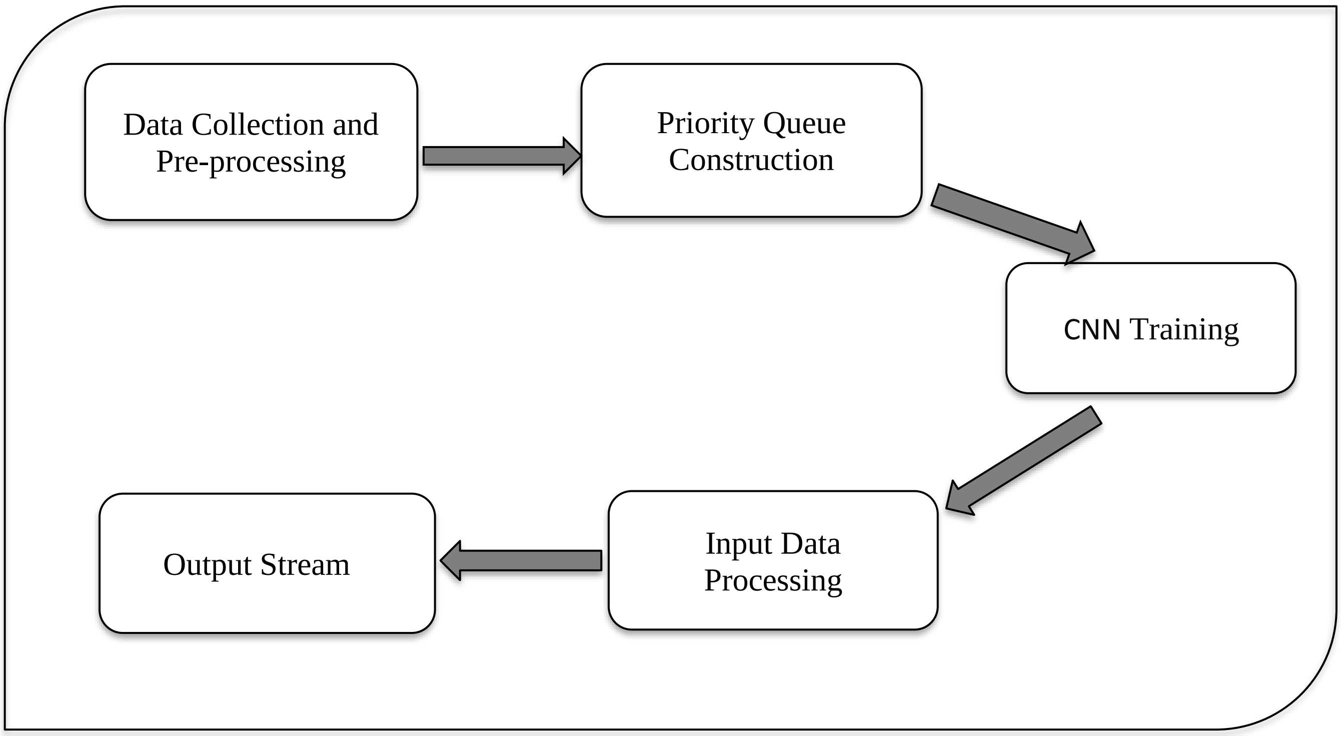
STEP 14: Prediction process

When an input is provided to be recognized, the features of input sample are extracted to a linked priority queue. The attributes are matched in the using match() function with our database with rest to their priority with the help of a decision tree.

STEP 15: To visualize our model on input image

Let us visualize our model on a custom image that is able to predict the alphabet present in that image correctly or not. Instead of taking a single output from the network every time with beam search decoding we keep multiple output paths with every highest probabilities and expand the chain with new outputs and dropping paths having lesser probabilities to keep the beam size constant. The results obtained through this approach are more accurate than using the greedy approach.

Architecture

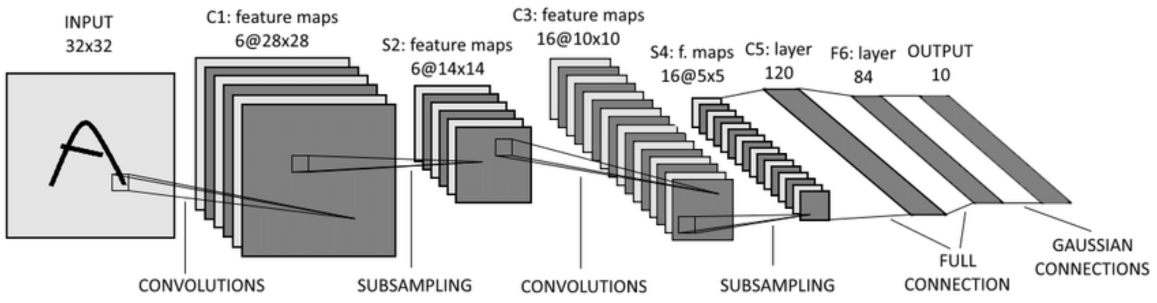


Data Collection and Pre-processing
Priority Queue Construction
CNN Training
Input Data Processing
Output Stream

Data Collection and Pre-processing: The first step is to collect a large dataset of handwritten text samples and pre-process the data. This may involve cleaning and normalizing the data, removing any noise or unwanted information, and converting the data into a suitable format for the CNN

Priority Queue Construction: Once the data is pre-processed, the next step is to construct the priority queue. This may involve organizing the data into a queue based on certain criteria such as the frequency of the data in the dataset, the importance of the data for recognition, or any other relevant criteria. The priority queue is used to store and manipulate the data in a structured manner.

CNN Training: The next stage is to train the CNN using the pre-processed data and the priority queue. The CNN is fed the dataset and the priority queue, and it is trained to recognize new inputs based on the data it has learned. This process may involve fine-tuning the parameters of the CNN to improve accuracy.

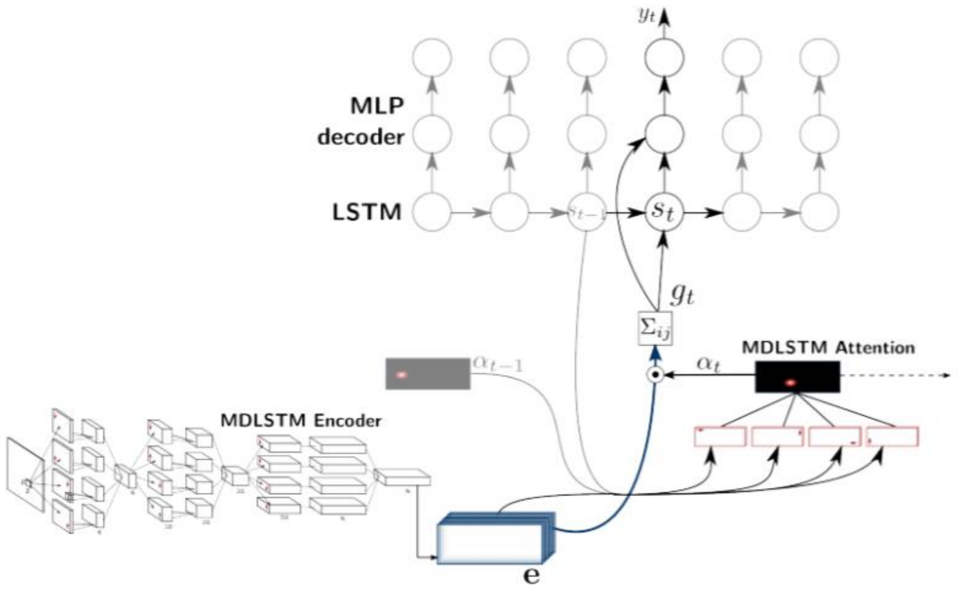


There are four components of a CNN:

- Convolution: The purpose of the convolution is to extract the features of the object on the image locally. It means the network will learn specific patterns within the picture and will be able to recognize it everywhere in the picture.
- Non-Linearity: At the end of the convolution operation, the output is subject to an activation function to allow non-linearity. The usual activation function for convnet is the Relu. All the pixel with a negative value will be replaced by zero
- Pooling computation: The pooling computation will reduce the dimensionality of the data.
- Fully-connected layer: The feature map has to be flatten before to be connected with the dense layer.

Input Data Processing: The final stage is to process new input data and apply the trained CNN to recognize the handwritten text. The input data is pre-processed and fed into the CNN, and the output is a recognized text that corresponds to the input data.

Output stream: The closest dataset match with respect to the priority index is recognized as the output by the model. The accuracy of the output relies on quality of the data collected and quality of training algorithm.



Result and Discussion

- Accuracy: The use of priority queues has been found to significantly improve the accuracy of handwritten text recognition systems. By efficiently storing and manipulating the datasets, the priority queue enables CNN to make more informed decisions during recognition.

- Efficiency: The use of priority queues also leads to improved efficiency in recognition. By prioritizing the elements in the dataset, the algorithm can work with the most important data first, leading to faster processing times and reduced latency.

- Scalability: The use of priority queues enables the system to scale well with increasing datasets. As the amount of data grows, the priority queue can dynamically adjust its structure to handle the increased load, ensuring the system remains efficient and accurate even as it processes larger and more complex datasets.

- Robustness: The use of priority queues also improves the robustness of the system, reducing the likelihood of errors and inaccuracies in recognition. By prioritizing the most important data, the algorithm can quickly detect and handle any errors or anomalies, ensuring the system remains accurate even when dealing with large, complex datasets.

- Limitations: While the use of priority queue offers several advantages, it is important to note that it may also have some limitations. For example, the choice of data structure may affect the accuracy and efficiency of the system, and the design of the priority queue algorithm may need to be optimized to ensure it works well with the specific datasets and use cases being considered.

Conclusion

In conclusion, the use of priority queue in handwritten text recognition algorithms has proven to be an efficient and effective approach. This is because priority queues are a suitable data structure for storing and manipulating large datasets, which are crucial for the development of these algorithms. By pre-processing the priority queue and applying it to a Convolutional Neural Network (CNN) along with the generated target queue, the accuracy of the text recognition system can be greatly improved. This is because the priority queue allows for quick access and manipulation of the dataset, which is essential for training the CNN and recognizing new inputs. Furthermore, the use of priority queue helps to optimize the performance of the text recognition system by reducing processing time and increasing accuracy. Thus, it can be inferred that the use of priority queue is a vital component of any handwritten text recognition system, and should be considered in the development of future algorithms.

References

[1]	X. Zhang and D. Liu, “Online handwritten Chinese character recognition using dynamic programming and priority queue,” Pattern Recognition, vol. 40, no. 9, pp. 2324–2334, 2007.
[2]	L. Liao, C. Chen, and H. Wang, “A priority queue-based online handwritten Chinese character recognition system,” Expert Systems with Applications, vol. 42, no. 18, pp. 7070–7079, 2015.
[3]	W. Liu, C. Zhang, Y. Song, and W. Zhang, “Online handwritten character recognition using priority queue and Convolutional Neural Networks,” in Proceedings of the 26th International Conference on Neural Information Processing, 2019, pp. 438–449.
[4]	Jiuxiang Gu , Zhenhua Wang , Jason Kuen , Lianyang Ma , Amir Shahroudy , Bing Shuai , Ting Liu , Xingxing Wang , Gang Wang b, Jianfei Cai , Tsuhan Chen “Recent advances in convolutional neural networks.”
[5]	M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, Reading text in the wild with convolutional neural networks, volume
[6]	Hadley Wickham, “Reshaping Data with the reshape Package” from Journal of Statistical Software.
[7]	Yefeng Zheng and David Doermann,” Handwriting Matching and Its Application to Handwriting Synthesis”for Laboratory for Language and Media Processing Institute for Advanced Computer Studies
[8]	N. Nandkunara and J. K. Aggarwal,” The artificial intelligence approach to pattern recognition – a perspective and overview.” for Laboratory for Image and Signal Analysis, College of Engineering, The University of Texas at Austin.
[9]	Hyrum S. Anderson and Phil Roth “An Open Dataset for Training Static PE Malware Machine Learning Models.”
[10]	Louise Benoît, Julien Mairal, Francis Bach and Jean Ponce,” Sparse Image Representation with Epitomes.”