www.ioirp.com
International Journal of Innovative Research in Computer Science and Engineering (IJIRCSE)
ISSN: 2394-6364, Volume – 2, Issue – 7. November 2017

Neural Network for hierarchical method in Optical Character Recognition

R.Shanmugam[1] &P.Subramaniam[2]

[1]*Research* Scholar, *Department of Computer Science,*
Muthayammal College of Arts & Science, India
[2]*Hod & Associate Professor, Department of Computer Science,*
Muthayammal College of Arts & Science, India

***Abstract*— The paper has employed the MLP technique mentioned and excellent results were obtained for a number of widely used font types. The technical approach followed in processing input images, detecting graphic symbols, analyzing and mapping the symbols and training the network for a set of desired Unicode characters corresponding to the input images are discussed in the subsequent sections. Even though the implementation might have some limitations in terms of functionality and robustness, the researcher is confident that it fully serves the purpose of addressing the desired objectives. This irregularity undoubtedly widens when one deals with handwritten characters. Hence the conventional programming methods of mapping symbol images into matrices, analyzing pixel and/or vector data and trying to decide which symbol corresponds to which character would yield little or no realistic results. Clearly the needed methodology will be one that can detect 'proximity' of graphic representations to known symbols and make decisions based on this proximity. To implement such proximity algorithms in the conventional programming one needs to write endless code, one for each type of possible irregularity or deviation from the assumed output either in terms of pixel or vector parameters, clearly not a realistic fare. Such networks can be fed the data from the graphic analysis of the input picture and trained to output characters in one or another form. Specifically some network models use a set of desired outputs to compare with the output and compute an error to make use of in adjusting their weights. Such learning rules are termed as Supervised Learning. One such network with supervised learning rule is the Network layer Perceptron (MLP) model. It uses the Generalized Delta Learning Rule for adjusting its weights and can be trained for a set of input/desired output values in a number of iterations. The very nature of this particular model is that it will force the output to one of nearby values if a variation of input is fed to the network that it is not trained for, thus solving the proximity issue. Both concepts will be discussed in the introduction part of this report.**

Keywords—Neural Network, Optical Character Recognition, MLP, Applications

I. INTRODUCTION

In this modeling systems and functions using neural network mechanisms is a relatively new and developing science in computer technologies. The particular area derives its basis from the way neurons interact and function in the natural animal brain, especially humans. The animal brain is known to operate in massively parallel manner in recognition, reasoning, reaction and damage recovery. All these seemingly sophisticated undertakings are now understood to be attributed to aggregations of very simple algorithms of pattern storage and retrieval. Neurons in the brain communicate with one another across special electrochemical links known as synapses. At a time one neuron can be linked to as many as 10,000 others although links as high as hundred thousands are observed to exist. The typical human brain at birth is estimated to house one hundred billion plus neurons. Such a combination would yield a synaptic connection of 10^{15}, which gives the brain its power in complex spatio-graphical computation. Unlike the animal brain, the traditional computer works in serial mode, which is to mean instructions are executed only one at a time, assuming a uni-processor machine. The illusion of multitasking and real-time interactivity is simulated by the use of high computation speed and process scheduling. In contrast to the natural brain which communicates internally in electrochemical links, that can achieve a maximum speed in milliseconds range, the microprocessor executes instructions in the lower microseconds range. A modern processor such as the Intel Pentium-4 or AMD Opteron making use of multiple pipes and hyper-threading technologies can perform up to 20 MFloPs (Million Floating Point executions) in a single second.

It is the inspiration of this speed advantage of artificial machines, and parallel capability of the natural brain that motivated the effort to combine the two and enable performing complex 'Artificial Intelligence' tasks believed to be impossible in the past. Although artificial neural networks are currently implemented in the traditional serially operable computer, they still utilize the parallel power of the brain in a simulated manner. Neural networks have seen an explosion of interest over the last few years, and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics. Indeed, anywhere that there are problems of prediction, classification or control, neural networks are being introduced. This sweeping success can be attributed to a few key factors:

A. The Network Layer in Neural Network Model

In this every step activation function is used (i.e., the neuron's output is 0 if the input is less than zero, and 1 if the input is greater than or equal to 0) then the neuron acts just like the biological neuron described earlier (subtracting the threshold from the weighted sum and comparing with zero is equivalent to comparing the weighted sum to the threshold). Actually, the step function is rarely used in artificial neural networks, as will be discussed. Note also that weights can be negative, which implies that the synapse has an inhibitory rather than excitatory effect on the neuron: inhibitory neurons are found in the brain. This describes an individual neuron. The next question is: how should neurons be connected together? If a network is to be of any use, there must be inputs (which carry the values of variables of interest in the outside world) and outputs (which form predictions, or control signals). Inputs and outputs correspond to sensory and motor nerves such as those coming from the eyes and leading to the hands. However, there also can be hidden neurons that play an internal role in the network. The input, hidden and output neurons need to be connected together.

A typical feedforward network has neurons arranged in a distinct layered topology. The input layer is not really neural at all: these units simply serve to introduce the values of the input variables. The hidden and output layer neurons are each connected to all of the units in the preceding layer. Again, it is possible to define networks that are partially-connected to only some units in the preceding layer; however, for most applications fully-connected networks are better. The Network layer Perceptron Neural Network is perhaps the most popular network architecture in use today. The units each perform a biased weighted sum of their inputs and pass this activation level through an activation function to produce their output, and the units are arranged in a layered feedforward topology. The network thus has a simple interpretation as a form of input-output model, with the weights and thresholds (biases) the free parameters of the model. Such networks can model functions of almost arbitrary complexity, with the number of layers, and the number of units in each layer, determining the function complexity. Important issues in Multilayer Perceptrons (MLP) design include specification of the number of hidden layers and the number of units in each layer.

B. Optical Language Symbols

Medium structure alphabets like the Ethiopic (Ge'ez) conserve space due to representation of whole audioglyphs and tones in one symbol, but dictate the necessity of having extended sets of symbols and thus a difficult level of use and learning. Some alphabets, namely the oriental alphabets, exhibit a very low amount of structuring that whole words are delegated by single symbols. Such languages are composed of several thousand symbols and are known to need a learning cycle spanning whole lifetimes. These characters are either a delegate of a specific audioglyph, accent or whole words in some cases. In terms of structure world language characters manifest various levels of organization. With respect to this structure there always is an issue of compromise between ease of construction and space conservation. Highly structured alphabets like the Latin set enable easy construction of language elements while forcing the use of additional space. Representing alphabetic symbols in the digital computer has been an issue from the beginning of the computer era.

The initial efforts of this representation (encoding) was for the alphanumeric set of the Latin alphabet and some common mathematical and formatting symbols. It was not until the 1960's that a formal encoding standard was prepared and issued by the American computer standards bureau ANSI and named the ASCII Character set. It is composed of and 8-bit encoded computer symbols with a total of 256 possible unique symbols. In some cases certain combination of keys were allowed to form 16-bit words to represent extended symbols. The final rendering of the characters on the user display was left for the application program in order to allow for various fonts and styles to be implemented. At the time, the 256+ encoded characters were thought of suffice for all the needs of computer usage. But with the emergence of computer markets in the non-western societies and the internet era, representation of a further set of alphabets in the computer was necessitated. Initial attempts to meet this requirement were based on further combination of ASCII encoded characters to represent the new symbols. This however led to a deep chaos in rendering characters especially in web pages since the user had to choose the correct encoding on the browser. Further difficulty was in coordinating the usage of key combinations between different implementers to ensure uniqueness. Network components are prone to a variety of faults such as packet loss, link cut, or node outage. To prevent the faulty components from hindering network applications, it is important to diagnose (i.e., detect and localize) the components that are the root cause of network faults. However, it is also desirable to repair the faulty components to enable them to return to their operational states.

Therefore, we focus on network fault correction, by which we mean not only to diagnose, but also to repair all faulty components within a network. In addition, it has been shown that a network outage can bring significant economic loss. For example, the revenue loss due to a 24-hour outage of a Switzerland based Internet service provider can be more than CHF 40 million. As a result, we want to devise a cost effective network fault correction mechanism that corrects all network faults at minimum cost. To diagnose (but not repair) network faults, recent approaches like use all network nodes to collaboratively achieve this. For instance,

in hop-by-hop authentication each hop inspects packets received from its previous hop and reports errors when packets are found to be corrupted. While such a distributed infrastructure can accurately pinpoint network faults, deploying and maintaining numerous monitoring points in a large-scale network introduces heavy computational overhead in collecting network statistics and involves complicated administrative management. In particular, it is difficult to directly monitor and access all overlay nodes in an externally managed network, whose routing nodes are independently operated by various administrative domains. In this case, we can only infer the network condition from end-to-end information.Here, we consider an end-to-end inference approach which, using end-to-end measurements, infers components that are probably faulty in forwarding data in an application-layer overlay network whose overlay nodes are externally managed by independent administrative domains.

We start with a routing tree topology with a set of overlay nodes, since a tree-based setting is typically seen in destination-based routing and where each overlay node builds a routing tree with itself as a root, as well as in multicast routing, where a routing tree is built to connect members in a multicast group. We then monitor every root-to-leaf overlay path. If a path exhibits any "anomalous behavior" in forwarding data, then some "faulty" overlay node on the path must be responsible. In practice, the precise definition of an "anomalous behavior" depends on specific applications. For instance, a path is said to be anomalous if it fails to deliver a number of correct packets within a time window. Using the path information collected at the application endpoints (i.e., leaf nodes), we can narrow down the space of possibly faulty overlay nodes. In the above end-to-end solution, one can tell whether a path behaves anomalously, but cannot specifically tell which and how many overlay nodes on the path are faulty. Since we cannot directly monitor and access externally managed overlay nodes, in order to correct the faulty nodes, we need to contact the administrators of the corresponding domains to manually check a sequence of potentially faulty nodes and fix any nodes that are found to be actually faulty. Given the anomalous paths in a tree, our main goal is to infer the best node (or the best set of nodes) that should be first checked so as to minimize the expected cost of correcting all faulty nodes.

2. TECHNICAL PLANNING

The operations of the network implementation in this paper can be summarized by the following steps:

A. **Training phase**

- Analyze image for characters
- Convert symbols to pixel matrices
- Retrieve corresponding desired output character and convert to Unicode
- Lineraize matrix and feed to network
- Compute output
- Compare output with desired output Unicode value and compute error
- Adjust weights accordingly and repeat process until preset number of iterations
- **Testing phase**
- Analyze image for characters
- Convert symbols to pixel matrices
- Compute output
- Display character representation of the Unicode output
- Essential components of the implementation are:
- Formation of the network and weight initialization routine
- Pixel analysis of images for symbol detection
- Loading routines for training input images and corresponding desired output characters in special files named character trainer sets (*.cts)
- Loading and saving routines for trained network (weight values)
- Character to binary Unicode and vice versa conversion routines
- Error, output and weight calculation routines

3. EXISTING SYSTEM

Existing monitoring link delays and faults in a service provider or enterprise IP network. Our two-phased approach attempts to minimize both the monitoring infrastructure costs as well as the additional traffic due to probe messages. In the first phase of our approach, we compute the locations of a minimal set of monitoring stations such that all network links are covered, even in the presence of several link failures. Subsequently, in the second phase, we compute a minimal

set of probe messages that are transmitted by the stations to measure link delays and isolate network faults. We show that both the station selection problem as well as the probe assignment problem is NP-hard. We then propose greedy approximation algorithms that achieve a logarithmic approximation factor for the station selection problem and a constant factor for the probe assignment problem.

4. PROPOSED SYSTEM

We propose several efficient heuristics for inferring the best node to be checked in large-scale networks. By extensive simulation, we show that we can infer the best node in at least 95% of time, and that first checking the candidate nodes rather than the most likely faulty nodes can decrease the checking cost of correcting all faulty nodes. As a result, we want to devise a cost effective network fault correction mechanism that corrects all network faults at minimum cost. To diagnose (but not repair) network faults, recent approaches like use all network nodes to collaboratively achieve this. For instance, in hop-by-hop authentication each hop inspects packets received from its previous hop and reports errors when packets are found to be corrupted. While such a distributed infrastructure can accurately pinpoint network faults, deploying and maintaining numerous monitoring points in a large-scale network introduces heavy computational overhead in collecting network statistics and involves complicated administrative management. We present the optimality results for an end-to-end inference approach to correct (i.e., diagnose and repair) probabilistic network faults at minimum expected cost. One motivating application of using this end-to-end inference approach is an externally managed overlay network, where we cannot directly access and monitor nodes that are independently operated by different administrative domains, but instead we must infer failures via end to-end measurements. We show that first checking the node that is most likely faulty or has the least checking cost does not necessarily minimize the expected cost of correcting all faulty nodes.

The MLP Network implemented for the purpose of this paper is composed of 3 layers, one input, one hidden and one output. The input layer constitutes of 150 neurons which receive pixel binary data from a 10x15 symbol pixel matrix. The size of this matrix was decided taking into consideration the average height and width of character image that can be mapped without introducing any significant pixel noise.The hidden layer constitutes of 250 neurons whose number is decided on the basis of optimal results on a trial and error basis. The output layer is composed of 16 neurons corresponding to the 16-bits of Unicode encoding.

5. IMPLEMENTATION

Implementation is the stage of the paper when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

A. MANAGED OVERLAY NETWORK

Network components are prone to a variety of faults such as packet loss, link cut, or node outage. To prevent the faulty components from hindering network applications, it is important to diagnose (i.e., detect and localize) the components that are the root cause of network faults. However, it is also desirable to repair the faulty components to enable them to return to their operational states. Therefore, we focus on network fault correction, by which we mean not only to diagnose, but also to repair all faulty components within a network. We want to devise a cost effective network fault correction mechanism that corrects all network faults at minimum cost in diagnosing and repairing faulty nodes in an externally managed overlay network, in which overlay nodes are independently operated by multiple administrative domains.

B. TRANSMITTER MODULE

The transmitter sends a packet to the receiver and waits for its acknowledgment. Based on error-detection results, the receiver generates either a negative acknowledgment (NACK) or a positive acknowledgment (ACK) for each received packet and sends it over a feedback channel. If an ACK is received, the transmitter sends out a next packet; otherwise, if an NACK is received, retransmission of the same packet will be scheduled immediately, and this process continues until the packet is positively acknowledged.

6. CONCLUSIONS

Some symbol sequences are orthogonally inseparable. This is to mean there cannot be a vertical line that passes between the two symbols without crossing bitmap areas of either. Such images could not be processed for individual symbols within the limits of the paper since it requires complex image processing algorithms. Some cases are presented below:

References

[1]Artificial Intelligence and cognitive science © 2006, Nils J. Nilsson Stanford AI Lab.
[2] Neural Networks to produce an Adaptive quality gratitude System © 2002, Alexander J. Faaborg Cornell University, Ithaca NY
[3] Hand-Printed Character Recognizer using Neural Network © 2000, Shahzad Malik
[4] Neural Networks and Fuzzy Logic. © 1995, Rao, V., Rao, H. MIS Press, New York
[5] Neural Networks © 2003, StatSoft Inc. http://www.statsoft.com/textbook/stneunet.html.
[6] M.Balaji, E.Aarthi, K.Kalpana, B.Nivetha, D.Suganya "Adaptable and Reliable Industrial Security System using PIC Controller" 2017/5, Journal International Journal for Innovative Research in Science & Technology, Volume 3, Issue 12, Page 56-60.
[7]Off-line Handwriting Recognition Using Artificial Neural Networks © 2000, Andrew T. Wilson University of Minnesota, Morris http://wilsonat@mrs.umn.edu
[8] Using Neural Networks to Create an Adaptive Character Recognition System © 2002, Alexander J. Faaborg Cornell University, Ithaca NY
[9] Hand-Printed Character Recognizer using Neural Network © 2000, Shahzad Malik Neural Networks and Fuzzy Logic. © 1995, Rao, V., Rao, H.MIS Press, New York
[10] Neural Networks © 2003, StatSoft Inc. http://www.statsoft.com/textbook/stneunet.html