A Comparative Study of Random Hidden Node Networks for Pattern Recognition

Eyal Niv
Department of Electrical and
Computer Engineering
University of Canterbury
Christchurch, New Zealand
Email: eyal.niv@pg.canterbury.ac.nz

Steve Weddell
Department of Electrical and
Computer Engineering
University of Canterbury
Christchurch, New Zealand
Email: steve.weddell@canterbury.ac.nz

***Abstract*—Computer vision attempts to replicate certain processes, such as object recognition, classification, and perception, naturally inherent within a mammalian visual cortex, in terms of simplified structures that can be emulated by evolutionary computation defined within the discipline of machine learning. For object recognition and classification problems, the study of receptive fields has provided insight into how a stimulus can alter the firing of neurons associated with the visual system. Through the adaptation of specialised, reservoir-based artificial neural networks, stimuli can be used to generate a spectral response that is analogous to a receptive field and used for classification. In this study, image receptive field artificial neural networks are evaluated for object classification and a comparison is made between reservoir-based leaning structures and more traditional feed-forward configurations.**

I. Introduction

Artificial neural networks (ANNs) have been used over several decades for object recognition and classification problems. ANNs provide an alternative to more traditional methods, such as Bayesian and rule-based classifiers [1], and are generally based on back-propagation algorithms. ANNs are often applied to problems where the parameter space is quite complex, i.e., solutions resolved in high-dimensionality space are required. However, a significant disadvantage is that training methods are non-linear. Adaptation of weights typically use gradient decent based algorithms and these can be extremely slow, especially for large datasets.

Reservoir-based computing [2] has simplified training of recurrent neural networks (RNNs). Such architectures employ a randomised hidden-node configuration that effectively requires no training. Early examples are echo state networks (ESNs) [3], which have been used in a variety of engineering applications, and liquid state machines (LSMs) [4], that have been applied to neurotechnical problems. However, whilst reservoir computing has made significant contributions to machine learning, in terms of simplified training and performance, their recurrent architecture is more suitable for prediction rather than classification; this is due to the recurrent nature of such networks, i.e., temporal information is captured through recurrent pathways. However, reservoir-based computing has recently been applied to object classification in the form of image receptive field neural networks (IRF-NNs) [5]. This study compares the performance and training times of IRF-NNs with traditional feed forward neural networks (FF-NNs), firstly by generating a simple object database, and secondly, by employing a handwritten digit library for classification.

In Section II an overview of IRF-NNs is given. Training and verification datasets used in this study are given in Section III. A comparative study of IRF-NNs with a more conventional feed-forward architecture is given in Section IV, and the results of this work are presented in Section V. Lastly, a conclusion and planned future work is discussed in Section VI.

II. Image Receptive Fields Neural Networks

It has been suggested that image receptive fields neural networks (IRF-NNs) are based on reservoir computing architectures, such as ESNs and LSMs. However, such architectures are in contrast with more traditional feed-forward artificial neural networks. This section explores the background of reservoir computing, and more specifically, details the training approach taken with IRF-NNs for object classification.

1) Background: The basis of reservoir computing is the generation of a *hidden* network layer, often referred to as a *state matrix*. In terms of an ESN, a sparse, pseudo-randomly matrix is created, where the dynamics of the network is based on the spectral radius, $\rho(\mathbf{W})$, defined as the magnitude of largest Eigenvalue within the sparse matrix. However, IRF-NNs use a set of randomly generated Gaussian functions, $g(\cdot)$, and input data, $\mathbf{u}$, to map the input to higher dimensional space. This domain can be represented by a state vector, $\mathbf{h}$, of dimension, M. Image data $\mathbf{u}$, which was generated by reshaping the image to a vector of dimension, D, is presented to the network. A succession of N images, $\mathbf{U}$, is used to generate a state matrix, $\mathbf{H}$. Such a construct is shown in Figure 1.

2) Training: The training of reservoir-based ANNs is simplified due to the incorporation of a randomised weight matrix that projects the inputs into higher dimensional space. Rather than employ back-propagation, as would be required in a feed-forward network, a series of randomised functions are used to enhance separability of input data, thereby facilitating object classification without the need to train a hidden layer

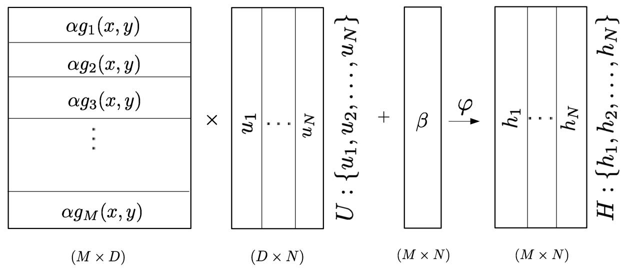

Fig. 1. Generation of a state matrix, $\mathbf{H}$.

using back-propagation. However, the adaptation of a weight matrix is required for *linear combination* to a series of output nodes. Since separability has been achieved in the hidden layer of M nodes, a weight matrix can be trained to represent a generalised solution for object classification using a simple linear regression algorithm.

Training of this network is divided into two phases. The first phase is the generation of M random hidden nodes [6], referred to here as a state vector. The state vector, $\mathbf{h}$, is essentially formed by the dot product of sampled image data, $\mathbf{u}$, and a randomised weight matrix, $\mathbf{G}$. This result is scaled, offset and a non-linear response is achieved, typically using a sigmoidal activation function, $\varphi(\cdot)$. The resulting activation vector is given as

$$\mathbf{h} = \varphi(\alpha \mathbf{G} \cdot \mathbf{u} + \beta), \tag{1}$$

where, $\varphi(\cdot)$ for this implementation was $\tanh(\cdot)$, the scaling function α was dataset dependant, the offset function $\beta = 0$, and the randomised activation matrix, $\mathbf{G}$, generated using a set of Gaussian vectors, $\mathbf{g}$, is defined as

$$\mathbf{G} = \begin{bmatrix} \mathbf{g_1} \\ \mathbf{g_2} \\ \vdots \\ \mathbf{g_M} \end{bmatrix} = \begin{bmatrix} g_1(x,y) \\ g_2(x,y) \\ \vdots \\ g_M(x,y) \end{bmatrix}. \tag{2}$$

In order to train the network without applying feature extraction, a set of Gaussian vectors defined in Equation 2 was generated using the method described by Daum et al., [5],

$$g_i(x,y) = \exp\Big[- \left(\frac{x - \mu_{xi}}{n_x \sigma_{xi}}\right)^2 - \left(\frac{y - \mu_{yi}}{n_y \sigma_{yi}}\right)^2 \Big], \tag{3}$$

where σ_i, and μ_i are randomly defined, and n_x and n_y are the width and height of the image, respectively.

The second training phase requires the adaptation of an output weight matrix to ensure trained input patterns projected to the state vector, $\mathbf{h}$, are asserted to the appropriate output node defined by vector, $\mathbf{s}$. Hence, the relationship between weight matrix, $\mathbf{W}$, a given output, and the state vector, $\mathbf{h}$, is given by

$$\hat{\mathbf{s}}_n = \mathbf{W}\mathbf{h_n}, \tag{4}$$

where output vector $\hat{\mathbf{s}}_n$ is of dimension, C, and is the estimated classification for image $\mathbf{u_n}$.

In order to solve Equation 4, specifically in terms of output weight matrix, $\mathbf{W}$, supervised learning is required. The state vector, defined as $\mathbf{h}$ in Equation 1, is trained on image data, $\mathbf{u}$. Given the previously generated state matrix, $\mathbf{H}$, in addition to known *a priori* classifications, $\mathbf{S}$, linear regression is performed, such that,

$$\mathbf{W} = \mathbf{S}\mathbf{H}^{\dagger}, \tag{5}$$

where $\mathbf{H}^{\dagger}$ represented the pseudoinverse of matrix $\mathbf{H}$.

A custom, object database was generated for network training and verification. This is discussed in Section III.

III. Training and Verification Datasets

A. Simple Geometric Database

In order to test both network architectures for simple affine transformations, a training and verification dataset was generated based on a 30×30 pixels binary image data. Each image in the dataset presents a scaled version of a translated simple geometric shape as shown in Figure 2.

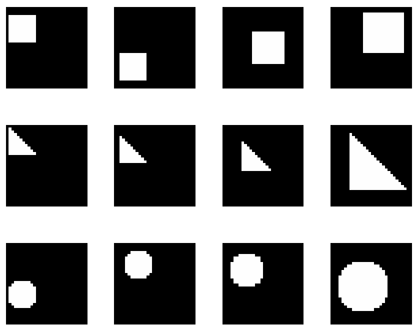

Fig. 2. Dataset samples.

The entire dataset included more than 9000 samples per shape and the training subset included 4500 random samples per shape. For verification, the entire dataset was used; hence, both networks presented in Section IV were verified with images they were trained for and also with new images. In order to address generalisation, both networks were initialised, trained and verified 10 times with a different random training subset each time.

B. MNIST Handwritten Digits Database

Both networks were also trained and tested using the MNIST database of handwritten digits. The MNIST database is particularly important as many different pattern recognition algorithms have been tested on it. A few samples from the MNIST database are presented in Figure 3.

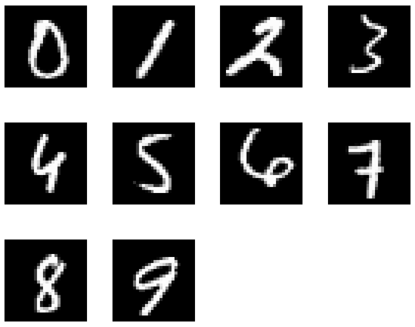

Fig. 3. MNIST samples.

IV. A Comparison of Neural Network Architectures

A conventional FF-NN was designed as shown in Figure 4 using a similar structure to the IRF-NN. Both networks had a similar number of neurons in the hidden layer and used the same sigmoidal activation function to pass the response to the next layer. The IRF-NN constrained the input weight matrix and adapted only the output weight matrix without any iterations as described in Section II; however, the FF-NN adapted both weight matrices while trying to minimise a cross entropy error. The FF-NN used a conventional cross entropy error function pairing for multi-class classification [6] given by

$$E = -\sum_{n=1}^{N}\sum_{k=1}^{C} t_{nk} \ln y_{nk} \tag{6}$$

where y_{nk} stands for the SOFTMAX function of the kth class on the nth input which is given by

$$y_{nk} = \frac{\exp(x_{nk})}{\sum_{j=1}^{C}\exp(x_{nj})} \tag{7}$$

and x_{nk} which stands for the network's input to the last SOFTMAX layer is given by

$$x_{nk} = \sum_{i=1}^{M} b_{ik} \tanh(\sum_{j=1}^{D} a_{ij} u_{nj}) \tag{8}$$

The minimisation algorithm chosen for the learning, i.e., the weight updates, was resilient back-propagation, which is considered a very fast weight mechanism update [7]. In order to avoid over-fitting of the FF-NN to the training data, an early stopping technique of a validation set was used. The validation set was composed of 20% of the samples in the training set. Repetitive increments of the cross entropy error for the validation set stopped the training.

V. Results

A. Simple Geometric Database

Figure 5 presents the classification rate and the mean training period for a series of 10 runs. The results show a monotonic increase in the classification rate of the IRF-NN with a classification rate of close to 100% for larger scale networks. The FF-NN performed well for smaller scale networks but performance remained relatively constant, irrespective of size. The standard deviation bars in Figures 5(a) and 5(b) suggest IRF-NNs are more robust than the conventional FF-NNs, and also that robustness increases with size. Furthermore, the training time of the IRF-NN indicates high stability and is significantly lower than that of the FF-NN. The differences in training times are explained by the relatively slow, iterative, weight adaptation process of the FF-NN. The IRF-NN uses a non-iterative approach to adapt only a portion of the network's weight matrix as mentioned in Section II and therefore, trains more than one order of magnitude faster. However, it is important to note that a better classification rate for the conventional FF-NN can be achieved by allowing a longer training time. One advantage of the FF-NN over the IRF-NN is the *posteriori* probabilities for the class distributions given by the SOFTMAX transformation. For a given input, each one of the output classes suggests the probability of membership.

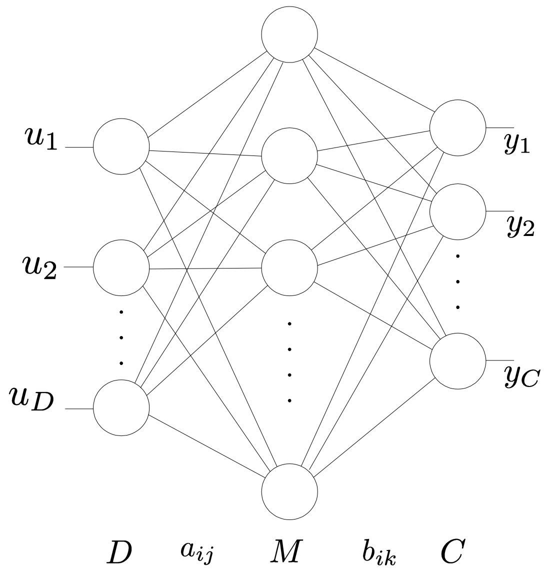

Fig. 4. Typical FF-NN architecture.

B. MNIST Handwritten Digits Database

A series of tests were conducted to optimise both configurations to work with the MNIST Database. The results are presented in Figure 6 and show similar classification rates with a major difference in training time. Further optimisation of the IRF-NN's classification rate is feasible by further optimising the receptive fields to work with the MNIST database.

VI. Conclusion and Future Work

The research presents a comprehensive background to the new idea of image receptive field neural networks and compares it with a more conventional feed forward neural network

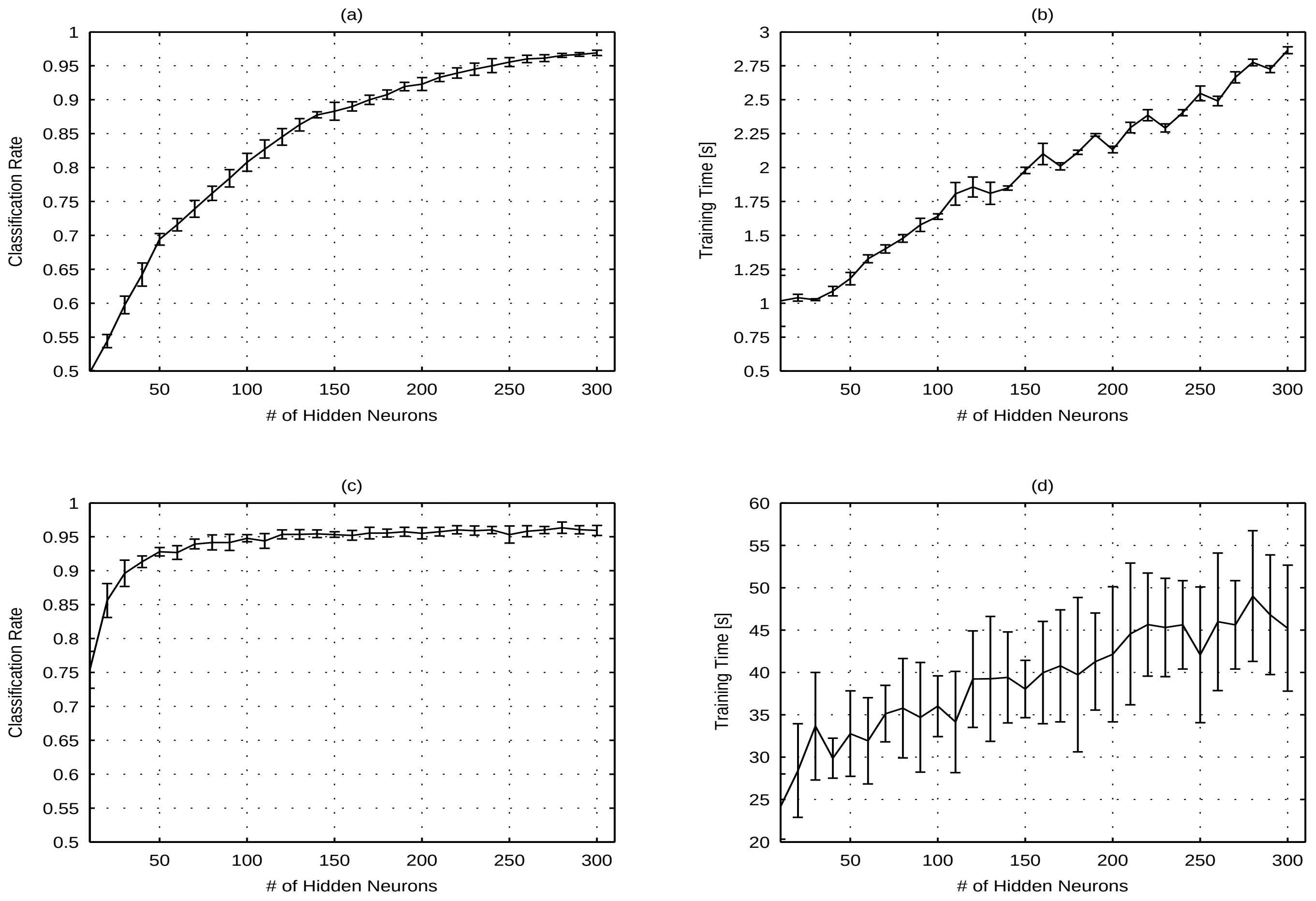

Fig. 5. Comparison of the Neural Networks on the geometric database: (a) Classification rate for IRF-NN. (b) Train time for IRF-NN. (c) Classification rate for FF-NN. (d) Train time for FF-NN.

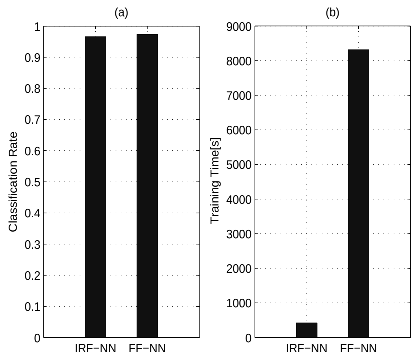

Fig. 6. Comparison of the Neural Networks on the MNIST database: (a) Classification rate. (b) Train time.

for the purpose of classification of simple objects. While both architectures present high classification rates, the IRF-NN has proved to be faster and more robust for the dataset which was used. In the future, it would be interesting to extend the comparison of both architectures with a more complex dataset and also include another reservoir type network such as the support vector machine.

REFERENCES

[1] J. Han, *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[2] B. Schrauwen, D. Verstraeten, and J. V. Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the 15th European Symposium on Artificial Neural Networks*, 2007, pp. 471–482. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.95.1215

[3] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," 2003. [Online]. Available: citeseer.ist.psu.edu/jaeger03adaptive.html

[4] W. Maass, T. Natschláger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[5] P. Daum, J.-L. Buessler, and J.-P. Urban, "Image receptive fields neural networks for object recognition," in *ICANN (2)*, ser. Lecture Notes in Computer Science, T. Honkela, W. Duch, M. A. Girolami, and S. Kaski, Eds., vol. 6792. Springer, 2011, pp. 95–102.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. Springer, 2007.

[7] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS*, 1993, pp. 586–591.