Neural Power Unit
– A Bayesian Approach To Neural Arithmetic

Niklas Heim, Václav Šmídl, Tomáš Pevný
Artificial Intelligence Center
Czech Technical University
Prague, CZ 120 00
{niklas.heim, vasek.smidl, tomas.pevny}@aic.fel.cvut.cz

Abstract

Common Neural Networks can approximate simple arithmetic operations, but fail to generalize well beyond the training manifold. A new class of units called *Neural Arithmetic Units* aim to overcome this difficulty, but are limited either to operate on positive numbers (NALU by Trask et al. [2018]), or can only represent simple addition and multiplication (NAU & NMU by Madsen and Johansen [2020]). We present the first Neural Arithmetic Unit that operates on the full domain of real numbers and is capable of learning arbitrary power functions.

1 Introduction

Numbers and simple algebra are essential not only to our human society but also to the survival of many other species [Dehaene, 2011, Gallistel, 2018]. This can be taken as a hint that arithmetic is an important ingredient to a successful, intelligent agent. State of the art neural networks are capable of learning simple arithmetic, but they fail to extrapolate beyond the ranges seen during training [Suzgun et al., 2018, Lake and Baroni, 2018]. The failure of numerical extrapolation on simple arithmetic tasks has been shown by Trask et al. [2018], who also introduce a new class of *Neural Arithmetic Units* that show exceptional extrapolation performance on most arithmetic tasks. ...now explain NALU...

The inability to generalize to unseen inputs is a fundamental problem that hints at a lack of *understanding* of the given task. The model merely memorizes the seen inputs and fails to abstract the true learning task. This can be demonstrated by training a traditional, dense NN to approximate three functions

$$f(x) = e^x, \qquad g(x) = \log(x), \qquad h(x) = \sin(x) \tag{1}$$

Neural Arithmetic Units aim to overcome this problem with units that are able to represent simple arithmetic operations exactly. By composition of these units it becomes possible to learn e.g. exponentials, logarithms, and power functions. This promises to improve on the extrapolation capabilities of NNs even on tasks that are beyond artificial arithmetic tasks such as learning multiplication, or a certain power function .

cite some examples

mention ODEs as use case

mention explainability

2 Related Work

Trask et al. [2018] have demonstrated how severe the extrapolation problem is even for the simplest arithmetic operations, such as summing or multiplying two numbers. In order to increase the power of abstraction of NNs they propose a *Neural Arithmetic Logic Unit* (NALU) which is capable of

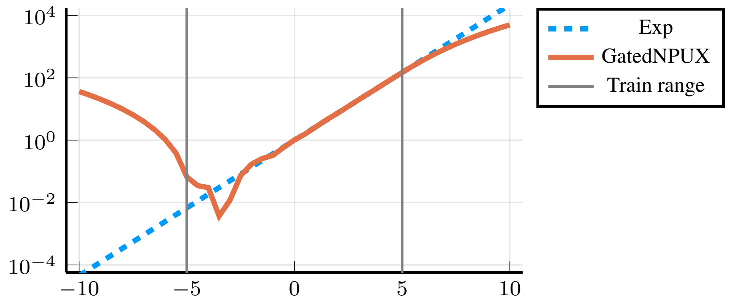

Figure 1:

Table 1: Comparison of different 2×2 layers on the task $(x, y) \to (xy, \frac{x}{y})$. $(x, y) \in \mathcal{U}^2(-2, 2)$.

Model	MultTrain	DivTrain	MultTest	DivTest
NPU	6.5401982e-6	0.029891482	0.00010193295,	6.5048704
NMU	0.0	83.64025	0.0,	20139.203
NALU	1.0725704	83.89075	23.036715,	20136.848

learning arithmetic addition, subtraction, multiplication, and division $\{+, -, \times, \div\}$ with stunning extrapolation accuracy. However, the NALU comes with the severe limitation not being able to handle negative inputs due to the logarithm in the multiplication part of the NALU:

$$\text{Addition: } \boldsymbol{a} = \boldsymbol{W}\boldsymbol{x} \qquad \boldsymbol{W} = \tanh(\hat{\boldsymbol{W}}) \odot \sigma(\hat{\boldsymbol{M}}) \tag{2}$$

$$\text{Multiplication: } \boldsymbol{m} = \exp \boldsymbol{W}(\log(|\boldsymbol{x}| + \epsilon)) \tag{3}$$

$$\text{Output: } \boldsymbol{y} = \boldsymbol{a} \odot \boldsymbol{g} + \boldsymbol{m} \odot (1 - \boldsymbol{g}) \qquad \boldsymbol{g} = \sigma(\boldsymbol{G}\boldsymbol{x}) \tag{4}$$

A multiplication layer that can handle negative inputs was introduced by Madsen and Johansen [2020]. The *Neural Multiplication Unit* (NMU) is defined by Eq. 5 and is typically used in conjunction with the so-called *Neural Addition Unit* (NAU) in Eq. 6.

$$\text{NMU: } y_j = \prod_i M_{ij} z_i + 1 - M_{ij} \qquad M_{ij} = \min(\max(M_{ij}, 0), 1) \tag{5}$$

$$\text{NAU: } \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \qquad A_{ij} = \min(\max(A_{ij}, 0), 1) \tag{6}$$

In both NMU and NAU the weights are clipped to $[0, 1]$, and typically regularized with $\mathcal{R}$:

$$\mathcal{R} = \sum_{ij} \min(W_{ij}, 1 - W_{ij}) \tag{7}$$

The combination of NAU and NMU can thus learn $\{+, -, \times\}$, but no division.

put this in acknowledgements all the results in this paper were create with the help of the following Julia packages: Rackauckas and Nie [2017] + (Flux.jl)

add sqrt and positive range to table 1?

3 Neural Power Unit

Our *Neural Power Unit* (NPU) can learn arbitrary power functions (which includes division) while still being able to correctly deal with negative inputs. The NPU layer is defined by

$$k_i = \begin{cases} 0 & x_i \leq 0 \\ \pi & x_i > 0 \end{cases} \tag{8}$$

$$\boldsymbol{r} = |\boldsymbol{x}| \tag{9}$$

$$\boldsymbol{m} = \exp(\boldsymbol{M} \log(\boldsymbol{r})) \odot \cos(\boldsymbol{M}\boldsymbol{k}) \tag{10}$$

where $\boldsymbol{k}$ is a vector that is zero where $\boldsymbol{x}$ is positive and π where it is negative, $\boldsymbol{r}$ is the elementwise absolute value, and $\boldsymbol{M}$ the multiplication matrix that we want to learn.

The NPU is inspired by the multiplication part of the NALU Eq. 3, extended to negative inputs by taking advantage of the complex logarithm. In general, for any complex number z[1]

$$\log(z) = \log\left(r \cdot e^{i\varphi}\right) = \log(r) + i\varphi. \tag{11}$$

However, as z is assumed to be real, we can simplify as follows:

$$\log(z) = \log(r) + ik\pi, \tag{12}$$

where $k \in [0, 1]$ is zero for positive inputs and one for negative inputs z.

$$\boldsymbol{m} = \exp(\boldsymbol{M}\log(\boldsymbol{x})) \tag{13}$$

$$= \exp(\boldsymbol{M}(\log(\boldsymbol{r}) + i\pi\boldsymbol{k})) \tag{14}$$

$$\boldsymbol{m}_{\text{re}} = \text{real}(\exp(\boldsymbol{M}\log\boldsymbol{r} + i\pi\boldsymbol{M}\boldsymbol{k})) \tag{15}$$

$$= \text{real}(\exp(\boldsymbol{M}\log\boldsymbol{r}) \odot (\cos(\pi\boldsymbol{M}\boldsymbol{k}) + i\sin(\pi\boldsymbol{M}\boldsymbol{k}))) \tag{16}$$

$$= \exp(\boldsymbol{M}\log\boldsymbol{r}) \odot \cos(\pi\boldsymbol{M}\boldsymbol{k}) \tag{17}$$

4 Experimetns

4.1 10 param func

4.2 Gradient surfaces

References

Stanislas Dehaene. *The Number Sense: How the Mind Creates Mathematics, Revised and Updated Edition*. Oxford University Press, April 2011. ISBN 978-0-19-991039-7. Google-Books-ID: 1p6XWYuwpjUC.

C. R. Gallistel. Finding numbers in the brain. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 373(1740):20170119, February 2018. ISSN 0962-8436, 1471-2970. doi: 10.1098/rstb.2017.0119. URL `https://royalsocietypublishing.org/doi/10.1098/rstb.2017.0119`.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv:1711.00350 [cs]*, June 2018. URL `http://arxiv.org/abs/1711.00350`. arXiv: 1711.00350.

Andreas Madsen and Alexander Rosenberg Johansen. Neural Arithmetic Units. page 31, 2020.

Christopher Rackauckas and Qing Nie. DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. *Journal of Open Research Software*, 5:15, May 2017. ISSN 2049-9647. doi: 10.5334/jors.151. URL `http://openresearchsoftware.metajnl.com/articles/10.5334/jors.151/`.

Mirac Suzgun, Yonatan Belinkov, and Stuart M Shieber. On Evaluating the Generalization of LSTM Models in Formal Languages. page 10, 2018.

Andrew Trask, Felix Hill, Scott Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural Arithmetic Logic Units. *arXiv:1808.00508 [cs]*, August 2018. URL `http://arxiv.org/abs/1808.00508`. arXiv: 1808.00508.

[1]Note that complex numbers can be represented in the polar, complex plane where $z = re^{i\varphi}$ with $r > 0$ and $\varphi \in (0, 2\pi)$.