Interpretable Network Architectures for Machine Learning
(Patent Pending)

Randal Allen
Lone Star Analysis
Orlando, FL
rallen@lone-star.com

ABSTRACT

With limited success, artificial neural networks bring several disadvantages. These shortcomings are related to architectural selection (e.g., number of neurons, number of layers) which are dependent on the number of inputs and outputs and the complexity of the input-output relationship. Also, training methods may require additional neurons and layers, increasing the size of the network, and may lead to underfitting or overfitting, rendering the network useless beyond the data used for training and testing. The design process becomes an academic exercise in numerical investigation resulting in an untrusted "black box" where the designer has no influence over what is being learned. In the end, because of the depth of complexity, it's impossible to understand how conclusions were reached.

A system is needed with an architecture where the designer has control over what is being learned and thus provides inherent elucidation. This paper presents and discusses such a system architecture comprising a set of mathematical functions and logic gates lending transparency and explanation to applications based on artificial neural networks. A relatively simple example shows how the system architecture replaces the regression form of supervised learning to determine the aerodynamic rolling moment coefficient given aileron deflections, using much less data than required by traditional system identification methods. The paper concludes by discussing the implications this system architecture has on the other forms of machine and deep learning (classification and clustering), predictive and prescriptive analytics, and due to the inclusion of logic gates, quantum computation.

ABOUT THE AUTHOR

Randal Allen is the Chief Scientist of Lone Star Analysis. He is responsible for applied research and technology development across a wide range of MS&A disciplines. He holds a CMSP with NTSA. He is co-author of the textbook, "Simulation of Dynamic Systems with MATLAB and Simulink." He is an Associate Fellow of AIAA. He holds a Ph.D. in Mechanical Engineering (University of Central Florida), an Engineer's Degree in Aeronautical and Astronautical Engineering (Stanford University), an M.S. in Applied Mathematics and a B.S. in Engineering Physics (University of Illinois, Urbana-Champaign). He serves as an Adjunct Professor in the MAE department at UCF.

Interpretable Network Architectures for Machine Learning

Randal Allen
Lone Star Analysis
Orlando, FL
rallen@lone-star.com

MOTIVATION

There are several sources motivating interpretable network architectures. An MIT Technology Review on Intelligent Machines states, the U.S. Military wants its autonomous machines to explain themselves. The latest machine learning techniques are essentially *black boxes*. DARPA is funding efforts to open them up (Knight, W., 2017). DARPA also announced its Artificial Intelligence Exploration (AIE) program, a key component of the agency's broader artificial intelligence (AI) investment strategy aimed at ensuring the United States maintains an advantage in this critical and rapidly accelerating technology area (DARPA, 2018). In February, President Trump enacted the "American AI Initiative," identifying AI as a priority for government research and development. The next day, the DoD released its own AI strategy, positioning its newly created Joint Artificial Intelligence Center (JAIC) at the forefront of its efforts (Lamberth, M., 2019). The proposed system architecture removes the opaqueness of machine learning *black boxes* so practitioners can easily interpret the results, see if they correspond with intuition, and fully explain how the transparent system works.

BACKGROUND

Kaplan and Haenlein (2018) define Artificial Intelligence (AI) as "a system's ability to correctly interpret external data, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation." AI dates to the mid-1950s with times of promise followed by disappointment and lack of funding. However, AI has seen a resurgence due to increased computational power, the ability to manipulate large amounts of data, and an influx of commercial research funding.

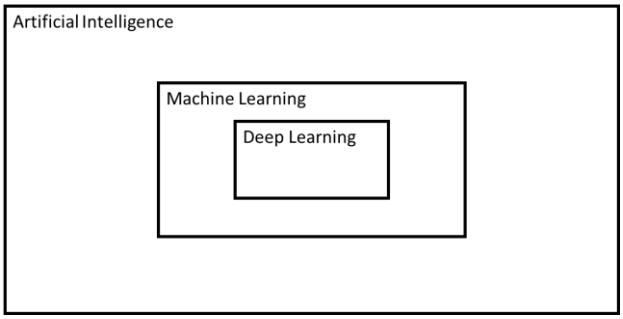

Figure 1: Artificial Intelligence, Machine Learning, and Deep Learning

For the purposes of this paper, assume machine learning is a subset of AI (Figure 1), with applications to image, speech, and voice recognition, and natural language processing. In business applications, machine learning may be referred to in the context of predictive analytics. Unlike computer programs which execute a set of instructions, machine learning is based on models which learn from patterns in the input data. A major criticism of machine learning models is that they are black boxes without explanation for their reasoning.

There are three types of machine learning which depend on how the data is being manipulated. The first type is supervised learning where a model is trained on known input and output data to predict future outputs. There are two subsets to supervised learning: *regression* techniques for continuous response prediction and *classification* techniques for discrete response prediction. The second type is unsupervised learning which uses clustering to identify patterns in the input data only. There are two subsets to unsupervised learning: *hard clustering* where each data point belongs

to only one cluster and *soft clustering* where each data point can belong to more than one cluster. Finally, the third type is reinforcement learning where a model is trained on successive iterations of decision-making, where rewards are accumulated based on the results of the decisions. A machine learning practitioner will recognize there are many methods to solve these problems, each having their own set of implement requirements. Figure 2 shows a sample of the machine (supervised/unsupervised) learning state of the art.

Regression	Classification	Soft Clustering	Hard Clustering
Ensemble methods	Decision trees	Fuzzy-C means	Hierarchical clustering
Gaussian process	Discriminant analysis	Gaussian mixture	K-means
General linear model	K-nearest neighbor		K-medoids
Linear regression	Logistic regression		Self-organizing maps
Nonlinear regression	naïve Bayes		
Regression tree	Neural nets		
Support vector machine	Support vector machine		

Figure 2: Machine Learning Methods

Current focus is on deep learning, a subset of machine learning (see Figure 1). Applications include face, voice, and speech recognition and text translation which employ the classification form of supervised learning. Deep learning gets its name from the multitude of cascaded artificial neural networks. Figure 3 shows a typical artificial neural network architecture used in machine learning. In its most basic form, the artificial neural network has an input layer, a hidden layer, and an output layer.

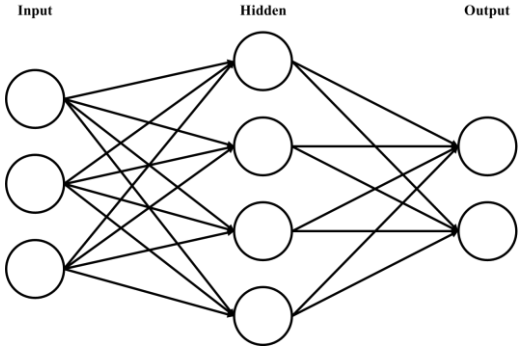

Figure 3: Artificial Neural Network

For deep learning applications, the more layers, the deeper the learning. Figure 4 shows a simplistic artificial neural network architecture used in deep learning where additional hidden layers have been added providing depth. In practice, deep learning networks may have tens of hidden layers.

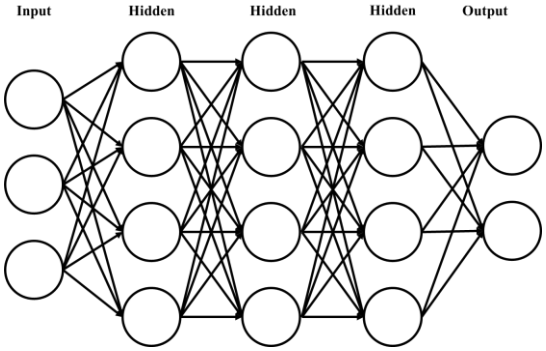

Figure 4: Artificial Neural Network for Deep Learning

As an example of the burden on the model designer, consider the application of supervised machine learning (classification) for object recognition or detection. The designer must manually select the relevant features to extract from the data, decide which classification method to use to train the model, and tune hyperparameters associated with

fitting the data to the model. The designer does this for various combinations of features, classifiers, and hyperparameters until the best results are obtained.

In the case of deep learning, the manual step of selecting the relevant features to extract from the data is automated. However, to accomplish this, thousands of images are required for training and testing. Also, the designer is still responsible for determining the features. In the end, even highly experienced data scientists can't tell whether a method will work without trying it. Selection depends on the size and type of the data, the insights sought, and how the results will be used.

While artificial neural networks are the basis for artificial intelligence, machine learning, and deep learning, there are problems associated with this technology. Significant issues include lack of transparency, depth of deep learning, under-fitting or over-fitting data, cleaning the data, and hidden-layer weight selection.

Because the artificial neural network was modeled (somewhat) after the human brain, it is difficult to see the connection between the inputs and outputs which leads to a lack of transparency. The designer is often unable to explain why one architecture is used over another. This unknown opaqueness leaves the user wondering if the architecture can be trusted. For the designer, architectural selection becomes an exercise in numerical investigation. Architectural choices naturally include the number of inputs and outputs but becomes artificial when hidden layers and corresponding nodes are added. The number of hidden layers and the number of nodes comprise the depth of deep learning and is arbitrary. If you happen upon an architecture that appears to work, congratulations, but good luck explaining why to the user. Furthermore, architecture selection is based on the number of hidden layers and nodes: too few may lead to under-fitting, whereas too many may lead to over-fitting. In both cases, the overall performance and predictive capability may be compromised. As an example, (Zhou, 2019) discusses how an artificial neural network may be pruned (from 50% all the way up to 99.5%) without impacting the model's performance. Clearly, there is a lot of unnecessary overhead, regarding the architecture and the data. This candid discussion leaves a few questions about the formulation of the problem/model that need to be answered.

- How were the original weights (that were subsequently pruned) selected?
- Why could the model be pruned up to 99.5% without impacting its performance?
- If the pruned neurons didn't matter, why were they there in the first place?
- What does the pruned network represent?

The current answer to all four questions is, "We don't know." With this, we can empathize with the U.S. Military wanting its autonomous machines to explain themselves.

Other problems with artificial neural networks are the need to clean the data and, seemingly arbitrary, weight selection. Why should some data (outliers) be omitted from the training or test set? Maybe there is a plausible reason for the outlier's existence and it should be kept because it represents reality. For instance, maybe the outlier represents what is known as a black swan – Nassim Taleb's metaphor for an improbable event with colossal consequences. The outlier should not be omitted simply to make the architecture more robust. Also, who is to say which weight factor should be placed on a hidden layer or set of nodes? Data cleansing and parameter tuning may lead to architectural fragility.

Disadvantages of Current Systems

Upon surveying the prior art associated with machine learning in general, machine learning practitioners will recognize the disadvantages of current methods. Refer to Figure 2 for a sampling of the state of the art, where each method has its own set of implementation requirements. In the case of supervised classification, the designer is required to manually select features, choose the classifier method, and tune the hyperparameters.

Deep learning brings with it, its own set of demands. Enormous computing power through high performance graphics processing units (GPUs) is needed to process the data, lots of data. The number of data points required is on the order of 10^5 to 10^6. Also, the data must be numerically tagged. Plus, it takes a long time to train a model. In the end, because of the depth of complexity, it's impossible to understand how conclusions were reached.

The mathematical theory associated with artificial neural networks is the Universal Approximation Theorem (UAT)[1] – which states a network, with a single hidden layer, can approximate a continuous function. Some practitioners rely on this too heavily and seem to ignore the assumptions associated with this approach. For example, as seen in Figure 4, a relatively simple deep learning model has more than a single hidden layer. By implementing a deep learning model with multiple hidden layers, the UAT assumption is grossly violated. Also, for practical applications serving state of the art technologies, problem complexity surely increases. Once a model has been built, the architect may not be entirely sure the mathematical functions are continuous – another violation of UAT assumptions. While increasing the number of neurons may improve the functional approximation, any improvement is certainly offset by the curse of dimensionality. In other words, while additional neurons (for a single hidden layer) may improve the functional approximation, by increasing the number of hidden layers, the number of neurons increases accordingly. Other version of the UAT come with their own limitations. In one version, linear outputs are assumed. In another version, convex continuous functions are assumed. Finally, the UAT itself says nothing about the artificial neural network's ability to *learn*! The artificial neural network architecture supporting machine/deep learning is supposedly inspired by the biologic nervous system. The model learns through a process called back propagation which is an iterative gradient method to reduce the error between the input and output data. But humans don't back-propagate when learning, so the analogy is weak in that regard. That aside, more significant issues are its black box nature and the designer having no influence over what is being learned.

Therefore, a system is needed with an architecture where the designer has control over what is being learned and thus provides inherent elucidation. This architecture must be innovative and avoid the pitfalls of artificial neural networks with their arbitrary hidden layers, iterative feature and method selection, and hyperparameter tuning. The system must not require enormous computing power, it should quickly train and run on a laptop. Depending on the application, data tagging, while necessary, should be held to a minimum. Lastly, the system must not require thousands of (cleaned) data points.

TECHNICAL APPROACH

Where the current state of the art creates a connection between two sets of data with a multitude of nodes, layers, and arbitrarily simple functions, the proposed system architecture instead inserts a curated set of lucid mathematical functions between the two sets of data. This is a fundamental difference in that mathematical nonlinearities, and/or nonconvexities, and/or discontinuities can more quickly be approximated to reveal relationships between the two sets of data.

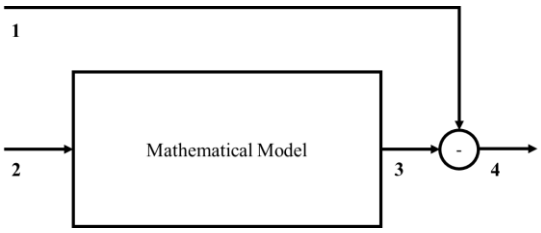

Figure 5: System Architecture

Referring to the system architecture of Figure 5, signal (2) is sent to the mathematical model yielding output signal (3). The error signal (4), which is the difference between the feedforward signal (1) and the output signal (3), is minimized. The mathematical model may be generic or specific, depending on the application. If available, a practitioner with area expertise should incorporate *a priori* knowledge into the design of the mathematical model. For example, if the problem is associated with mechanical vibration, then the mathematical model should include Fourier sine and cosine terms. Minimization of the error signal (4) is achieved through optimization techniques. Through this process, signal (3) is forced to match signal (1) by adjusting parameters associated with the mathematical model.

This approach is unique in that it serves as a unifying system architecture among the many varied specialized sciences, including machine/deep learning. For example, in supervised learning (classification), output is related to input. Referring again to Figure 5, one manifestation of the system architecture solves this type of problem by connecting known *input* data to signal (2) and known *output* data to signal (1). In supervised learning (regression), output is

[1] https://en.wikipedia.org/wiki/Universal_approximation_theorem

related to output. Another manifestation of the system architecture solves this type of problem by connecting known *output* to both signal (1) and signal (2). For both supervised learning cases, parameters associated with the mathematical model are varied until the computed result matches the known result. In the case of unsupervised learning (clustering), another manifestation of the system architecture solves this type of problem by connecting the known *input* to both signal (1) and signal (2). By minimizing the error, signal (3) will match signal (1) and thus, characterize the input data based on the mathematical model. While these manifestations leverage the same system architecture, only the assignment of signal (1), signal (2), and the mathematical model architecture differ.

Details

To understand how the system operates, consider a manifestation of the system architecture where the designer has no *a priori* knowledge about the relationships of the data. In this case, assume the mathematical model contains generic mathematical functions such as a second-order polynomial, sine and cosine terms, an exponential function, and a logarithmic function, $\boldsymbol{a_0 + a_1 x + a_2 x^2 + \cdots + b_s\, sin(nx) + b_c\, cos(nx) + c\, exp(\, nx) + d\, ln(\, nx)}$. (Other manifestations may involve different mathematical functions and operations, including classical (Boolean) or quantum logic gates. To guard against such discontinuities, an optimization algorithm is employed which avoids partial derivatives and their associated numerical instabilities.)

[Referring to I/ITSEC abstract 19109, entitled "Adaptive Nonconvex Optimization for AI and Machine Learning," a system of methods which reaps the benefits of both grid search and random search, without their corresponding limitations, is uniquely combined with a method of multipliers to produce an approach to solving general nonconvex optimization problems. At its core, independent random variables adapt themselves to produce a finer search for an extremum. Because the system is gradient-free, the architecture allows for logic gates with implications for machine learning and quantum computing.]

The coefficients $\boldsymbol{a_0, a_1, a_2, b_s, b_c, c, d}$ may be random variables between 0 and 1 and weighted such that they sum to 1. Because the system architecture is designed to minimize the differential error between some computed quantity and a known quantity, the coefficients are changed to place different weights on each of the mathematical functions. Since the coefficients are random variables, their adaptation (over multiple Monte Carlo iterations) is probabilistic. All the statistics are available such that the designer can explore any set of coefficients for interesting (rare condition) cases. Nominally, however, the designer selects the median coefficient values which define a transparent, interpretable, and explainable relationship between the known input and the computed output. The system architecture is self-defined because the coefficients are determined empirically. There is no need for the designer to perform a numerical investigation of trial and error as in the case for artificial neural nets. The system architecture is transparent, interpretable, and explainable because the designer can show the mathematical function that relates known data to computed data.

EXAMPLES

The following manifestations are just a few examples and are discussed with intentions to demonstrate the flexibility of the system architecture as applicable to the problem space of current technologies, e.g., cryptography, information theory, and quantum computation/information. Practitioners of these arts will understand and appreciate their content.

System Identification

As a practical example, consider the process of system identification as applied to the estimation of the rolling moment aerodynamic parameter, $\boldsymbol{C_l}$. System identification is the process of determining an adequate mathematical model, usually containing differential equations, with unknown parameters which are determined indirectly from measured data (Jategaonkar, 2006). System identification is accomplished by a variety of methods: output error, filter error, equation error, recursive parameter estimation, and artificial neural networks. For this example, the proposed system architecture will be compared with the artificial neural net method. To estimate the rolling moment aerodynamic parameter, the artificial neural net method uses 5 independent variables (one of which is aileron deflection, $\boldsymbol{\delta a}$) to simultaneously determine 3 dependent variables (one of which is the rolling moment aerodynamic parameter). After a preliminary exercise in numerical investigation (input/output scaling, initial network weights, number of hidden nodes, learning rate, momentum parameter, and slope factors of the sigmoidal activation functions) convergence is achieved after 2000 iterations, in approximately 30 seconds. The result is a complex, opaque, uninterpretable,

unexplainable relationship between the inputs and outputs. It's important to note, if there are any changes to the inputs or outputs, the model must be retrained.

Figure 6 refers to an architecture for system identification problems using the proposed system architecture. Let the aileron deflection be the known input corresponding to signal 2 of Figure 5. Let the roll moment aerodynamic parameter be the known output corresponding to signal 1 of Figure 5. Assuming the aerodynamic relationship between input and output is unknown, a generic mathematical model is used $\boldsymbol{a_0 + a_1 x + a_2 x^2 + a_e\, exp(nx) + a_l\, ln(x) + a_s\, sin(x) + a_c\, cos(x)}$. The reason for a generic mathematical model is to demonstrate it's possible to describe the relationship between aileron deflection and rolling moment aerodynamic parameter without having *a priori* knowledge regarding aerodynamics. Of course, an aerodynamics practitioner may implement a model describing the mathematics from aileron input to rolling moment aerodynamic parameter output. Minimizing the difference between the computed output and the known output (signal 4 of Figure 5) determines the coefficients of the mathematical functions. The coefficients describe the model and are used to explain the relationship between the input (aileron deflection) and output (roll moment aerodynamic parameter). Rather than using an input/output ratio of 5:3, a 1:1 ratio is used with the proposed system architecture. By using the proposed system architecture (instead of using multiple inputs/multiple outputs in the context of a complicated artificial neural net) much less data is required to determine the relationship between the two data sets (aileron deflection and rolling moment aerodynamic parameter). Also, the results are achieved in 200 iterations – an order of magnitude less than required by the artificial neural net approach – in approximately 20 seconds. Additional resultant data are shown in Table1 and Table 2.

Table 1: Relative Error – Proposed System vs Neural Net

	Proposed System		Neural Net	
Iterations	Time (s)	Relative Error	Time (s)	Relative Error
200	16	10%	4	20%
500	40	3%	8	20%
1000	80	2%	15	20%
2000	160	1%	30	3%

As seen in Table 1, the proposed system takes longer to run, but this is offset by lower relative error. For 2000 artificial neural net iterations, it takes 30 seconds to run with 3% relative error. For the same relative error, the proposed system takes 10 seconds longer to run, but only required 500 iterations. This may appear as trading execution time for iterations. However, what is not included is the time for the practitioner to endure the numerical exercise in determining that, in this very particular case, 2000 iterations of an artificial neural net with 8 hidden layer nodes yield a *decent* solution. In fact, Table 2 demonstrates the varying results when determining the number of iterations and number of hidden layer nodes. The obvious anomaly is seen by examining the nonlinear behavior between node number and relative error within a set of iterations.

Table 2: Relative Error – Neural Net Iterations

	Neural Net		Neural Net	
Nodes	Iterations	Relative Error	Iterations	Relative Error
4	200	57%	2000	20%
8	200	20%	2000	3%
12	200	32%	2000	8%
16	200	441%	2000	2%

Referring to Table 1, another advantage of the proposed system is the ability to get a quick estimate. In this example, the practitioner can run 200 iterations in 16 seconds and obtain half as much relative error (10%) when compared with the artificial neural net approach. Keep in mind, this doesn't include the artificial neural net practitioner's time spent performing numerical investigations.

Furthermore, the artificial neural net approach required the time series data to be in chronological order. The proposed system architecture is agnostic to any timestamp. The only requirement is to maintain the correlation between the two sets of data. The mathematical model is relatively simple, transparent, interpretable, and explainable. Because of these attributes, the proposed system architecture is much more reliable for flight safety certification. The mathematical model can be subsequently exercised to explore extreme cases, e.g., letting variables go to zero and letting variables approach infinity. Hence increasing confidence model deployment.

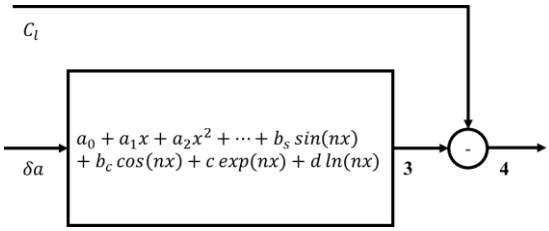

Figure 6: A Generic Mathematical Model for System Identification

To conclude this subsection, it is recognized that classical system identification has a counterpart in quantum process tomography. Therefore, the proposed system may have applications to understanding and controlling noisy quantum systems.

Cryptography

Using the proposed system architecture to emulate cryptography, a sinusoidal signal, composed of a summation of three individual frequency components ($cos\theta, cos2\theta, sin4\theta$) is used as an input to a mathematical model of a discrete Fourier transform. By minimizing the difference between the computed signal (signal 3 of Figure 5) and the "unknown" reference signal (signal 1 of Figure 5), the "unknown" reference signal is decomposed to determine its frequency content (Figure 7).

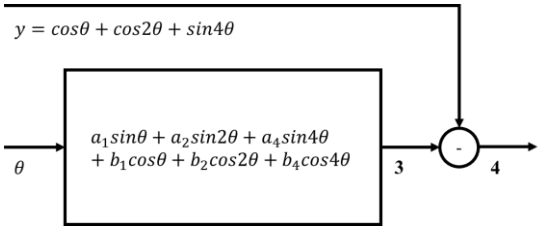

Figure 7: A Mathematical Model for Fourier Transforms

Continuing with another cryptography example, the proposed system architecture is used to perform the task of order finding (Figure 8). Efficient order-finding can be used to break RSA public cryptosystems. In this problem, the integer value of r is sought which satisfies the expression $a^r \equiv 1(mod\ N)$ where $mod\ N$ means modulus N. In this example embodiment, the problem has been formulated as $a^r(modN) - 1$, where the difference has been minimized over different integer values of r. Again, the same architectural approach is applied to a completely different problem type. Regarding Figure 5, the integer 1 corresponds to signal 1 and the $a^r(modN)$ portion corresponds to signal 2. Additional examples may be extended from Fourier transforms and cryptography to their quantum counterparts, i.e., quantum Fourier transforms and quantum cryptography.

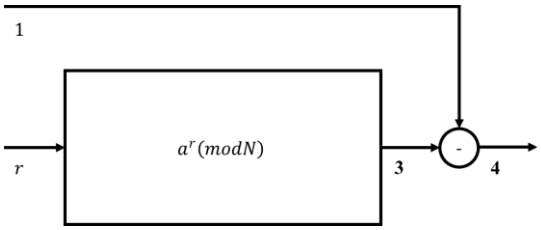

Figure 8: A Mathematical Model for Order Finding

Logic Gates

To demonstrate the flexibility of the proposed system architecture, the next example emphasizes its ability to handle mathematical models containing discontinuities. Consider Figure 9 which is a classical (Boolean) circuit with three inputs A, B, and C. The output of the classical (Boolean) circuit corresponds to signal 3 of Figure 5, while the input (with value 1) corresponds to signal 1 of Figure 5.

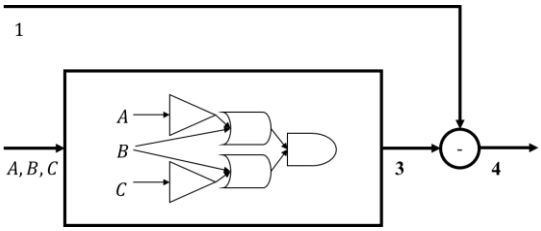

Figure 9: A Boolean Circuit for Classical Logic

Minimizing the difference yields five of the truth table values with entries -1, while maximizing the difference yields the other three values of the truth table with entries 1, see Table 1.

Table 1: Truth Table

A	B	C	DIFF
0	0	0	-1
0	0	1	1
0	1	0	-1
0	1	1	-1
1	0	0	1
1	0	1	1
1	1	0	-1
1	1	1	-1

This is significant because as some mathematical models include an increasing number of logic gates (e.g., decision-making) the complexity of the model architecture may render the problem intractable. Yet, the proposed system architecture allows the practitioner to simply exercise the system to yield the corresponding truth table leading to the discovery of cause-effect relationships. This example of a classical Boolean circuit, within an acyclic directed graph, may be extended to quantum computation/information by implementation of quantum circuits which form the basis for implementing various computations. While physicists and mathematicians view quantum computation as hypothetical experiments, computer scientists view quantum computation as games where players, typically "Alice" and "Bob" optimize their performance in various abstractions. Applications include the minimization of bits for quantum error correction, and GHZ (Greenberger, Horne, and Zeilinger) and CHSH (Clauser, Horne, Shimony, and Holt) games.

Self-Organized Systems

Another example of the proposed system architecture emulates information and self-organized complex systems. The human brain and behavior are shown to exhibit features of pattern-forming, dynamical systems, including multi-stability, abrupt phase changes, crises, and intermittency. How human beings perceive, intend, learn, control, and

coordinate complex behaviors is understood through dynamic systems. Here, a dynamic system is modeled by a power series ($\sum_n a_n x^n$) as a solution to an ordinary differential equation. A second-order harmonic oscillator (mass, spring, damper system) is used to create a set of input-output relations. Using the proposed system architecture, the (spring and damping) coefficients are determined through the power series implementation of the differential equation (Figure 10). Again, this demonstrates the flexibility of this unifying system architecture which is adaptable to a wide range of technological applications.

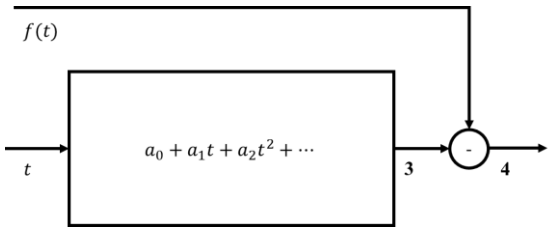

Figure 10: A mathematical model for power series

Real-Time Systems

To conclude this section, consider an example for real-time systems. As practitioners are aware, real-time requirements for aerospace guidance, navigation, and control (GNC) processes are different than real-time requirements for e-commerce transactions. However, in either case, the proposed system architecture may be augmented such that known constraints (if any) could be built into the objective function *a priori*. Also, by selecting an appropriate resolution, the system may be configured to execute in a deterministic time frame. This single approach for multifunctional systems may be used for industrial applications. These multifunctional systems must manage diverse objectives, multiple resources, and numerous constraints. A factory might use several types of power (pneumatic, electrical, hydraulic), several types of labor skills, many different raw materials, all while making multiple products. A production optimization system based on the Industrial Internet of Things (IIoT) can collect data from thousands of sensors. A system with the computational efficiency to support real-time monitoring and control is a valuable advance in optimization techniques.

SUMMARY/CONCLUSION

Practitioners of various arts have complete control over what is being learned. If the practitioner has *a priori* knowledge, then mathematical or logical representations may be included accordingly. The adaptive discovery of the proposed system architecture finds the best configuration of terms contributing to a scientific equation (based on a combination of elementary mathematical functions) which matches real-world observations. Because of mathematical transparency, practitioners can easily interpret the results to see if they correspond with intuition and explain how the system works.

Back-propagation methods are replaced by an adaptive system for solving nonlinear, nonconvex, and discontinuous problems. Paired with a rich set of options for mathematical functions, the unrestricted system architecture can be optimized for a training set of nearly any size. In the case of multiple inputs/outputs, prior knowledge of the hyperspace is not needed. The mathematical architecture is independent of the input/output complexity. Inputs and outputs can be discrete, continuous, deterministic, random, or any combination thereof.

Regarding data, normalization may be performed to avoid domination by any one input. Otherwise, there is no need to manipulate the data. Furthermore, much less data is needed in the case of the system identification architecture compared with the artificial neural net approach. This demonstrates a reduction in the need for massive training sets.

Also, there is no need for enormous computing power. Every manifestation discussed in the examples section runs on a laptop personal computer.

Other benefits include, but aren't limited to:

- Minimized risk associated with data security legislation
- Reduced reliance on large, clean data sets which otherwise limit practical applications

- Reduced footprint for real-time applications dominating networks, servers, and GPUs

It is the author's hope that the proposed system architecture may contribute to training, simulation, and/or education in the context of the American AI Initiative. It is certainly a different and perhaps, unique approach to solving today's technological problems.

REFERENCES

1. Bazaraa, M., et al., (2006), Nonlinear Programming: Theory and Algorithms, Wiley.
2. Chen, R., et al., (2019), Neural Ordinary Differential Equations, arXiv:1806.07366.
3. DARPA, (2018), https://www.darpa.mil/news-events/2018-07-20a.
4. Fouskakis, D., and Draper, D., (2001), Stochastic Optimization: A Review, International Statistical Review.
5. Haken, H., (2010), Information and Self-Organization, Springer.
6. Jain, P. and Kar, P., (2017), Non-Convex Optimization for Machine Learning, arXiv:1712.07897.
7. Jategaonkar, R., (2006), Flight Vehicle System Identification: A Time Domain Methodology, AIAA.
8. Kaplan A., and Haenlein, K., (2018), Siri, Siri in my Hand, who's the Fairest in the Land?, https://doi.org/10.1016/j.bushor.2018.08.004.
9. Kelso, J.A.S., (1995), The Self-Organization of Brain and Behavior, MIT Press.
10. Knight, W., (2017), The U.S. Military Wants Its Autonomous Machines to Explain Themselves, https://www.technologyreview.com/s/603795/the-us-military-wants-its-autonomous-machines-to-explain-themselves.
11. Lamberth, M., (2019), The White House and Defense Department unveiled AI strategies. Now what?, https://www.c4isrnet.com/opinion/2019/02/27/the-white-house-and-defense-department-unveiled-ai-strategies-now-what.
12. Sutton, R.S., and Barto, A.G., (2018), Reinforcement Learning: An Introduction, MIT Press.
13. Taleb, N., (2010), The Black Swan – The Impact of the Highly Impossible, Random House.
14. Zhou, H., et al., (2019), Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask, https://eng.uber.com/deconstructing-lottery-tickets