Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 95 (2016) 159 – 167

Procedia Computer Science

Complex Adaptive Systems, Publication 6
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2016 - Los Angeles, CA

Implementing Multilayer Neural Network Behavior Using Polychronous Wavefront Computation

Fred Highland[a]*, Corey Hart[b]

[a]*University of Maryland Baltimore County, Baltimore, MD, USA*
[b]*nNetworx, LLC, Philadelphia, Pennsylvania, USA*

Abstract

Polychronous Wavefront Computation (PWC) is an abstraction of spiking neural networks that provides a potentially practical model for implementing neuromorphic computing systems. While it's has been shown to exhibit some basic computational capabilities, its use in complex neuro-computational models remains to be explored. The paper presents a model and approach for configuring PWC transponders to implement multilayer neural network behavior to provide a basis for more complex applications of the technology. The model uses a set of input transponders representing pattern features to stimulate hidden layer transponders that combine features and trigger output layer transponders to identify patterns. The input layer transponder geometry is selected to create wavefront intersections for all relevant feature combinations. Hidden layer transponders are positioned by solving the intersection of the circles equations defined by sets of input transponders. Output layer transponders are defined to collect complete sets of features for recognition based on the hidden layer transponder geometry. The approach uses the intersections of three wavefronts to maximize transponder selectivity and increase information density. The concept is experimentally demonstrated and analyzed with a 7-segment display digit recognition application which provides a simple but representative example of more complex pattern recognition problems.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: polychronous wavefront computation; multilayer perceptrons; spiking neural networks

* Corresponding author. Tel.: +0-301-471-4685.
E-mail address: fred.highland.mobile@gmail.com

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2016.09.307

1. Introduction

Polychronous Wavefront Computation (PWC)[1] was proposed as an abstraction of the spiking neural network paradigm[2] based on temporal and spatial patterns of wavefront activity in a pulse propagating media and their interaction with transponders. It provides a potentially practical model for implementing neuromorphic computing systems, that operate on the same principles as the human brain, by eliminating the need for direct connections between computational units simplifying implementation. Such neuromorphic systems offer the potential for advanced pattern recognition and computation capabilities that are adaptive and require less power than von Neumann approaches.

Izhikevich and Hoppensteadt[1] have defined small configurations of PWC transponders that can perform computations such as signal analysis and logical operations. The mathematical properties of simple PWC configurations have been explored in detail and some conceptual sensor configurations suggested[3]. It has also been suggested that numerical programming methods could be used to configure PWC transponders[4]. Although prior work has defined many fundamental properties of PWC configurations, specific methods and design patterns to construct multilayer configurations capable of complex pattern recognition have not been developed. Such methods would support further exploration of complex PWC properties; inform efforts to develop supervised and unsupervised learning techniques; and provide the basis for researching advanced applications such as deep learning.

The objective of this work is to develop methods to create multilayer PWC configurations that can form the basis for exploring the application of PWC to more complex problems. The paper presents an approach for configuring PWC transponders to implement multilayer neural network recognition behavior. The behavior is similar to a multilayer perceptron model but has the spiking neural network characteristics of PWC. The model features a set of input transponders to represent feature values, hidden layer transponders that combine features values, and an output layer of transponders to represent the results. The paper first discusses the strategy for implementing such a configuration then goes into the details of configuration of the input, hidden and output layers. It then discusses the application of the approach to a 7-segment display digit recognition problem and the results obtained. It concludes with a discussion of the characteristics of the proposed method and directions for further research.

2. Design Approach

2.1. Strategy

Recognizing complex patterns from a set of features using a multilayer configuration of transponders (nodes) requires the definition of a set of input nodes representing the features, hidden nodes that combine subsets of the input features and output nodes to classify feature sets. Since the relevant set of feature combinations is not known a priori, the hidden nodes must be capable of representing all possible combinations of the inputs and the output nodes must be capable of identifying all relevant input combinations. Representing this with PWC raises a number of issues including node location, efficiently combining the inputs together in hidden nodes and avoiding unwanted node activations in large scale configurations.

In the simplest case, two PWC nodes can activate a third node representing the combination of the two. Figure 1 shows how two nodes (A and B), activated at different times, trigger a third node (C) at the intersection of two wavefronts (circles). The possible wavefront intersections over time form a hyperbola defined by the radii of the wavefronts generated by the two nodes, which in turn are defined by the positions and relative activation times of the nodes. In this example, node A was activated before node B resulting in a larger diameter wavefront for node A. The intersection points for the A and B activation are defined by the solid hyperbola. Other possible intersections, defined by different relative activation times, are shown as dashed hyperbolae.

Applying the two wavefront intersection activation criteria to larger configurations creates an intersection hyperbola for each pair of activated nodes resulting in $(n^2-n)/2$ hyperbolae for an n node configuration. Each curve extends to infinity in each direction producing a large number of possibilities for unwanted node activations that can result in chaotic behavior. This can be controlled by proper spacing and choice of position but the number of intersection hyperbolae in complex configurations makes this approach difficult to manage.

A better approach, which is analyzed by Thomas[5], is to use a criteria based on the intersections of three wavefronts. The intersection of three wavefronts is defined by the intersection of the intersection hyperbolae of each of the three

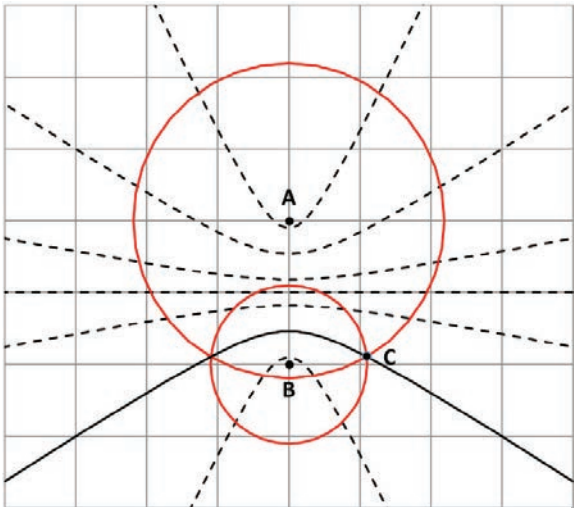


Figure 1 - PWC Two Node/Wavefront Activation

node pairs. Although this results in $(n^3-3n^2+2n)/6$ intersections for an n node configuration, each intersection is a single point in space significantly limiting the possibility of unwanted node activations and increasing the potential information density of the intersections and the configuration.

The intersection region of three nodes is roughly the interior of a triangle formed by the nodes[5] suggesting a circular pattern for input node arrangement. If the nodes are arranged in a circle with simultaneous activation times, the intersections are always in the center, which does not provide the needed discrimination between combinations. However, if the activation times are offset, the intersections occur in an approximately circular region of the interior of the circle defined by the nodes and can produce unique intersection locations for all possible combinations of inputs with the right choice of input positions and time offsets.

The three node intersection approach uniquely defines hidden node location based on the input node combinations and activation times. Similarly, combinations of hidden nodes can uniquely define the location of output nodes. The indirect control over position and time of the hidden nodes may limit flexibility and feasibility of output nodes but if output nodes can be defined by multiple hidden node combinations, the limitations can be minimized.

Other input configurations may also be possible. Semi-circles are a variant on full circles but their geometry reduces the interior space of three node triangles packing the intersection points more densely which is undesirable. Thomas[5] has suggested a matrix form with input (sensor) points evenly distributed in two dimensions. This has similar characteristics to the circular pattern but can generate intersection points over input nodes due to linear arrangement of some combinations. This problem may be solved using time and/or position offsets. Patterns that use greater than three wavefront intersections could further improve the density of the configurations. However, input geometries that generate all combinations of higher order intersections have not been identified.

2.2. Input Nodes

As mentioned above, a circular pattern of feature inputs with offset times can produce a geometry that contains all possible three node intersections. A constant time delay between possible input features can be used to simplify the computations as discussed below.

In general, each feature must be represented as multiple input nodes to provide all possible values of the feature. Since the hidden nodes are activated by input node wavefronts, the absence of a value for a feature is not the same as a default or 0 value. Therefore, a node must be provided for each feature-value combination or relevant partition of values. At this stage in the research, only binary values have been used. Continuous valued features have not been explored but there are a number of possible representations (see 4 Discussion).

The size of the input layer circle and the time delay between input signals have a direct impact on the spacing of the hidden nodes. Roughly, the hidden nodes form in a semicircle in the middle of the input node circle with a minimum radius approximately equal to the time delay (assuming wavefront propagation of one distance unit per time unit). The outer position of the hidden nodes is a function of the input node position and delay time but the density of hidden nodes is higher toward the center of the input node circle. This characteristic can be exploited in positioning output nodes to minimize conflicts. The distribution of hidden nodes using this pattern is not strictly a circle but rather a set of closely packed spirals. As a result, large time delays can position hidden nodes significantly outside the input circle which may not be feasible.

2.3. Hidden Nodes

Hidden nodes are defined to represent each possible combination of three inputs feature values. Given multiple mutually exclusive values for each feature, the number of hidden nodes for three wavefront intersections is:

$$n_{hidden} = \binom{f}{3} v^3 = \frac{f!}{3!(f-3)!} v^3 \quad (1)$$

Where

f is the number of input features
v the number of values for each feature

The position of each hidden node is determined by the equation of the intersection of two circles[6]:

$$x_{hidden} = \frac{x_2+x_1}{2} + \frac{(x_2-x_1)(r_1^2-r_2^2)}{2d^2} \pm \frac{y_2-y_1}{2d^2}\sqrt{\left((r_1+r_2)^2-d^2\right)\left(d^2-(r_1-r_2)^2\right)} \quad (2)$$

$$y_{hidden} = \frac{y_2+y_1}{2} + \frac{(y_2-y_1)(r_1^2-r_2^2)}{2d^2} \mp \frac{x_2-x_1}{2d^2}\sqrt{\left((r_1+r_2)^2-d^2\right)\left(d^2-(r_1-r_2)^2\right)} \quad (3)$$

$$d = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \quad (4)$$

$$\textit{Subject to the constraints: } r_1 + r_2 > d \textit{ and } |r_1 - r_2| < d \quad (5)$$

Where

x_n,y_n are the coordinates of the input node i_n
d is the distance between the input nodes
r_n is the radius of the circle (wavefront) from input node i_n

Figure 2 illustrates the relationship of r to time and distance in determining these intersections. Three nodes (A, B and C) that are activated at different times (t_A, t_B and t_C respectively), in sequence (i.e., $t_A < t_B < t_C$) and with wavefront (shown as circles) radii r_A, r_B and r_C respectively. The dashed lines (actually curves), AB, AC and BC represent the intersection hyperbolae for the corresponding nodes. The point ABC is the intersection of the three wavefronts. The relationship of the wavefront radii and activation times in this example is:

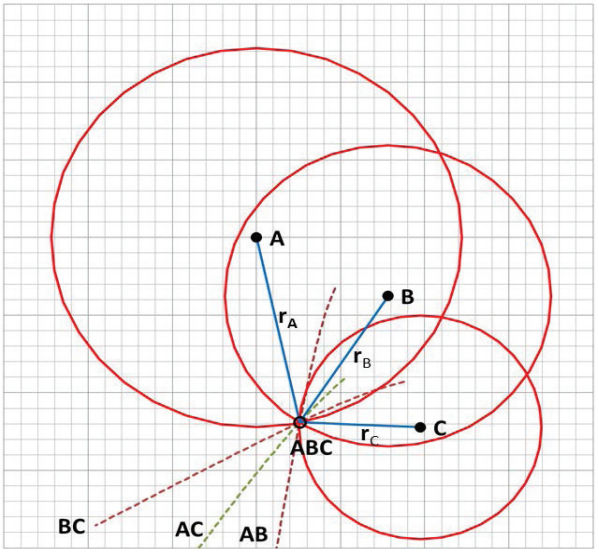


Figure 2 - Relationship of r, distance and time in 3 wave intersections

$$r_A = r_B + (t_B - t_A) = r_C + (t_C - t_A) \quad (6)$$

This dependence can be exploited by defining r_A and r_B in terms of r_C (since t_A, t_B and t_C are given) providing a single variable for optimization.

Thomas[5] showed that the location of the intersection of three wavefronts is the intersection of the intersection hyperbolae for each of the pairs of nodes. However, the equations of circle intersection (2 & 3) have two solutions for each pair of input nodes representing the branches of the hyperbola on either side of the line between the nodes. Since there are three node pairs that define a three wavefront intersection, there are $2^3 = 8$ possible solutions to the intersection point, if it exists, for positive values of r. Depending on the locations of the nodes and the values of r, there may be no solutions (i.e. the constraints of (5) are violated) or the solutions may be impractical due to extreme distances involved. This can be a problem for determining the hidden node locations if the input time delays are too large relative to the diameter of the circle of inputs.

The feasibility of the triplets of intersection equations depends on the orientation of the input nodes and the timings involved. An analytic solution for this has not yet been derived but the equations can be easily solved using gradient decent to find the value for r_C (from (6)) that minimizes the distance between pairwise intersections across all possible solutions.

2.4. Output Nodes

In the general case, output nodes should represent all relevant combinations of input feature values. This can include all input features or subsets of them as dictated by the recognition problem. The node locations can be defined using the same approach used for hidden nodes by selecting a set of three hidden nodes, representing the features required, and finding the intersections of their pairwise intersection hyperbolae. The number of output nodes required depends on the nature of the problem. In general, it should be the maximum defined by the number of meaningful feature combinations to allow analysis or training methods to prune unneeded hidden and output nodes as suggested in 4 Discussion.

For an output node to represent an input feature set, it must be activated from hidden nodes that represent only the required features. Since the output feature set is formed from the union of the features of the three hidden nodes, each of which represents three input features, there are multiple possible hidden node combinations representing feature sets less than or equal to nine. As long as each of the input values is represented, no additional values are included and the hidden nodes used are unique, any triplet of hidden nodes can be used. Table 1 shows the number of combinations of three unique hidden nodes available to define output nodes based on the number of input features. The feasibility of these combinations is limited by the constraints of (5), the positions of the hidden nodes and the implied values of r (6). Only one combination of hidden nodes that represents the features set and meets constraints is needed for each output so the number of hidden nodes needed can be significantly less that the number in Table 1.

Using combinations of three feature hidden nodes to define outputs limits feature representation to nine per output. For problems that require more than nine features, additional hidden node layers can be added to combine the features. Each additional layer increases the number of features represented by up to a factor of three over the previous layer. As this also increases the number of hidden nodes and the effort required to compute these nodes exponentially, it is best to limit the scope of processing for each configuration as is done in deep learning approaches.

The determination of the solutions, with respect to r, for the equations defining output nodes is best accomplished using gradient descent. A selection criterion should be used to limit the search since there are many possible combinations of hidden nodes that can define output node locations. Selection of candidates can be pruned by considering the constraints of (5) along with objectives on the distance between the solution and the center of the input circle to minimize unwanted node activations.

2.5. Approach Summary

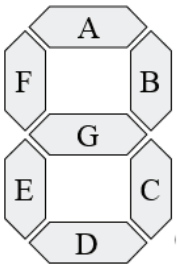
The approach is summarized in the following steps:

Table 1 - Hidden Node Combinations vs. Input Features.

Input Features	Hidden Node Combinations
4	4
5	100
6	480
7	945
8	840
9	280

1. Define position and timing for input nodes representing each possible input value. A circular arrangement with fixed offset times between features is recommended.
2. Define hidden nodes positions for each possible combination of three input nodes based on the solutions of the equations of circle intersection using gradient descent.
3. Select output nodes that represent combinations of features from the intersections of three hidden nodes containing those features. Select the hidden node triplets that best define the output nodes based on feasibility of solving the circle intersect equations and the output node location relative to other nodes.

Once the configuration is defined, various methods may be used to optimize the configuration by removing unneeded nodes as discussed in 3 Results.



Decimal	Segments
0	ABCDEF
1	BC
2	ABDEG
3	ABCDG
4	BCFG
5	ACDFG
6	ACDEFG
7	ABC
8	ABCDEFG
9	ABCDFG

Figure 3 - Standard Label Nomenclature for a 7-Segment Display

3. Results

To evaluate the proposed method, a simple 7-segment display digit recognition problem was implemented. 7-segment display devices have been used to display decimal digits on electronic devices since about 1910[7]. They are driven by decoders that convert binary numbers to a decimal representation of seven illuminated segments. The problem used here is to input the seven segments (the standard notation is shown in Figure 4) and categorize them into one of ten outputs labeled 0-9. This problem represents a simple version of a multi-feature classification problem providing a practical basis for detailed experimentation and analysis.

The inputs were encoded as seven features of two mutually exclusive values (segment “on” or “off”). Input nodes were defined evenly distributed around a circle with a radius of 100 units. The time delay between each input feature’s nodes was defined as 6 units. The “off” value units were positioned halfway between the “on” value units but activated at the same time as the feature values are mutually exclusive. The “on” value nodes were activated in a clockwise sequence and the “off” value nodes in a counterclockwise sequence to better distribute the hidden nodes in the center region of the circle minimizing conflicts.

Output nodes were selected based on: satisfaction of constraints of (5), distance from the center (less than 150% of the input circle radius), and minimizing interference with hidden nodes (distance to center greater than 25% of the input circle radius). If unwanted output node activations occurred during training (due to proximity of nodes), alternate solutions (hidden node combinations) were selected.

A spreadsheet model was created to define the node configuration and analyze the solution. This supported input node layout, computation of hidden node locations, evaluating output node alternatives and analysis of the problem.

Figure 5 shows the node layout diagram for the resulting configuration. Inputs are shown in the outer circle with black dots representing on values (labeled with upper case letters per standard label nomenclature) and circled grey dots representing off values (labeled with lower case letters). Placement of the hidden nodes is shown by small red dots (unlabeled). The location of hidden nodes is defined by the input node and tend to be concentrated near the center although some outliers can be seen. The blue diamonds are the output nodes labeled with the numbers values they represent. Only correct output labels are shown for simplicity. Output nodes were selected from possible alternatives and occur outside central region of the circle to minimize potential conflicts with hidden nodes.

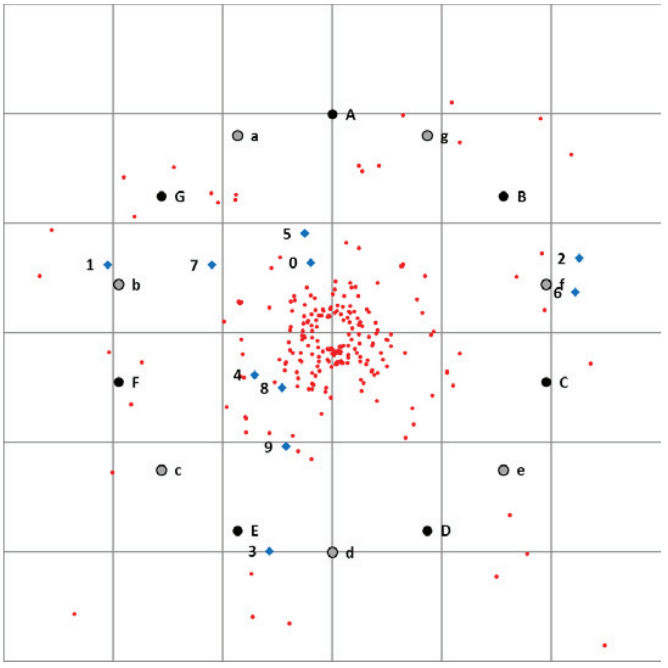


Figure 4 - 7-Segment Recognition Configuration Layout.

The resulting node configuration was run on a PWC Simulator built on NetLogo[8]. The simulator implements PWC transponders in a dynamic agent-based discrete-event simulation that allows the wavefront and transponder actions to be visualized. The simulation uses basic time and distance equations that do not require the circle intersection analysis of (3) thru (5) providing an independent verification of correct behavior. The simulator also supports Leaky Integrate-then-Fire[9,10] semantics on the nodes allowing the simulation to be tuned to adjust for node distances.

Tests were run with a series of ten patterns representing the digits 0-9 (Positive Training Cases) and twenty randomly generated invalid (non-digit) patterns (Negative Training Cases). Note that in the standard nomenclature, there are only ten valid input patterns. Positive Cases were verified to activate only the appropriate digit output nodes and none of the non-digit output nodes (100% accuracy). Negative Cases were verified not to activate any digit output nodes (0% errors). If incorrect output activations occurred, alternate hidden/output node combinations were selected to correct the issues.

Table 2 - Summary Node Statistics for Positive Training Cases

Node Type	Count	Subcategory	Count
Input	14		
Hidden	280	Activated	174
		Unactivated	106
		Used by Outputs	76
		Unused by Outputs	204
Output	128	Activated	10
		Unactivated	118
Total	422	Activated	168
		Unactivated	224
		Used	100
		Unused	322

Table 2 shows the summary statistics for the positive training cases for hidden and output nodes. The data is subcategorized into activated and used nodes. Activated nodes were activated by the training data. Used nodes are prerequisites for outputs according to analysis of the node configuration data and unused nodes are not associated with any positive output. These two measures are an indication of possibly unneeded nodes that could be removed to optimize the configuration. The overall counts match the predictions of the equations above. Not all hidden nodes are activated because many combinations of feature-values are not used in the positive training cases. Also, a large number of hidden nodes are not used in the definitions of output nodes. Some of the hidden nodes are activated even though they are not used in the definition of outputs because their input combinations are activated.

4. Discussion

The approach is applicable to a wide range of problems but may require significant computing resources to generate configurations. The computational complexity of transponder definition is dominated by the nature of the outputs but driven by the number of features. The cost of defining the input layer linear with the number of feature values required based on the arrangement of input nodes. The hidden layer definition cost is a driven by the combinations of three features that can be created from the inputs which is given by (1) and is $O(f^3v^3)$. Each hidden node requires the solution to multiple intersection equations ((2), (3) and (4)) using gradient descent which is finite because equations 2 and 3 are effectively quadratic but are not computationally trivial and feasibility constraints must be applied. The most expensive element is the determination of output node positions. In the worst case, outputs must be computed for all possible input combinations which is $O(f^v)$ and there can be as many as 945 combinations to consider based on the feature value set (see Table 1). Each of these requires a gradient descent solution of intersection equations. Fortunately, many of these will be infeasible and can be pruned using constraints of (5) and geometric selection criteria (distance from other nodes, etc.). The computational needs are tractable with modern computer resources but they are not trivial.

The current implementation represents only the equivalent of black and white (binary) light intensity vision similar to that provided by rod structures in the human eye[11]. It could be extended to color for other imaging problems by adding the analog of cone structures which sense three different wavelength ranges (approximating red, green and blue). Similar approaches could be used to represent more complex features with continuous values. In addition, the underlying transponder simulation uses a Leaky Integrate-then-Fire model that can also provide some representation of continuous values by summing inputs over time and adjusting node activation thresholds. Furthermore, PWC implementations can encode real valued features directly in terms of continuous time delays between different pulses.

The concept of reducing unneeded output nodes (and hidden nodes) suggests a method for learning by pruning the resulting configuration using methods such as potentiation decay[12]. By applying an attrition function during training cases, unused or infrequently used nodes can be suppressed optimizing the structure of the configuration. The statistics on the 7-segment example suggest this could reduce the number of nodes significantly since more than half of the nodes (224 of 422) were unactivated in the positive test cases. Those statistics also show that many other hidden nodes are not associated with outputs and are candidates for elimination. This can be determined by a post-training analysis of the results and can make a significant further reduction in the hidden node requirements.

The configurations generated do not exhibit all characteristics of multilayer perceptrons. Perceptrons use weighted connections that can define the best fit of a set of features. The spiking nature of this model along with the Leaky-Integrate-Then-Fire behavior can be provide a similar capability under some conditions but it is not as general as weighted connections. The multilayer PWC model tends to identify exact patterns based on input features which may be an advantage for some problems. The model also does not specify how to combine multiple outputs into classes (e.g., multiple patterns that represent the same digit). This can be done external to the PWC computations, using a logical OR mechanism within PWC[1] or possibly using multiple layers of this model. Methods to address this are currently being investigated.

One of the keys to the success of this work is the use of a three wavefront intersection criteria for PWC node activation. The two wavefront criteria results in intersection hyperbola covering a large portion of node space making the configuration subject to unwanted activations as the number of nodes increases. A three wavefront intersection criteria defines a point in space that is much less likely to result in unwanted activations as the configuration grows.

5. Summary and Directions for Further Work

A general approach has been presented for defining PWC configurations that exhibit multilayer neural network behavior. It has been successfully applied to a simple, but representative, pattern classification problem. Analysis of the results shows that the configurations produced can recognize input patterns and there are opportunities for optimization through node elimination. It is suggested that the results are applicable to a wide range of problems.

A more general approach to learning useful outputs may be to apply a Spike Timing Dependent Plasticity (STDP) approach to find output nodes locations[9]. This would involve randomly placing a sufficiently dense set of candidate output nodes within the configuration and letting the PWC STDP algorithm determine their optimal positions.

The current work has been focused on feed-forward configurations but the PWC framework is well suited for implementing recurrent and recursive behaviors. The approach described constrains the placement of transponders in order to minimize the likelihood of recurrent or reverberating excitation and interference between patterns. This provides a simplifying constraint but tasks with more computational complexity or temporal components may require some degree of recurrent excitation and therefore methods for determining placement while avoiding runaway activations. The use of recurrent configurations[12] may also provide an approach to unsupervised training of multilayer PWC configurations. If output node activation could be used to reactivate the associated hidden nodes, the recursion could potentiate the active nodes and a form of potentiation decay could be used to suppress the inactive nodes.

This work can provide the basis to explore advanced research topics in the area of PWC configurations including:

- Exploration of additional input node configurations
- Development of learning approaches to determine output and hidden node locations
- Construction of more complex configurations such as recurrent structures and category recognition
- Application to other problems such as approximate classification and deep learning approaches.

The objective of this work was to define a starting point for the understanding of multilayer PWC configuration to support complex pattern recognition. While it requires some intervention to adjust the output nodes for best performance, it provides a basis for the development of further methods for supervised and unsupervised learning in these configurations and support further exploration using the PWC framework in complex problems.

References

1. Izhikevich, E.M. and Hoppensteadt, F. Polychronous Wavefront Computations. International Journal of Bifurcation and Chaos; Vol. 19, No. 5:1733-1739. 2009.
2. Izhikevich, E.M. “Polychronization: Computation with spikes.” *Neural Computing*. 18, 245–282. 2006.
3. Thomas, J. A Mathematical Treatise on Polychronous Wavefront Computation and its Application into Modeling Neurosensory Systems. ResearchGate; 2014. https://www.researchgate.net/publication/264044389_A_Mathematical_Treatise_on_Polychronous_Wavefront_Computation_and_its_Application_into_Modeling_Neurosensory_Systems_Under_Review
4. Hart, C. Towards a Compiler for a Polychronous Wavefront Computer: Programming by Optimization. *Complex Adaptive Systems, Procedia Computer Science*; Vol. 36. 2014. http://www.sciencedirect.com/science/article/pii/S1877050914012599

5. Thomas, J. The Geometry of Polychronous Wavefront Computation. ResearchGate; 2015. https://www.researchgate.net/publication/281156794_The_Geometry_of_Polychronous_Wavefront_Computation
6. Wilson, R. S. Analytic Foundations of Geometry. Sonoma State University; 2010. http://www.sonoma.edu/users/w/wilsonst/papers/Geometry/default.html
7. Wood, F. W. U.S. Patent 974,943. Patent and Trademark Office; 1910.
8. Wilensky, U. NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. 1999. http://ccl.northwestern.edu/netlogo
9. Gerstner, W. and Kistler, W. *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge University Press; 2002.
10. Highland, F. and Hart, C. Adaptation of Spike-Timing-Dependent Plasticity to Unsupervised Learning for Polychronous Wavefront Computing. *Complex Adaptive Systems, Procedia Computer Science*; Vol. 61. 2015. http://www.sciencedirect.com/science/article/pii/S1877050915029762
11. Nave, C.R. Hyperphysics. Georgia State University; 2012. http://hyperphysics.phy-astr.gsu.edu/hbase/vision/rodcone.html
12. Miller, A and Jin, Dezhe Z. Potentiation decay of synapses and length distributions of synfire chains self-organized in recurrent neural networks. *Physical Review* E 88, 062716; 2013.