HANDWRITING RECOGNITION USING MACHINE LEARNING METHODS

Software Requirement Specifications

Bachelor of Technology

in

Computer Science and Engineering

of

Cochin University of Science and Technology

by

Aniljith K

Aravind Sankar

Balu R Krishnan

Karthik K

Sangeeth Dev S

November, 2017

Department of Computer Engineering

College of Engineering, Chengannur -689121

Phone: (0479) 2454125, 2451424; Fax: (0479) 2451424

College of Engineering, Chengannur

Kerala

Contents

List of Figures

1 INTRODUCTION

Handwritten character recognition is a field of research in artificial intelligence, computer vision, and pattern recognition. A computer performing handwriting recognition is said to be able to acquire and detect characters in paper documents, pictures, touch-screen devices and other sources and convert them into machine-encoded form. Its application is found in optical character recognition and more advanced intelligent character recognition systems. Most of these systems nowadays implement machine learning mechanisms such as neural networks.

Neural networks are learning models used in machine learning. Their aim is to simulate the learning process that occurs in an animal or human neural system. Being one of the most powerful learning models, they are useful in automation of tasks where the decision of a human being takes too long, or is imprecise. A neural network can be very fast at delivering results and may detect connections between seen instances of data that human cannot see

Despite the abundance of technological writing tools, many people still choose to take their notes traditionally: with pen and paper. However, there are drawbacks to handwriting text. Its difficult to store and access physical documents in an efficient manner, search through them effi- ciently and to share them with others. Thus, a lot of important knowledge gets lost or does not get reviewed because of the fact that documents never get transferred to digital format

2 SCOPE

The aim of this project is to further explore the task of classifying handwritten text and to convert handwritten text into the digital format. Handwritten text is a very general term, and we wanted to narrow down the scope of the project by specifying the meaning of handwritten text for our purposes. In this project, we took on the challenge of classifying the image of any handwritten word, which might be of the form of cursive or block writing.

3 OVERALL DESCRIPTION

In summary, our models take in an image of a word and output the content of the image.

It has already been stated that the primary goal of our project a System application is able to recognize handwritten characters based on user's input and image/camera input in an offline manner. Also, the application provides means of adding and learning a new character and learning interactively from user's feedback.''

3.1 MACHINE LEARNING

The next major upgrade in producing high OCR accu- racies was the use of a Hidden Markov Model for the task of OCR. This approach uses letters as a state, which then allows for the context of the character to be accounted for when determining the next hidden variable [8]. This lead to higher accuracy compared to both feature extraction tech- niques and the Naive Bayes approach [7]. The main draw- back was still the manual extraction features, which requires prior knowledge of the language and was not particularly robust to the diversity and complexity of handwriting.

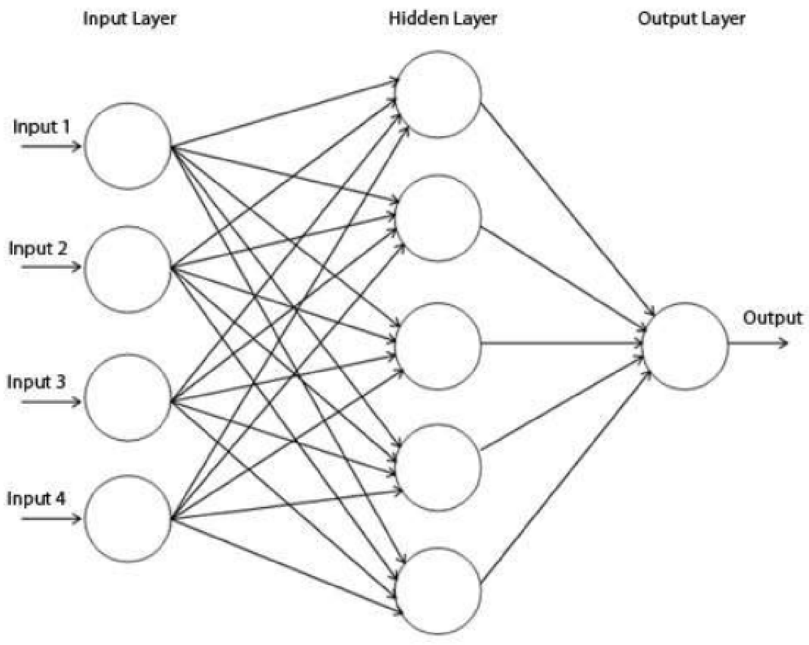


Figure 3.1: Architecture Of Neural Network

4 RELATED WORKS

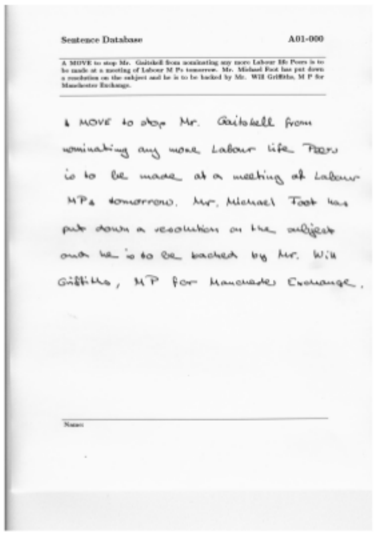
4.1 Early Scanners

The first driving force behind handwritten text classifi- cation was for digit classification for postal mail. Jacob Rabinows early postal readers incorporated scanning equip- ment and hardwired logic to recognize mono-spaced fonts [3]. Allum et. al improved this by making a sophisticated scanner which allowed for more variations in how the text was written as well as encoding the information onto a bar- code that was printed directly on the letter [4].

4.2 To the digital age

The first prominent piece of OCR software was invented by Ray Kurzweil in 1974 as the software allowed for recog- nition for any font [5]. This software used a more developed use of the matrix method (pattern matching). Essentially, this would compare bitmaps of the template character with the bitmaps of the read character and would compare them to determine which character it most closely matched with. The downside was this software was sensitive to variations in sizing and the distinctions between each individuals way of writing.

To improve on the templating, OCR software began us- ing feature extraction rather than templating. For each char- acter, software would look for features like projection his- tograms, zoning, and geometric moments [6].



Sentence Database A01-000

A MOVE to stop Mr. Gaitskell from nominating any more Labour life Peers is to be made at a meeting of Labour M Ps tomorrow. Mr. Michael Foot has put down a resolution on the subject and he is to be backed by Mr. Will Griffiths, M P for Manchester Exchange.

A MOVE to stop Mr. Gaitskell from
nominating any more Labour life Peers
is to be made at a meeting of Labour
MPs tomorrow. Mr. Michael Foot has
put down a resolution on the subject
and he is to be backed by Mr. Will
Griffiths, MP for Manchester Exchange.

Name:

Figure 1. An example form from the IAM Handwriting dataset. Word images in the dataset were extracted from such forms.

Figure 4.1: Dataset

5 DESIGN

Design phase is mainly classified as follows:

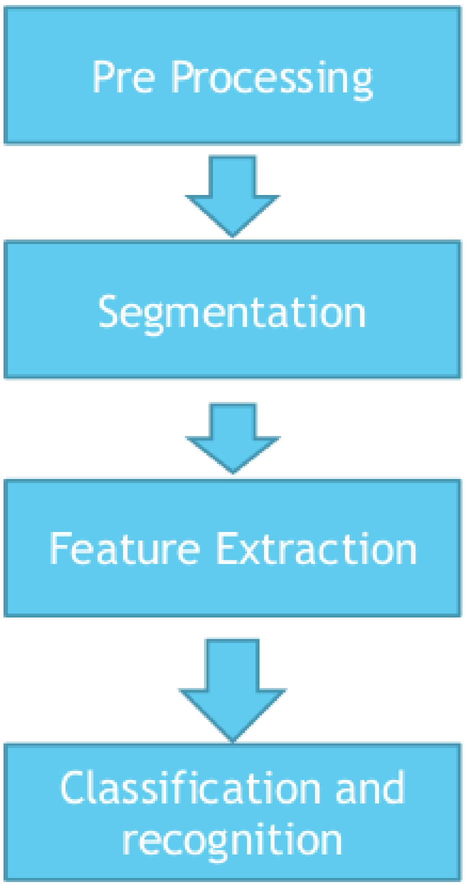


Figure 5.1: Methodology

5.1 PRE-PROCESSING

Before training our models with the dataset, we have applied various preprocessing and data augmentation tech- niques on our dataset in order to make our data more com- patible with the models and to make our dataset more robust to real life situations.

5.2 SEGMENTATION

For word-level classification, we suspected that our per- formance was suffering because of the large softmax layer output size (there were over 10,000 words in our training vocabulary and well over 100,000 words in the English language alone) and the difficulty of fine-grained recogni- tion of images of words. We decided that character-level classification may be a more promising approach because a fairly comprehensive character vocabulary is relatively much smaller than a similarly comprehensive word vocab- ulary (the characters A-Za-z0-9 are only 62 distinct sym- bols), significantly limiting the computational complexity of the softmax.

Furthermore, recognition of a character in an image is a simpler task than recognition of a word in an image because of the limited range of characters. How- ever, the first main challenge we would have to encounter in order to test this approach would be segmentation of word images into their component character images. The sec- ond main challenge would be recognizing word breaks in image and stringing together consecutive recognized char- acters in between these word breaks to form words. We will address the earlier of these challenges in this section. In order to implement this task, we employed the new Tesser- act 4.0 neural network-based CNN/LSTM engine[13]. This model is configured as a textline recognizer originally developed by HP and now maintained by Google that can rec- ognize more than 100 languages out of the box. For Latin- based languages, including English, the Tesseract model had been trained on about 400000 textlines spanning about 4500 fonts. We then finetuned the parameters of this pre- trained model on our IAM handwriting dataset. After we finetuned the model, we segmented the original word in- put images into their hypothesized component character images, feeding in these output segmented character images into our character-level classification.

5.3 FEATURE EXTRACTION

Features of input data are the measurable properties of observations, which one uses to analyze or classify these instances of data. The task of feature extraction is to choose relevant features that discriminate the instances well and are independent of each other. According to [3], selection of a feature extraction method is probably the single most important factor in achieving high recognition performance. There is a vast amount of methods for feature extraction from character images, each having different characteristics, invariance properties, and reconstructability of characters. [3] states that in order to answer to the question of which method is best suited for a given situation, an experimental evaluation must be performed. The methods explained in [3] are template matching, deformable templates, unitary image transforms, graph description, projection histograms, contour profiles, zoning, geometric moment invariants, Zernike moments, spline curve approximation, and Fourier descriptors. To describe the way feature extraction is sometimes done in handwriting recognition, we briefly explain one of them.

5.4 CLASSIFICATION AND RECOGNITION

This is the last and final stage of our design phase.

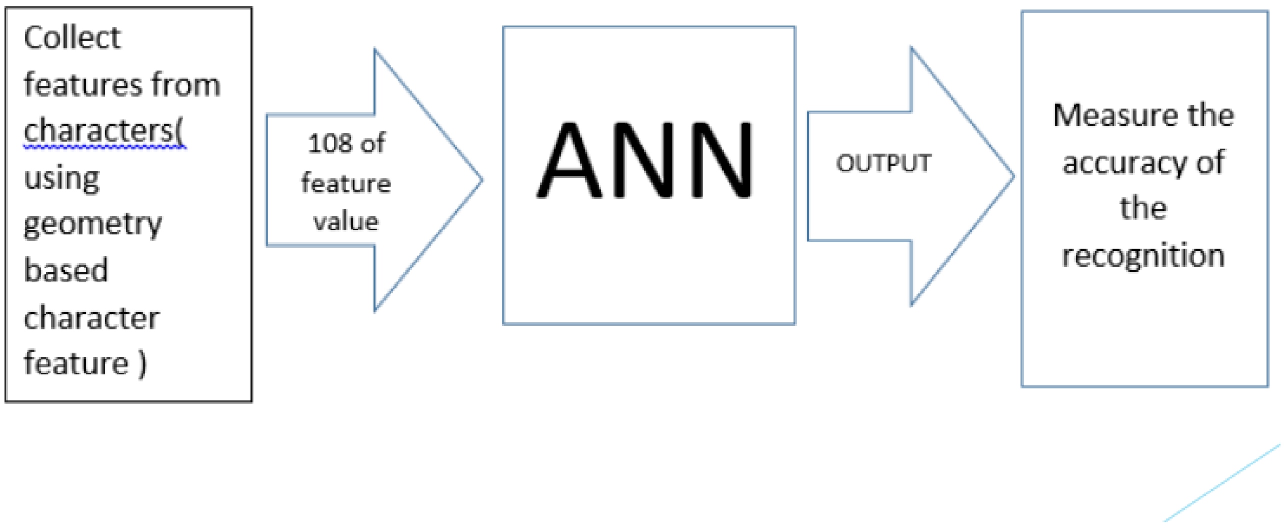

Collect features from characters(using geometry based character feature)
108 of feature value
ANN
OUTPUT
Measure the accuracy of the recognition

Figure 5.2: Design for the Artificial Neural Network

6 IMPLEMENTATION

CHARACTER RECOGNITION ALGORITHMS

The main algorithm used in Character Recognition is classification. They are normally used in sequence image preprocessing helps to make feature extraction a smoother process, while feature extraction is necessary for correct classification.

6.1 CLASSIFICATION

Classification is defined as the task of assigning labels (categories, classes) to yet unseen observations (instances of data). In machine learning, this is done on the basis of training an algorithm on a set of training examples. Classification is a supervised learning problem, where a teacher links a label to every instance of data. Label is a discrete number that identifies the class a particular instance belongs to. It is usually represented as a non- negative integer.

There are many machine learning models that implement classification; these are known as classifiers. The aim of classifiers is to fit a decision boundary (Figure 4) in feature-space that separates the training examples, so that the class of a new observation instance can be correctly labeled. In general, the decision boundary is a hyper-surface that separates an N- dimensional space into two partitions, itself being N1-dimensional.

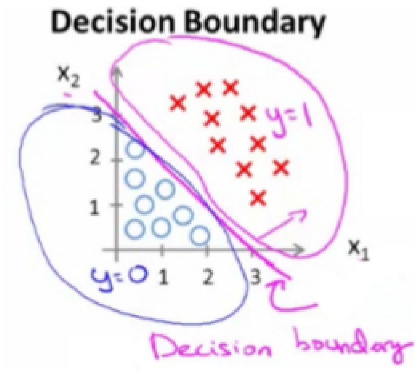


Figure 5: Visualization of a decision boundary. In feature-space given by x1 and x2, a decision boundary is plotted between two linearly separable classes [9].

Figure 6.1: Decision Boundary

6.1.1 Logistic Regression

Logistic regression is a simple linear classifier. This algorithm tries to find the decision boundary by iterating over the training examples, trying to fit parameters that describe the decision boundary hyper-surface equation. During this learning process, the algorithm computes a cost function (also called error function), which represents the error measure of its hypothesis (the output value, prediction). This value is used for penalization, which updates the parameters to better fit the decision boundary. The goal of this process is to converge to parameter values that minimize the cost function. It has been proved [8] that logistic regression is always convex, therefore the minimization process can always converge to a minimum, thus finding the best fit of the decision boundary this algorithm can provide.

Until now, we have been discussing binary classification. To apply logistic regression to handwriting recognition, we would need more than 2 distinguishing classes, hence we need multi-class classification. This can be solved by using the one-vs-all approach [9].

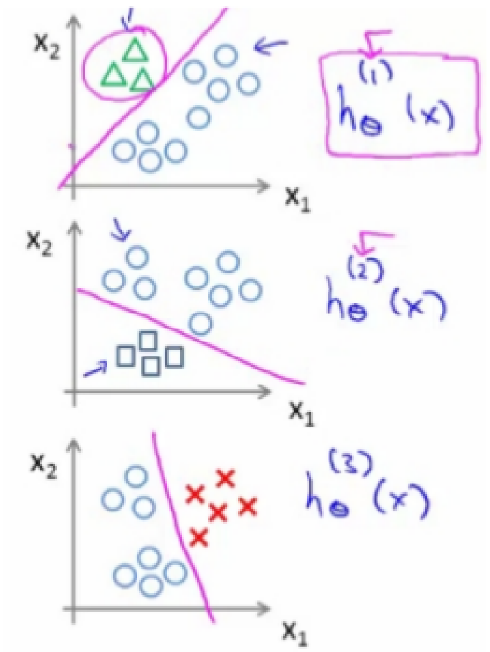


Figure 6: Multi-class classification of three classes as it is split into three sub-problems [9].

Figure 6.2: Logical Regression

The principle of one-vs-all is to split the training set into a number of binary classification problems. Considering we want to classify handwritten digits, the problem degrades into 10 sub-problems, where individual digits are separated from the rest. Figure 6 shows the same for 3 classes of objects. The output of each sub-problem is a probability measure representing how likely a data instance belongs to the class at hand. The overall class is chosen as the class the sub-problem of which has the maximum probability.

6.1.2 Multilayer Perceptron

Multilayer perceptrons (MLPs) are artificial neural networks, learning models inspired by biology. As opposed to logistic regression, which is only a linear classifier on its own, the multilayer perceptron learning model, which we already have mentioned in terms of feature extraction, can also distinguish data that are not linearly separable. We have already outlined the architecture of an MLP, as seen in (Figure 4).

In order to calculate the class prediction, one must perform feedforward propagation. Input data are fed into the input layer and propagated further, passing through weighted connections into hidden layers, using an activation function. Hence, the node's activation (output value at the node) is a function of the weighted sum of the connected nodes at a previous layer. This process continues until the output layer is reached.

The learning algorithm, backpropagation, is different from the one in logistic regression. First, the cost function is measured on the output layer, propagating back to the connections between the input and the first hidden layer afterwards, updating unit weights.

6.2 SOFTWARE REQUIREMENTS

6.2.1 PYTHON

Python is a widely used high-level programming language for general-purpose programming.Python has a design philosophy that emphasizes code readabil- ity and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java.Python features a dynamic type system and automatic memory management and supports multi- ple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems.Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name reso- lution, which binds method and variable names during program execution.

6.2.2 OPEN CV

OpenCV (Open Source Computer Vision) is a library of programming func- tions mainly aimed at real-time computer vision.OpenCV supports the Deep Learning frameworks TensorFlow, Torch/PyTorch and Caffe.It is written in C++ and its primary interface is in C++, but it still retains a less comprehen- sive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. OpenCV runs on a variety of platforms. The features include,advance vision research by providing not only open but also optimized code for basic vision infrastructure, disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable,advance vision-based commercial appli- cations by making portable, performance-optimized code available for freewith a license that did not require code to be open or free itself.The major applica- tions are OpenCVs application areas include:2D and 3D feature toolkits,Facial recognition system, Gesture recognition, Humancomputer interaction (HCI), Mobile robotics, Motion understanding, Object identification etc.