Neural Arithmetic Logic Units with Two Transition Matrix and Independent Gates

Sthefanie Jofer Gomes Passo
Electrical and Computer Engineering Department
University of Texas at San Antonio
Sthefanie.Passo@utsa.edu

Vishal Kothavade
Electrical and Computer Engineering Department
University of Texas at San Antonio
Vishal.Kothavade@utsa.edu

Wei-Ming Lin
Electrical and Computer Engineering Department
University of Texas at San Antonio
WeiMing.Lin@utsa.edu

***Abstract*—Neural Networks always have been used to manipulate numerical information based on the training, however, their learning behavior does not display systematic generalization. Due to its inability to successfully perform operations when the numerical range used during the test time lie outside during training. To address the systematic generalization, we propose an existed architecture called Neural Arithmetic Logic Units (NALU) but with addition of Independent Gates to it. We would like to call this architecture as Neural Arithmetic Logic Units with Independent Gates (NALUIG) that can represents numerical quantities as linear activations using: primitive arithmetic, operators, controlled by learned gates which will be independent of the input; use of separate weight matrices for the adder and multiplier. Along with that, we have also introduced two more architectures called, Neural Arithmetic Logic Unit with two Transition Matrix (NALU2M), and Neural Arithmetic Logic Unit with two Transition Matrix and Independent Gates (NALU2MIG), by changing the specifications of the original architecture, NALU. Based on our strategy to represent numerical quantities as individual neurons without a non-linearity using independent gates, our experiments show that NALUIG, NALU2M, and NALU2MIG enhanced neural networks can learn to perform arithmetic over images of numbers, and execute computer code with very lower error rates than other existing neural networks. In this paper, we obtain better results in terms of numerical range generalization then the state of art.**

***Index Terms*—Neural Arithmetic Logic Unit, Independent Gates, Transition Matrix, Linear and Non-linear Data.**

I. Introduction

Since the Neural networks does not display systematic generalization even though they can classify and identify numerical quantities given an appropriate learning signal [1] [2], the learned behavior can be better characterized by memorization than by systematic abstraction. In this paper, we successfully re-implemented the architecture design NALU [4] and we further made changes in the specifications to develop new architectures called, NALUIG, NALU2M, and NALU2MIG, by adding independent gates and matrices to the original architecture. These new architectures can be used in traditional neural network architectures that are in the state of art or that have the objective to develop architectures with mathematical methods and Ordinary Differential Equations [3]. Our approach is to represent numerical quantities without a non-linearity; operators will be representing simple functions (e.g., +, -, *, /, etc.). These operators will be controlled by learned gates; independent of the input. That is separate weight matrices for the adder and multiplier and compare with the state of art [4]. By giving different input the operations will calculate the output and this input and outputs will be given to the network. However, despite this combinational character, they will be differentiable, making it possible to learn them with backpropagation [5].

We have experimented and compared all architectures mentioned above with respect to the inter-polarization and extra-polarization error rates for static tasks. The inter-polarization is an evaluation that looks at performance of the model on held out values from within the training range while the extra-polarization values from outside of the training range. We find that our proposed models perform better and can learn functions over representations that capture the underlying numerical nature of the data and generalize to numbers that are several orders of magnitude larger than those observed during training.

II. Related Work

One related work is where authors [6], presents a general-purpose framework (PyPEF: pythonic protein engineering framework) for performing data-driven protein engineering using machine learning methods combined with techniques from signal processing and statistical physics as the prediction and effect on functional properties remain one of the most significant challenges in protein engineering. They show that the program could efficiently predict the fitness of protein sequences for different target properties (predictive models with coefficient of determination values ranging from 0.58 to 0.92). By combining machine learning and protein evolution, PyPEF enabled the screening of proteins with various functions, reaching a screening capacity of more than 500,000 protein sequence variants in the timeframe of only a few minutes on a personal computer. In essence, PyPEF can provide a powerful solution to current sequence exploration and combinatorial problems faced in protein engineering through exhaustive in silico screening of the sequence space.

Also, neural networks have been used to build SCAN domain [7], consisting of a set of simple compositional navigation commands paired with the corresponding action sequences. The authors constructed a SCAN with recurrent neural network (RNN) and found that RNNs can make successful zero-shot generalizations when the differences between training and testing commands are small, so that they can apply "mix-and-match" strategies to solve the task. They conclude that with a proof-of-concept experiment in neural machine translation, suggesting that lack of systematicity might be partially responsible for neural networks notorious training data thirst.

On the other hand, a recent work of weight initialization in neural networks rather than optimization methods is found [8]. Neural networks rely on lucky random initial weights of subnetworks called "lottery tickets" that converge quickly to a solution. To investigate how weight initializations affects performance, the authors, examine small convolutional networks that are trained to predict steps of the two-dimensional cellular automation Conway's Game of Life, the update rules of which can be implemented efficiently in a small CNN. They found that gradient descent converges to a solution are sensitive to small perturbations, such as a single sign change. Their results are consistent with the lottery ticket hypothesis.

One of the research works is advocating our research where, a deep learning framework for a spatiotemporal attention network (STANet) with a neural algorithm logic unit (NALU), the so-called STANet-NALU, to understand the dynamic aggregation effect of private cars on weekends [9]. Because, intelligent transportation management and urban planning are challenging on weekends, due to the inefficient representations of spatiotemporal features for such aggregation effect and the considerable randomness of private car mobility. The authors utilize the stay-time of private cars as a temporal feature to represent the nonlinear temporal correlation of the aggregation effect. They conduct extensive experiments based on real-world private car trajectories data.

Symbolic regression, which is a powerful technique to discover analytic equations that describe data, can lead to explainable models and the ability to predict unseen data [10]. In this paper, the authors are seeking the use a neural network-based architecture for symbolic regression called the equation learner (EQL) network and integrate it with other deep learning architectures such that the whole system can be trained end-to-end through backpropagation. To demonstrate the power of such systems, they have studied their performance on several substantially different tasks. They have shown that the neural network can perform symbolic regression and learn the form of several functions and with an MNIST arithmetic task where a convolutional network extracts the digits. This related work also demonstrates the prediction of dynamical systems where an unknown parameter is extracted through an encoder.

To utilize the raw inputs and symbolic knowledge simultaneously, some recent neuro-symbolic learning methods use abduction, i.e., abductive reasoning, to integrate sub-symbolic perception and logical inference. While the perception model, e.g., a neural network, outputs some facts that are inconsistent with the symbolic background knowledge base, abduction can help revise the incorrect perceived facts by minimizing the inconsistency between them and the background knowledge. However, to enable effective abduction, previous approaches need an initialized perception model that discriminates the input raw instances. This limits the application of these methods, as the discrimination ability is usually acquired from a thorough pre-training when the raw inputs are difficult to classify. One can view that our work as advocating to propose a novel abduction strategy, which leverages the similarity between samples, rather than the output information by the perceptual neural network, to guide the search in abduction [11].

Over the years, combinatorial optimization is a well-established area in operational research. Until recently, its methods have focused on solving problem instances in isolation. In recent tends of graph neural networks (GNNs), as a key building block for combinatorial tasks, either directly as solvers or by enhancing exact solvers. The inductive bias of GNNs effectively encodes combinatorial and relational input due to their invariance to permutations and awareness of input sparsity [12]. This research work also collaborates with the research we have been working on.

One of the theoretical analyses is exclusively related to research topic we have been working on. The authors [13], presents how feedforward neural networks, a.k.a. multilayer perceptrons (MLPs), do not extrapolate well in certain simple tasks, while Graph Neural Networks (GNNs) – structured networks with MLP modules – have shown some success in more complex tasks.

In the other related works, there is a OccamNet [14], a neural network model that finds interpretable, compact, and sparse solutions for fitting data. This model defines a probability distribution over a non-differentiable function space by introducing a two-step optimization method that samples functions and updates the weights with backpropagation based on cross-entropy matching in an evolutionary strategy. OccamNet is able to fit a variety of symbolic laws including simple analytic functions, recursive programs, implicit functions, simple image classification, and can outperform noticeably state-of-the-art symbolic regression methods on real world regression datasets. This model requires minimal memory footprint, does not require AI accelerators for efficient training, fits complicated functions in minutes of training on a single CPU, and demonstrates significant performance gains when scaled on a GPU.

Collectively, numerical reasoning is central to many problems in intelligence and by extension is an important topic in deep learning. An arithmetic logic unit using neural networks (NALU) (with linear activations which are manipulated using primitive arithmetic operators and controlled by learned gates) would improve results looking for the objectives and overcome the existing limitations of artificial intelligence.

III. Methodology

In this research work, we have re-implemented the original architectures Neural Accumulator (NAC) and NALU, and then proposed and implemented modified architectures called, NALUIG, NALU2M, and NALU2MIG.

Basically, NAC supports the ability to accumulate quantities actively for linear extrapolation which it's arithmetic functions can be effectively incorporated into an end-to-end model. The NAC is a special case of a linear layer whose transformation matrix W have the values -1, 0 or 1; the outputs are additions or subtractions would work of rows in the input vector. The state of art [4] improved the inductive bias in each layer by encouraging 0's, 1's and -1's within W and we maintained this configuration.

For the original architecture, the authors [4] have used only one transformation matrix W for the implementation of NAC, whereas, in our research, we have further expanded the use of transformation matrices to two, $W0$ and $W1$. Each transformation matrix has a differential pasteurization of W in terms of unconstrained parameters:

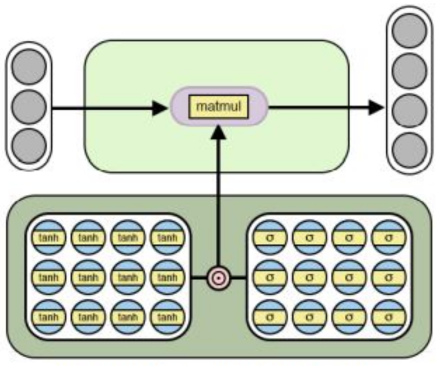
$$W = W0 = W1 = tanh(W) \odot \sigma(M) \quad (1)$$

NAC is designed for addition and subtraction but it would be desirable to have a model that have a systematic generalization for other mathematical functions like multiplication. Therefore, the authors [4] implemented, NALU, as shown in Figure 1, which learns a weighted sum between two sub cells, one capable of addition and subtraction and the other capable of multiplication, division, and power functions such as square roots. We can use one transformation matrix W for calculation or two separate weight matrices for the adder $(W0)$ and the multiplier $(W1)$. Experiments have been made with both combinations and the results were evaluated with the error rate metric.

The NALU consists of two NAC cells (the purple cells) interpolated by a learned sigmoidal gate g (the orange cell), such that if the add/subtract subcell's output value is applied with a weight of 1 (on), the multiply/divide subcell's is 0 (off) and vice versa. The first NAC (the smaller purple subcell) computes the accumulation vector a, which stores results of the NALU's addition/subtraction operations; it is computed identically to the original NAC, (i.e., a = Wx). The second NAC (the larger purple subcell) operates in log space and is therefore capable of learning to multiply and divide, storing its results in m:

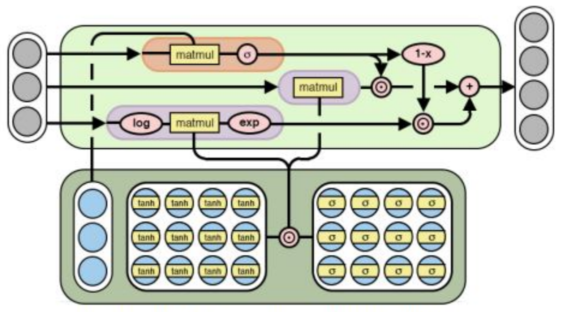
NAC:

$$a = Wx \quad (2)$$



(a) Neural Accumulator (NAC)

Fig. 1. The Neural Accumulator (NAC).



(b) Neural Arithmetic Logic Unit (NALU)

Fig. 2. The Neural Arithmetic Logic Unit (NALU) uses two NACs with tied weights to enable addition/subtraction (smaller purple cell) and multiplication/division (larger purple cell), controlled by a gate (orange cell).

NALU:

$$a = Wx \quad (3)$$

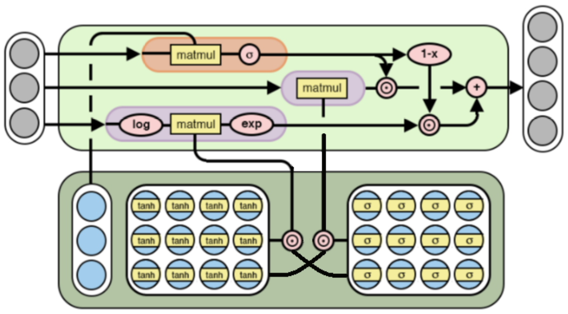
$$m = expW(log(|x| + \epsilon)) \quad (4)$$

$$g = \sigma(Gx) \quad (5)$$

$$y = g \odot a + (1 - g) \odot m \quad (6)$$

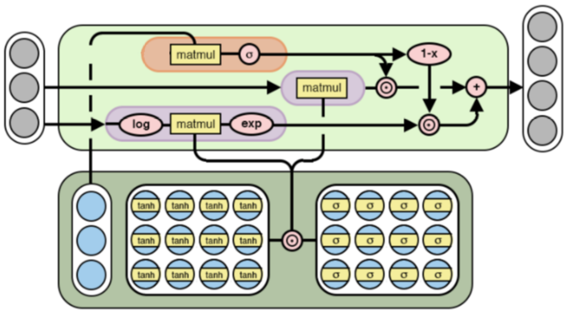
Based on the architecture of NALU, we have implemented a Neural Arithmetic Logic Unit with two Transition Matrix (NALU2M), a Neural Arithmetic Logic Unit with Independent Gates (NALUIG) and a Neural Arithmetic Logic Unit with two Transition

Matrix and Independent Gates (NALU2MIG). Each architecture can be visualized in the Figure 3, Figure 4 and Figure 5 respectively.



(c) Neural Arithmetic Logic Unit with two Transition Matrix (NALU2M)

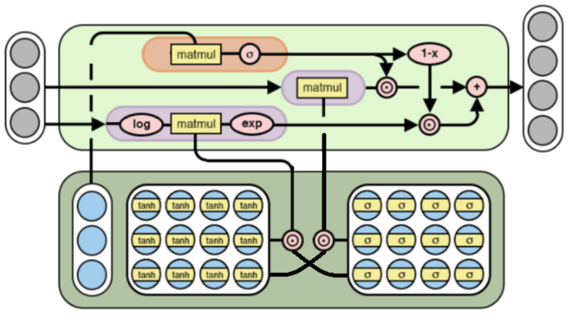
Fig. 3. Neural Arithmetic Logic Unit with two Transition Matrix (NALU2M).



(d) Neural Arithmetic Logic Unit with Independet Gates (NALUIG)

Fig. 4. Neural Arithmetic Logic Unit with Independent Gates (NALUIG).

For the implementation of these three modified architectures, we used, two different transition matrix $W0$ and $W1$ (calculated like the equation 1). Our idea is that, when we do the same calculation using different variables the software will understand better than we use one variable to do multiple things in machine learning. This can variate depending on the paradigm of the language one is using, here we have used Python and TensorFlow for the



(e) Neural Arithmetic Logic Unit with two Transition Matrix and Independet Gates (NALU2MIG)

Fig. 5. The Neural Arithmetic Logic Unit with two Transition Matrix and Independent Gates (NALU2MIG).

experiments. For each model NALU2M, NALUIG and NALU2MIG, we have used the following equations to the adder a, the multiplier m and image y:

NALU2M:

$$a = W0x \tag{7}$$

$$m = expW1(log(|x| + \epsilon)) \tag{8}$$

$$g = \sigma(Gx) \tag{9}$$

$$y = g \odot a + (1 - g) \odot m \tag{10}$$

NALUIG:

$$a = Wx \tag{11}$$

$$m = expW(log(|x| + \epsilon))) \tag{12}$$

$$g = \sigma(G) \tag{13}$$

$$y = g \odot a + (1 - g) \odot m \tag{14}$$

NALU2MIG:

$$a = W0x \tag{15}$$

$$m = expW1(log(|x| + \epsilon)) \tag{16}$$

$$g = \sigma(Gx) \tag{17}$$

$$y = g \odot a + (1 - g) \odot m \tag{18}$$

During the experiments, we have compared the models which are looking for the same metrices and have calculated the inter-polarization and extra-polarization error rates for static tasks. The inter-polarization is an evaluation that looks at performance of the model on held-out values from within the training range, while the extra-polarization values from outside of the training range. On the static task, for baseline models we compare the NALU2M, NALUIG and NALU2MIG to NAC, NALU and MLPs with a variety of standard non-linearities as well as a linear model.

IV. Experiments and Results

We test numeric reasoning, inter-polarization and extra-polarization in a variety of settings; looking at different activation functions such as ReLU, NAC, NALU, NALU2M, NALUIG and NALU2MIG. The experiments can be found at https://github.com/SthePasso/NALU2IM. In the original architecture, the authors have studied the explicit learning of simple arithmetic functions directly from numerical input, and indirectly from image data, whereas, in our models, we experimented such that numerical inputs will not be connected directly with the output gate, so called our approach, Neural Arithmetic Logic Unit with Independent Gates. We have considered temporal domains: the translation of text to integer values, and the evaluation of computer programs containing conditional logic and arithmetic.

A. *Interpolarization and Extrapolarization*

The summary of the results is shown below. It shows that while several standard architectures succeed at these tasks in the interpolation case, none of them succeed at extrapolation. However, in both interpolation and extrapolation, the NALU2M, NALUIG and NALU2MIG succeeds at modeling addition and subtraction, and multiplicative as well as division operations in the static tasks.

Tab. 1. Interpolation Test Errors

Interpolation Test Errors							
Fun	ReLU	Sigmoid	NAC	NALU	NALU2M	NALUIG	NALU2MIG
a + b	**0.00**	0.12	**0.00**	0.33	0.10	**0.00**	0.02
a - b	0.27	0.37	**0.00**	9.88	**0.00**	90.44	0.13
a * b	1.84	1.20	13.71	0.91	0.36	6.87	**0.00**
a / b	0.67	0.14	1.75	0.21	0.06	0.05	**0.01**
a ^{2}	4.02	0.43	21.79	1.31	0.17	7.42	**0.00**
$\sqrt{a}$	0.80	0.07	2.95	0.05	0.04	**0.01**	**0.01**

Tab. 2. Extrapolation Test Errors

Extrapolation Test Errors							
Fun	ReLU	Sigmoid	NAC	NALU	NALU2M	NALUIG	NALU2MIG
a + b	**0.00**	62.47	**0.00**	85.86	44.93	0.02	0.44
a - b	55.18	55.74	**0.00**	80.15	**0.00**	90.54	0.39
a * b	57.51	88.54	76.09	99.89	98.28	85.50	**0.00**
a / b	2.73	1.15	20.77	3.35	1.03	16.82	**0.11**
a ^{2}	58.25	82.84	77.55	98.61	76.69	88.29	**0.00**
$\sqrt{a}$	18.32	19.57	65.64	45.42	21.23	4.60	**0.19**

Scores are scaled relative to a randomly initialized model for each task such that 100.0 is equivalent to random, 0.0 is perfect error rates , and > 100 is worse than a randomly initialized model.

V. Conclusion

Our experiments show that NALU-enhanced neural networks i.e., NALU2M, NALUIG and NALU2MIG, can be useful in overcoming the issues faced by traditional neural networks. The NALU2MIG have a error rate lower then the state of art witch make it clear that it do a accurate generalization, numerical representations and numerical functions outside the range observed during training. The experiments show that, as compared to other traditional neural network activation functions, our models allow arbitrary numerical functions to be added to the module and controlled via independent learned gates and matrices.

References

[1] Jerry A. Fodor and Zenon W. Pylyshyn, "Connectionism and cognitive architecture: a critical analysis", *Cognition, 28(1–2):3–71*, 1988.

[2] Gary F. Marcus, "The Algebraic Mind: Integrating Connectionism and Cognitive Science", *MITPress*, 2003.

[3] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, "Neural Ordinary Differential Equations", *University of Toronto, Vector Institute*, 2019.

[4] Andrew T., Felix H., Scott R., Jack R., Chris D., Phil B., "Neural Arithmetic Logic Units", *Advances in Neural Information Processing Systems (NeurlPS)*, 2018.

[5] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors", *Nature, 323(6088):533*, 1986.

[6] N. Siedhoff, A. Illig, U. Schwaneberg, M. Davari, "PyPEFAn Integrated Framework for Data-Driven Protein Engineering", *Journal of Chemical Information and Modeling*, July 2021.

[7] Lake B., Baroni, M., "Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks", *International Conference of machine Learning (ICML)*, 2018.

[8] J. M. Springer, G. T. Kenyon, "It's Hard for Neural Networks to Learn the Game of Life", *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1-8, doi: 10.1109/IJCNN52387.2021.9534060.

[9] Z. Xiao, H. Fang, H. Jiang, J. Bai, V. Havyarimana, H. Chen, L. Jiao, "Understanding Private Car Aggregation Effect via Spatio-Temporal Analysis of Trajectory Data," in *IEEE Transactions on Cybernetics*, doi: 10.1109/TCYB.2021.3117705.

[10] S. Kim, P. Y. Lu, S. Mukherjee, M. Gilbert, L. Jing, V. Ceperic, "Integration of Neural Network-Based Symbolic Regression in Deep Learning for Scientific Discovery," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 9, pp. 4166-4177, Sept. 2021, doi: 10.1109/TNNLS.2020.3017010.

[11] Y. Huang, W. Dai, L. Cai, S. Muggleton, Y. Jiang, "Fast Abductive Learning by Similarity-based Consistency Optimization", *Advances in Neural Information Processing Systems (NeurlPS)*, 2021.

[12] Q. Cappart, D. Chetelat, E. B. Khalil, A. Lodi, C. Morris, P. Velickovic, "Combinatorial Optimization and Reasoning with Graph Neural Networks", *Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, 2021.

[13] K. Xu, M. Zhang, J. Li, S. S. Du, K. Kawarabayashi, S. Jegelka, "How Neural Networks Extrapolate: From Feed-Forward to Graph Neural Networks", *The International Conference on Learning Representations (ICLR-21)*, 2021.

[14] A. Costa, R. Dangovski, O. Dugan, S. Kim, P. Goyal, M. Soljacic, J. Jacobson, "Fast Neural Models for Symbolic Regression at Scale", *Advances in Neural Information Processing Systems (NeurlPS)*, 2021.