11-2015

Gaussian Nonlinear Line Attractor for Learning Multidimensional Data

Theus H. Aspiras
University of Dayton

Vijayan K. Asari
University of Dayton, vasari1@udayton.edu

Wesam Sakla
Air Force Research Laboratory

Gaussian Nonlinear Line Attractor for Learning Multidimensional Data

Theus H. Aspiras[1], Vijayan K. Asari[1] and Wesam Sakla[2]
[1]*Department of Electrical and Computer Engineering, University of Dayton, U.S.A.*
[2]*Air Force Research Laboratory, Wright Patterson Air Force Base, U.S.A.*

Keywords: Nonlinear Line Attractor, Multidimensional Data, Neural Networks, Machine Learning.

Abstract: The human brain's ability to extract information from multidimensional data modeled by the Nonlinear Line Attractor (NLA), where nodes are connected by polynomial weight sets. Neuron connections in this architecture assumes complete connectivity with all other neurons, thus creating a huge web of connections. We envision that each neuron should be connected to a group of surrounding neurons with weighted connection strengths that reduces with proximity to the neuron. To develop the weighted NLA architecture, we use a Gaussian weighting strategy to model the proximity, which will also reduce the computation times significantly. Once all data has been trained in the NLA network, the weight set can be reduced using a locality preserving nonlinear dimensionality reduction technique. By reducing the weight sets using this technique, we can reduce the amount of outputs for recognition tasks. An appropriate distance measure can then be used for comparing testing data and the trained data when processed through the NLA architecture. It is observed that the proposed GNLA algorithm reduces training time significantly and is able to provide even better recognition using fewer dimensions than the original NLA algorithm. We have tested this algorithm and showed that it works well in different datasets, including the EO Synthetic Vehicle database and the Sheffield face database.

1 INTRODUCTION

Much research work has been done on modeling brain functions and activities. The brain has around 10^{11} neurons and each neuron has up to ten thousand synaptic connections to other neurons around it. The brain has a unique ability to utilize groups of these neurons to formulate different lobes and structures to regulate bodily functions, emotional responses, and even cognition. All of these structures are able to unify together to handle numerous inputs from the different senses and have the ability to respond simultaneously.

When developing neurons and synaptic weights of the brain, models for these must have a set structure, an ability for firing/nonfiring of neurons through the synaptic junctions, training of the structure, and continuous training of the system. Artificial neural networks are the most explored models of the human brain, which encompass different modeling techniques of the brain. One area of neural networks is the recurrent neural network, which are able to send outputs to the same stage of the network. This type of neural network is also called an attractor network, which attracts towards a certain pattern. In this paper, we will be exploring attractor networks and how to improve them.

1.1 Attractor Neural Networks

There are different types of attractor networks which can model different dynamics of networks. One type is the fixed point attractor, which is the Hopfield network (Hopfield, 1982). Given an underlying energy function for minimization, the network asymptotically approaches a desired state. These networks are associative, meaning that they can approach towards a specific state given only part of the data. Other types include the cyclic attractor networks to govern oscillatory behaviors (Lewis and Glass, 1991). The type we will be focusing on is the line attractor (Zhang, 1996).

Line attractor networks work well for states that are not just a specific point but a line of points. Point attractor networks disregard any ability to attract to a specific trained image of a class, unless there are multiple point attractors for one class. Line attractor network assume continuity between data points, which allows an estimation of data points that are not fully trained in the dataset. Since manifolds in

Aspiras, T., Asari, V. and Sakla, W..
Gaussian Nonlinear Line Attractor for Learning Multidimensional Data.
In *Proceedings of the 7th International Joint Conference on Computational Intelligence (IJCCI 2015) - Volume 3: NCTA*, pages 130-137
ISBN: 978-989-758-157-1

a high-dimensional space are usually nonlinear, it is best to incorporate a nonlinear line attractor to model the manifold in space. Thus in this paper, we utilize the nonlinear line attractor (NLA) network as the supporting architecture for our work.

1.2 Biological Implications

With the use of recurrent associative networks, most are totally interconnected, meaning that every node is connected to all other nodes. This type of architecture allows influence of all nodes, especially if there are significant changes in input at farther nodes. But in biological structures, nodes are only interconnected with surrounding nodes with longer, weaker connections with farther nodes. These types are expressed by Tononi et al. (Tononi et al., 1999), who found that there is much redundancy in highly interconnected networks. They introduced an optimized degenerative case, which minimizes the amount of connections while maintaining the cognitive ability and reducing redundancy.

By using only neighborhood connections, portions of the network will be responsible for modeling that region while relying on the propagation of influence from other sections to travel as the network iterates. This local modeling will aid in faster convergence due to reliance to only closer nodes and give way to higher variance areas, which contain more information than other areas of input, for example a background region. This also reduces the amount of redundancies in the network for modeling a specific portion of data. Guido et al. (Guido et al., 1990) found that there is functional compensation when the visual cortex, which is highly modular, is damaged.

1.3 Modularity in Neural Networks

By developing neighboring connections, we can create modularity while still keeping inter-connectivity in the networks. Happel et al. (Happel and Murre, 1994) investigated various interconnections and modularity in networks and found that different configurations aided in further recognition of different tasks. Other techniques also used modularity, like modular principal component analysis (Gottumukkal and Asari, 2004), which takes specific portions of an image and takes the PCA of those sub-images to aid in the recognition of the whole image. Modularity reduces the model complexity which uses specific modules to learn only a portion of data to aid in the overall complex task (Gomi and Kawato, 1993).

Instead of fully incorporating modularity, algorithms also include overlap between modules to improve the recognition capabilities. Auda et al. (Auda and Kamel, 1997) used Cooperative Modular Neural Networks with various degrees of overlap to improve various classification applications. Also since modularity reduces the amount of connections between neurons by dividing processing into smaller subtasks, the amount of computations can be greatly reduced.

The type of neural network we will be looking at is the nonlinear line attractor, proposed by Seow et al. (Seow and Asari, 2004), which has been used for skin color association, pattern association (Seow and Asari, 2006), and pose and expression invariant face recognition (Seow et al., 2012). Given that modularity is able to reduce computation complexity and improve recognition in many cases, we aim to incorporate it into the nonlinear line attractor network. Instead of complete modularity, we propose a smooth, Gaussian weighting strategy to make smooth overlaps for each module. According to the weighting scheme of the network, modularity will also reduce the complexity of the modeling of the network. In (Seow et al., 2012), the weighting scheme is reduced using Nonlinear Dimensionality Reduction. We will look into the ability of the algorithm to reduce the weights and propose an improvement to the algorithm.

The main contributions of this paper are:

- Gaussian weighting strategy to the Nonlinear Line Attractor Network to introduce modularity
- Reduction of the computational complexity to improve the convergence time of the GNLA architecture
- An improved scenario for using the Nonlinear Dimensionality Reduction for object recognition

2 METHODOLOGY

The nonlinear line attractor network is a recurrent associative neural network, which aims to converge on a trained pattern given an input. Each trained pattern has connective information, which links one degree of information to another. These trained patterns usually are learned as a point in the feature space and thus variations of the pattern would be represented as a basin. Convergence would specify that the input pattern would be associated to one of the learned patterns. When considering a basin of attraction, single point representations of a particular set of patterns may be insufficient to totally encompass the patterns. The NLA architecture formulates a nonlinear line representation which would allow patterns to converge towards the line attractor. An example would be using manifold learning on a set of objects

with varying poses. As the poses move in the high-dimensional space, the manifold would form a non-linear line, which would be best modeled by the NLA architecture.

Let the response x_i of the i^{th} neuron due to the excitations x_j from other neurons for the s^{th} pattern in a fully connected recurrent neural network with n neurons be expressed as:

$$x_{(i,s)} = \frac{1}{N}\sum_{j=1}^{N}\Lambda_i(x_{(j,s)}) \text{ for } 1 \le i \le N \tag{1}$$

where Λ_i is defined by a k^{th} order nonlinear line as:

$$\Lambda_i(x_{(j,s)}) = \sum_{m=0}^{k} w_{(m,ij)}(x_{(j,s)})^m \text{ for } 1 \le i,j \le N \tag{2}$$

This equation defines a polynomial best fit line to encompass all input/output pairs given from each pattern s. The green line in Figure 1 shows this best fit line. The m^{th} order term of the resultant memory w_m can be expressed as:

$$w_m = \begin{pmatrix} w_{(m,11)} & \cdots & w_{(m,1N)} \\ \vdots & \ddots & \vdots \\ w_{(m,N1)} & \cdots & w_{(m,NN)} \end{pmatrix} \text{ for } 0 \le m \le k \tag{3}$$

The weights are in matrix form to show a fully interconnected weight system. Since the input and output have the same number of nodes due to the associative nature of the algorithm, we can show these weights in this regard. To calculate the weights, we can use error descending characteristics. The least squares estimation approach is able to calculate the best fit line using the polynomial method. To minimize the least squares error in the weight matrix, we can formulate the following equation, which yields the optimum weight set.

$$E_{ij}[w_{(0,ij)}, w_{(1,ij)}, ..., w_{(k,ij)}] = \sum_{s=1}^{P}[x_{(i,s)} - \Lambda_i(x_{(j,s)})]^2 \text{ for } 1 \le s \le P \tag{4}$$

To minimize the least squares error, we must equate the derivative of the error with respect to the weight to be zero, as shown in the following equation.

$$\frac{\delta E_{ij}}{\delta w_{(m,ij)}} = 0 \text{ for each } m = 0, 1, \ldots, k \tag{5}$$

We can then find that the equation can be reduced to a set of linear equations based on the order of the polynomial, as shown below.

$$\begin{aligned}
&w_{(0,ij)}\sum_{s=1}^{P}(x_{(j,s)})^0 + w_{(1,ij)}\sum_{s=1}^{P}(x_{(j,s)})^1 + \ldots \\
&\quad + w_{(k,ij)}\sum_{s=1}^{P}(x_{(j,s)})^k = \sum_{s=1}^{P}x_{(i,s)}(x_{(j,s)})^0 \\
&w_{(0,ij)}\sum_{s=1}^{P}(x_{(j,s)})^1 + w_{(1,ij)}\sum_{s=1}^{P}(x_{(j,s)})^2 + \ldots \\
&\quad + w_{(k,ij)}\sum_{s=1}^{P}(x_{(j,s)})^{k+1} = \sum_{s=1}^{P}x_{(i,s)}(x_{(j,s)})^1 \\
&\quad \vdots \\
&w_{(0,ij)}\sum_{s=1}^{P}(x_{(j,s)})^k + w_{(1,ij)}\sum_{s=1}^{P}(x_{(j,s)})^{k+1} + \ldots \\
&\quad + w_{(k,ij)}\sum_{s=1}^{P}(x_{(j,s)})^{2k} = \sum_{s=1}^{P}x_{(i,s)}(x_{(j,s)})^k
\end{aligned} \tag{6}$$

Given that there is a nonlinear line that models the relationship between inputs, as modeled by the trained weight sets, there must be another modeling of the variances of the data. This is done by creating an activation function, as shown in equation 7.

$$\Phi\{\Lambda[x_j(t+1)]\} = \begin{cases} x_i(t) \text{ if } \psi_{(ij,-)} \le \{\Lambda_i[x_j(t+1) - x_i(t+1)\} \\ \qquad \le \psi_{(ij,+)} \\ \Lambda_i[x_j(t+1)] \text{ otherwise} \end{cases} \tag{7}$$

where

$$\Lambda_i(x_j(t)) = \sum_{m=0}^{k} w_{(m,ij)}(x_j(t))^m \tag{8}$$

An activation function can be used to see if a data point that runs through the system is trained into the manifold. Since data points do not exactly fit in the best fit nonlinear line, thresholds $[\psi_{(ij,-)}, \psi_{(ij,+)}]$ are created to ensure that data points do not update if the estimated data point lies within the manifold. If the data point falls away from the manifold, the best fit equation can be used to attract the data point towards the manifold. These threshold regions can be expressed as:

$$\psi_{(ij,-)} = \begin{cases} \psi_{(1,ij,-)} & \text{if } 0 \le x_j < \frac{L}{\Omega} \\ \psi_{(2,ij,-)} & \text{if } \frac{L}{\Omega} \le x_j < \frac{2L}{\Omega} \\ \qquad \vdots & \\ \psi_{(\Omega,ij,-)} & \text{if } (\Omega-1)\frac{L}{\Omega} \le x_j < L \end{cases} \tag{9}$$

$$\Psi_{(ij,+)} = \begin{cases} \Psi_{(1,ij,+)} & \text{if } 0 \leq x_j < \frac{L}{\Omega} \\ \Psi_{(2,ij,+)} & \text{if } \frac{L}{\Omega} \leq x_j < \frac{2L}{\Omega} \\ \vdots & \\ \Psi_{(\Omega,ij,+)} & \text{if } (\Omega-1)\frac{L}{\Omega} \leq x_j < L \end{cases} \tag{10}$$

where L is the number of segments used for the piecewise threshold regions and Ω is the length of the manifold. When using a nonlinear dimensionality reduction technique, as shown in a further section, the threshold regions created will not be used, due to the nonlinear line attractor becoming a transform from the original image space to a reduced dimensionality space. Figure 1 shows how the weights are interconnect through the inputs.

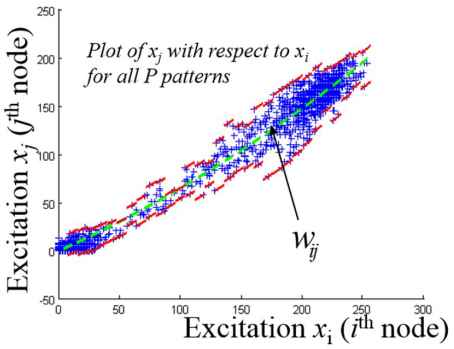

Figure 1: Interconnection of Weights. The green line captures the nonlinear line modeling and the red lines capture the variances of the data.

Since this is modeling of a specific manifold, multiple manifolds may be needed to encompass more of a class. Most will require a specific manifold per class, so there will be at least one weight set per class.

2.0.1 Computational Strategy

We have devised an effective computational strategy for training data. Given equation 6, previous models of the computational strategy require computing powers for each interconnection, in which there are several calculations that are repeated in the equations while traversing through each interconnection. Instead of having redundant calculations, we can divide the weight calculation into different steps.

Stage 1 would be the calculation of powers for the inputs. In equation 6, we see that every x_j^s has an order associated with it. Calculation of these terms would be redundant for all different combinations of inputs and outputs, since there are multiple terms with the same order, hence the same value. Computing only the powers in this stage would tremendously reduce the computation time of the system.

Stage 2 would be the calculation of the weights, given the set of normal equations and using the values obtained from stage 1. The solving of the normal equations can be done using a linear solve algorithm. This stage will take a considerable amount of time due to the volume of data, specifically the number of inputs, since the weight matrix size is $\#inputs \times \#inputs \times order$ where # inputs refer to the size of the image.

Stage 3 would be the calculation of the activation function, which will also require a considerable amount of computation time. Since the computation of the orders for all of the input data are already known, the activation function can be formulated using that data.

Stage 4 would be the calculation of the nonlinear dimensionality reduction. This step would require all of the weights and is dependent on the number of inputs and also the order of the weight system.

2.1 Gaussian Nonlinear Line Attractor (GNLA) Network

The Gaussian Nonlinear Line Attractor Network is a modification to the original NLA network, which uses a neighborhood approach to improve the algorithm. Local information is more important than distant information, when looking from a biological perspective, so it can be assumed that using this architecture for the NLA algorithm would improve run times and classification ability. When dealing with imagery, a 2-dimensional spatial relationship is created with each node. By incorporating this spatial relationship into the nodes, we can increase the recognition while reducing computation time.

When implementing a Gaussian neighborhood approach, we can change the coefficient in the front and add the distance equation. The equation can then be modified as:

$$x_{(i,s)} = \sum_{j=1}^{N} \alpha_{ij} \Lambda_i(x_{(j,s)}) \text{ for } 1 \leq i \leq N \tag{11}$$

where

$$\alpha_{ij} = exp\left(-\left(\frac{(\hat{x}_i - \hat{x}_j)^2}{2\sigma_{\hat{x}}^2} + \frac{(\hat{y}_i - \hat{y}_j)^2}{2\sigma_{\hat{y}}^2}\right)\right) \tag{12}$$

For these equations, $\hat{x}$ and $\hat{y}$ define the spatial coordinates of the input x. Instead of using the Gaussian function, we can use the Gaussian kernel , for example a 13x13 Gaussian kernel as shown in Figure 2. This will effectively reduce the computation time. We can then change the equation as

0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	1	2	3	4	3	2	1	0	0	0
0	0	1	4	9	15	18	15	9	4	1	0	0
0	1	4	13	29	48	57	48	29	13	4	1	0
0	2	9	29	67	111	131	111	67	29	9	2	0
1	3	15	48	111	183	216	183	111	48	15	3	1
1	4	18	57	131	216	255	216	131	57	18	4	1
1	3	15	48	111	183	216	183	111	48	15	3	1
0	2	9	29	67	111	131	111	67	29	9	2	0
0	1	4	13	29	48	57	48	29	13	4	1	0
0	0	1	4	9	15	18	15	9	4	1	0	0
0	0	0	1	2	3	4	3	2	1	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0

Figure 2: An example of a 13 x 13 Gaussian kernel.

$$x_{(i,s)} = \sum_{j=1}^{N} A_{ij}\Lambda_i(x_{(j,s)}) \text{ for } 1 \leq i \leq N \tag{13}$$

Where n is the size of the kernel and A_{ij}

$$A_{ij} = exp\left(-\left(\frac{(\hat{x}_i - \hat{x}_j)^2}{2\sigma_{\hat{x}}^2} + \frac{(\hat{y}_i - \hat{y}_j)^2}{2\sigma_{\hat{y}}^2}\right)\right) \tag{14}$$

We can also effectively reduce the computation time of the kernel by not computing any portion that contains zeros. According to the Gaussian kernel above (which is a 13x13), roughly 28% of the Gaussian kernel are zeros, thus a reduction of the computation time can be accomplished by ignoring those computations.

2.1.1 Nonlinear Dimensionality Reduction

To reduce the size of the weight sets, we can reduce the dimensionality of the weights. Since the weights obtained through training have embedded the manifolds, we can use the weights in the reduction process. Given that there are r different line attractor networks, there will be y different outputs, as shown in the following equation.

$$\begin{aligned} Y_1 &= W_{1,k}X^k + W_{1,k-1}X^{k-1} + \cdots + W_{1,0}X^0 \\ Y_2 &= W_{2,k}X^k + W_{2,k-1}X^{k-1} + \cdots + W_{2,0}X^0 \\ &\vdots \\ Y_r &= W_{r,k}X^k + W_{r,k-1}X^{k-1} + \cdots + W_{r,0}X^0 \end{aligned} \tag{15}$$

Previous algorithms of the Nonlinear Line Attractor use singular value decomposition (SVD) (Golub and Reinsch, 1970) to reduce the weight set. We propose using a different strategy to reduce the weight set. Given that the locally linear embedding (LLE) (Roweis and Saul, 2000) algorithm contains interconnected weight sets for different datapoints, we can use that strategy to reduce the weights of the NLA network.

Each m^{th} term of the networks' memory is evaluated using the LLE algorithm. We first obtain a sparse matrix M using the following equation.

$$\begin{aligned} M_{(m,d)} =& (I - w_{(m,d)})' * (I - w_{(m,d)}) \\ & \text{for } 0 \leq m \leq k;\ 1 \leq d \leq r \end{aligned} \tag{16}$$

We then take the smallest z eigenvectors from M and use them as the projection into the lower-dimensional subspace. The projection of the N-dimensional data to a z-dimensional subspace using a $z \times N$ sub-matrix obtained from the smallest z eigenvectors of the LLE yields a z-dimensional output Y_m' where $z << N$. Once the lower-dimensional subspace is obtained, we can then use an appropriate distance measure, like Euclidean distance, to find the closest match of a test image to the trained images in the system.

When calculating the weight matrices for the Gaussian NLA network, none of the weights contain any information of the Gaussian distances. To incorporate the function inside the weights, we must embed the terms into the weights by multiplying the normalized mask with the weights. Given equation 16, we can embed the normalized coefficients to multiply the weights using the following equation.

$$a^s = \begin{pmatrix} a^s_{(11)} & \cdots & a^s_{(1N)} \\ \vdots & \ddots & \vdots \\ a^s_{(N1)} & \cdots & a^s_{(NN)} \end{pmatrix} \tag{17}$$

The resultant multiplication of the weight set is given by the equation below.

$$\begin{aligned} &\bar{w}_{(m,s)} = \\ &\begin{pmatrix} \bar{w}_{(m,11,s)} & \cdots & \bar{w}_{(m,1N,s)} \\ \vdots & \ddots & \vdots \\ \bar{w}_{(m,N1,s)} & \cdots & \bar{w}_{(m,NN,s)} \end{pmatrix} \\ &\qquad \text{for } 0 \leq m \leq k \end{aligned} \tag{18}$$

This nonlinear dimensionality reduction technique using LLE computes each order of the weight matrices separately. There is no interconnection between the orders, in which the original algorithm contains a summation between the orders to results in the final outputs. We propose that instead of reducing this dimensionality by adding the orders of the matrix for the results, we can concatenate all of the orders, which are multiplied by the input, to yield a larger vector

without the reduction resulting from the summation. This will allow even better recognition results using fewer total weights, even though the resulting vector can be potentially larger.

2.1.2 Complexity

In Stage 1, we can find that it will be the same complexity as the previous algorithm since we are just computing the powers.

In Stage 2, the original complexity is $\#inputs \times \#inputs \times order^2$ due to the linear solve algorithm, but modified complexity is $\#inputs \times \#neighbors \times order^2$, where # neighbors is significantly smaller than the size of the network. For example, given that we have a network of 60x80, which is 4800, and an order of 4 for the polynomial, the complexity of the algorithm becomes $4800 \times 4800 \times 4^2 = 368640000$ computations. If we create kernel of size 13×13, we would have a complexity of $4800 \times 169 \times 4^2 = 12979200$ computations, which is 3.52% the computation time of the original. If we reduce the kernel by taking out all zeros in the function, we would have a complexity of $4800 \times 121 \times 4^2 = 9292800$ computation, which is only 2.52% the computation time of the original and 71.6% the computation time of the kernel.

In Stage 3, the complexity should be reduced just as stage 2. In Stage 4, the complexity should be the same as the previous algorithm. Table 1 shows the run times on the EO Synthetic Vehicle dataset. It is found that the kernel algorithm without using zeros provides faster training times than all of the other algorithms.

3 RESULTS

3.1 Datasets

Various datasets are being used to test the validity of using the Nonlinear Line Attractor network for usage in classification and modeling tasks. One dataset is a EO Synthetic Vehicle Database, which contains several different cars, various lighting conditions, and multiple viewing angles, as shown in Figure 3. This dataset tests the algorithms ability to model an entire vehicle manifold across these different scenarios. The other dataset is the Sheffield (previously UMIST) database, which contains several images of faces with varying poses. For all datasets, a 13x13 kernel is used for training and testing the dataset.

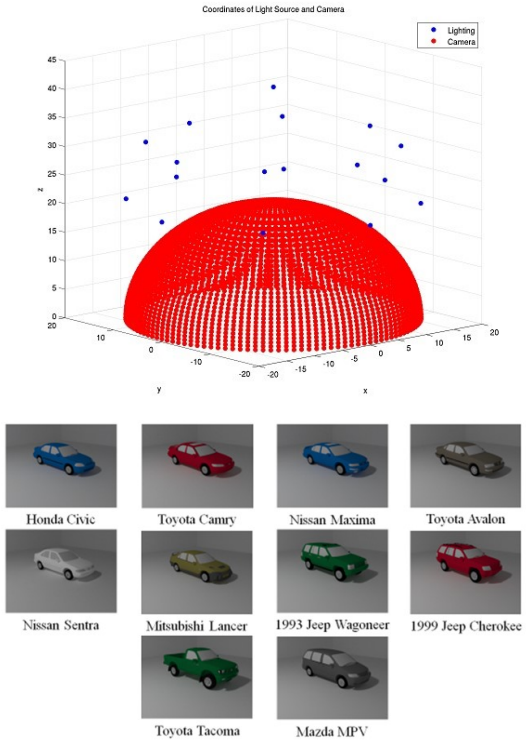

Figure 3: EO Synthetic Vehicle Database. The top image shows the different camera and lighting positions of the database and the images below show the different vehicles.

3.2 Results 1

The first dataset is the EO Synthetic Vehicle database. The results are shown in Table 2. We used a full 360 view of each vehicle and one lighting condition to train and test the validity of the technique. Weight matrices were trained with all views and tested according to the closest distance between each point (Euclidean distance). It is observed that by concatenating all of the results from the nonlinear dimensionality reduction, we are able to keep some of the distinction found in the orders rather than summing them. When we start adding more principal components obtained from the the method, we are able to increase the recognition result. It is observed that the GNLA architecture and concatenating the orders is able to decrease the number of principal components while keeping the recognition result of the vehicles high.

3.3 Results 2

The second dataset we use is the Sheffield database. The results are shown in Table 3. We split the whole database into two for testing and training sets. Once we have trained the network with the training set, we then perform nonlinear dimensionality reduction

Table 1: Training times for each stage of the algorithms.

Runtime	Original	Current	Kernel	Kernel (No zeros)
Stage 1	N/A	15.2 min	15.2 min	15.2 min
Stage 2	1280 min	95.3 min	3.35 min	2.4 min
Stage 3	87.5 min	87.5 min	3.08 min	2.21 min
Stage 4	10.8 min	10.8 min	10.8 min	10.8 min
Total Runtime	1378.3 min	208.8 min	32.43 min	30.61 min

Table 2: Recognition results on the EO Synthetic Vehicle Database.

# Prin. Comp.	NLA Reg.	NLA Concat.	GNLA Reg.	GNLA Concat.
1	47.50	60.56	35.83	43.89
2	59.72	66.94	82.78	76.94
3	86.67	79.72	63.06	85.28
4	61.39	85.00	76.94	88.61
5	63.89	86.11	81.67	88.89
6	69.17	88.06	85.00	91.11
7	66.94	87.50	86.39	91.39
8	75.28	88.89	87.22	92.22
9	72.22	86.39	91.39	95.00
10	78.61	87.22	91.94	94.72
All	99.19	85.28	99.44	99.17

Table 3: Recognition results on the Sheffield Database.

Algorithm	% Recognition
PCA	86.87
KPCA	87.64
LDA	90.87
DPCA	92.90
DCV	91.51
B2DPCA	93.38
GNLA	93.33
GNLA (concat)	94.07

and then found the closest match using Euclidean distance. We have found that the GNLA architecture works well to recognize the different faces. It is observed that the GNLA network with concatenating the orders produced the best results. One of the major components about the GNLA architecture is that it runs much faster than the previous architecture. Run time for one instance of the NLA architecture for this dataset takes 8.2 hours, while an instance of the GNLA architecture will take 0.8 hours.

3.4 Discussion

The biggest concern when using associative networks is that these algorithms work best using orthogonal data. Given a large dataset that has only small variances between different classes while having large variances within the classes, the data is much harder to classify with the network. Ideally, the network should have all orthogonal inputs so that modeling and classification of the data are much easier.

With the GNLA architecture, we are able to increase orthogonality by reducing the number of inputs sent to a given neuron. This is done by only using a given area for computation of the algorithm. If the input image is not any of the trained images, we will see that the image will diverge in the portions of the data that have the highest variance. Portions of the image that contain background information will not be able to converge or diverge since those sections do not vary.

More importantly, depending on the relationship created by the NLA architecture for the node connections, some relationships have a stronger discerning ability than others, meaning that the nonlinear line that has been created to model the relationship of the nodes should have the smallest error (smallest deviation from the created line). For example, if the image wants to converge to one of the trained images, we must allow those portions with stronger discerning ability to function more than those portions that are unable to converge or diverge from the input.

Since the NLA architecture allows a summation of the relationships of the different nodes to estimate one specific node, we must allow the portions that have more discerning ability to be forefront in the weighting scheme. This is why the Gaussian NLA architecture is able to reduce the run times of the algorithm and increase the convergence rate. Sections of the object region are assumed to have high amounts of variability for the relationship created by the NLA in

that specific area to all have high discerning ability, thus allowing better convergence of trained images and better divergence characteristics for untrained images.

4 CONCLUSIONS

The proposed GNLA architecture using the concatenation of the orders and the locality preserving algorithm works well for recognition tasks and for reducing the amount of computation time. Previous architectures are much slower and have decent recognition results, but by incorporating spatial information in the weighting of the architecture, we are able to increase the recognition results for a smaller number of dimensions while reducing the computation time. We plan to improve the weighting performance by exploring different weighting schemes to weight those portions of the network that are able to produce significant results.

REFERENCES

Auda, G. and Kamel, M. (1997). Cmnn: cooperative modular neural networks for pattern recognition. *Pattern Recognition Letters*, 18(11):1391–1398.

Golub, G. H. and Reinsch, C. (1970). Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420.

Gomi, H. and Kawato, M. (1993). Recognition of manipulated objects by motor learning with modular architecture networks. *Neural Networks*, 6(4):485–497.

Gottumukkal, R. and Asari, V. K. (2004). An improved face recognition technique based on modular pca approach. *Pattern Recognition Letters*, 25(4):429–436.

Guido, W., Spear, P., and Tong, L. (1990). Functional compensation in the lateral suprasylvian visual area following bilateral visual cortex damage in kittens. *Experimental brain research*, 83(1):219–224.

Happel, B. L. and Murre, J. M. (1994). Design and evolution of modular neural network architectures. *Neural networks*, 7(6):985–1004.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Lewis, J. E. and Glass, L. (1991). Steady states, limit cycles, and chaos in models of complex biological networks. *International Journal of Bifurcation and Chaos*, 1(02):477–483.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Seow, M.-J., Alex, A. T., and Asari, V. K. (2012). Learning embedded lines of attraction by self organization for pose and expression invariant face recognition. *Optical Engineering*, 51(10):107201–1.

Seow, M.-J. and Asari, V. K. (2004). Recurrent network as a nonlinear line attractor for skin color association. In *Advances in Neural Networks–ISNN 2004*, pages 870–875. Springer.

Seow, M.-J. and Asari, V. K. (2006). Recurrent neural network as a linear attractor for pattern association. *Neural Networks, IEEE Transactions on*, 17(1):246–250.

Tononi, G., Sporns, O., and Edelman, G. M. (1999). Measures of degeneracy and redundancy in biological networks. *Proceedings of the National Academy of Sciences*, 96(6):3257–3262.

Zhang, K. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory. *The journal of neuroscience*, 16(6):2112–2126.