NEURAL ARITHMETIC UNITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks can approximate complex functions, but they struggle to perform exact arithmetic operations over real numbers. The lack of inductive bias for arithmetic operations leaves neural networks without the underlying logic needed to extrapolate on tasks such as addition, subtraction, and multiplication. We present two new neural network components: the Neural Addition Unit (NAU), which can learn to add and subtract; and Neural Multiplication Unit (NMU) that can multiply subsets of a vector. The NMU is to our knowledge the first arithmetic neural network component that can learn multiplication of a vector with a large hidden size. The two new components draw inspiration from a theoretical analysis of recent arithmetic components. We find that careful initialization, restricting parameter space, and regularizing for sparsity is important when optimizing the NAU and NMU. Our results, compared with previous attempts, show that the NAU and NMU converges more consistently, have fewer parameters, learn faster, do not diverge with large hidden sizes, obtain sparse and meaningful weights, and can extrapolate to negative and small numbers.

1 INTRODUCTION

When studying intelligence, insects, reptiles, and humans have been found to possess neurons with the capacity to hold integers, real numbers, and perform arithmetic operations (Nieder, 2016; Rugani et al., 2009; Gallistel, 2018). In our quest to mimic intelligence we have put much faith in neural networks, which in turn has provided unparalleled and often superhuman performance in tasks requiring high cognitive abilities (Silver et al., 2016; Devlin et al., 2018; OpenAI et al., 2018). However, when using neural networks to learn simple arithmetic problems, such as counting, multiplication, or comparison they systematically fail to extrapolate onto unseen ranges (Lake & Baroni, 2018; Suzgun et al., 2019; Trask et al., 2018). The absence of inductive bias makes it difficult for neural networks to extrapolate well on arithmetic tasks as they lack the underlying logic to represent the required operations.

We would like to achieve a neural network component that can take an arbitrary hidden input, learn to select the appropriate elements, and apply the desired arithmetic operation. A recent attempt to achieve this goal is the Neural Arithmetic Logic Unit (NALU), by Trask et al. (2018).

The NALU consists of two sub-units: the NAC_+ for addition/subtraction and the $\text{NAC}_\bullet$ for multiplication/division. The sub-units are softly gated using a sigmoid function in order to exclusively select one of the sub-units. However, we find that the soft gating mechanism and the $\text{NAC}_\bullet$ are fragile and hard to learn.

In this paper, we analyze and improve upon the NAC_+ and $\text{NAC}_\bullet$ with respect to addition, subtraction, and multiplication. Our proposed improvements, namely the Neural Addition Unit (NAU) and Neural Multiplication Unit (NMU), are more theoretically founded and improve performance regarding stability, speed of convergence, and interpretability of results. Most importantly, the NMU can support a large hidden input-size.

The improvements, based on a theoretical analysis of the NALU and its components, are achieved by a simplification of the parameter matrix for a better gradient signal, a sparsity regularizer, and a new multiplication unit that can be optimally initialized and supports both negative and small numbers. The NMU do not support division. However, we find that the $\text{NAC}_\bullet$ in practice also only supports multiplication and cannot learn division (for theoretical findings on why division is hard to learn, see section 2.3).

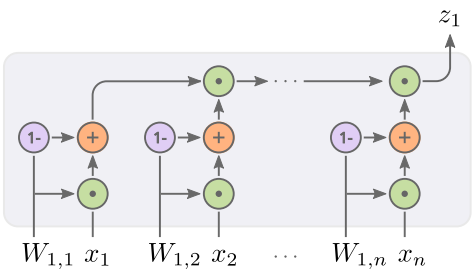


Figure 1: Visualization of NMU for a single output scalar z_1, this construction repeats for every element in the output vector $\mathbf{z}$.

To analyze the impact of each improvement we introduce several variants of the NAC$_\bullet$. We find that, allowing division makes optimization for multiplication harder, linear and regularized weights improve convergence, and that the NMU style of multiplication is critical when increasing the hidden input size.

Furthermore, we improve upon existing benchmarks in Trask et al. (2018) by expanding the "simple function task", using a multiplicative variant of "MNIST Counting and Arithmetic Tasks", and use an improved success-criterion Anonomous (2019). A success-criterion is important because the arithmetic layers are solving a logical problem. We propose the MNIST multiplication variant as we want to test the NMU's and NAC$_\bullet$'s ability to learn from real data.

1.1 Learning a 10 parameter function

Consider the static function $t = (x_1 + x_2) \cdot (x_1 + x_2 + x_3 + x_4)$ for $x \in \mathbb{R}^4$. To illustrate the ability of NAC$_\bullet$, NALU, and our proposed NMU, we conduct 100 experiments for each model, where we attempt to fit this function. Table 1 shows that NMU has a higher success rate and converge faster.

Table 1: Comparison of the success-rate, when the model converged, and the sparsity error for all weight matrices, with 95% confidence interval on the $t = (x_1 + x_2) \cdot (x_1 + x_2 + x_3 + x_4)$ task. Each value is a summary of 100 different seeds.

Op	Model	Success	Solved at		Sparsity error
		Rate	Median	Mean	Mean
$\times$	NAC$_\bullet$	$13\% \; {}^{+8\%}_{-5\%}$	$5.5 \cdot 10^4$	$5.9 \cdot 10^4 \; {}^{+7.8 \cdot 10^3}_{-6.6 \cdot 10^3}$	$7.5 \cdot 10^{-6} \; {}^{+2.0 \cdot 10^{-6}}_{-2.0 \cdot 10^{-6}}$
	NALU	$26\% \; {}^{+9\%}_{-8\%}$	$7.0 \cdot 10^4$	$7.8 \cdot 10^4 \; {}^{+6.2 \cdot 10^3}_{-8.6 \cdot 10^3}$	$9.2 \cdot 10^{-6} \; {}^{+1.7 \cdot 10^{-6}}_{-1.7 \cdot 10^{-6}}$
	NMU	$\mathbf{94\%} \; {}^{+3\%}_{-6\%}$	$\mathbf{1.4 \cdot 10^4}$	$\mathbf{1.4 \cdot 10^4} \; {}^{+2.2 \cdot 10^2}_{-2.1 \cdot 10^2}$	$\mathbf{2.6 \cdot 10^{-8}} \; {}^{+6.4 \cdot 10^{-9}}_{-6.4 \cdot 10^{-9}}$

2 Introducing differentiable binary arithmetic operations

We define our problem as learning a set of static arithmetic operations between selected elements of a vector. E.g. for a vector $\mathbf{x}$ learn the function $(x_5 + x_1) \cdot x_7$. The approach taking in this paper, is to develop layers around specific operations, and then let each layer decide which inputs to include using backpropagation.

We develop these layers by taking inspiration from a theoretical analysis of Neural Arithmetic Logic Unit (NALU) by Trask et al. (2018).

2.1 Introducing NALU

The Neural Arithmetic Logic Unit (NALU) consists of two sub-units; the NAC$_+$ and NAC$_\bullet$. The sub-units represent either the $\{+, -\}$ or the $\{\times, \div\}$ operations. The NALU then assumes that either NAC$_+$ or NAC$_\bullet$ will be selected exclusively, using a sigmoid gating-mechanism.

The NAC_+ and $\text{NAC}_\bullet$ are defined accordingly,

$$W_{h_\ell,h_{\ell-1}} = \tanh(\hat{W}_{h_\ell,h_{\ell-1}})\sigma(\hat{M}_{h_\ell,h_{\ell-1}}) \tag{1}$$

$$\text{NAC}_+ : \; z_{h_\ell} = \sum_{h_{\ell-1}=1}^{H_{\ell-1}} W_{h_\ell,h_{\ell-1}} z_{h_{\ell-1}} \tag{2}$$

$$\text{NAC}_\bullet : \; z_{h_\ell} = \exp\left(\sum_{h_{\ell-1}=1}^{H_{\ell-1}} W_{h_\ell,h_{\ell-1}} \log(|z_{h_{\ell-1}}| + \epsilon)\right) \tag{3}$$

where $\hat{\mathbf{W}}, \hat{\mathbf{M}} \in \mathbb{R}^{H_\ell \times H_{\ell-1}}$ are weight matrices and $z_{h_{\ell-1}}$ is the input. The matrices are combined using a tanh-sigmoid transformation to bias the parameters towards a $\{-1, 0, 1\}$ solution. Having $\{-1, 0, 1\}$ allows NAC_+ to perform exact $\{+, -\}$ operations between elements of a vector. The $\text{NAC}_\bullet$ uses an exponential-log transformation to create the $\{\times, \div\}$ operations within ϵ precision and for positive inputs only.

The NALU combines these units with a gating mechanism $\mathbf{z} = \mathbf{g} \odot \text{NAC}_+ + (1 - \mathbf{g}) \odot \text{NAC}_\bullet$ given $\mathbf{g} = \sigma(\mathbf{G}\mathbf{x})$. Thus allowing NALU to decide between all of $\{+, -, \times, \div\}$ using backpropagation.

2.2 Weight matrix construction and the Neural Addition Unit

Glorot & Bengio (2010) show that $E[z_{h_\ell}] = 0$ at initialization is a desired property, as it prevents explosion of both the output and the gradients. To satisfy this property with $W_{h_{\ell-1},h_\ell} = \tanh(\hat{W}_{h_{\ell-1},h_\ell})\sigma(\hat{M}_{h_{\ell-1},h_\ell})$, an initialization must satisfy $E[\tanh(\hat{W}_{h_{\ell-1},h_\ell})] = 0$. In NALU, this initialization is unbiased as it samples evenly between $+$ and $-$, or $\times$ and $\div$. Unfortunately, this initialization also causes the expectation of the gradient to become zero, as shown in (4).

$$E\left[\frac{\partial \mathcal{L}}{\partial \hat{M}_{h_{\ell-1},h_\ell}}\right] = E\left[\frac{\partial \mathcal{L}}{\partial W_{h_{\ell-1},h_\ell}}\right] E\left[\tanh(\hat{W}_{h_{\ell-1},h_\ell})\right] E\left[\sigma'(\hat{M}_{h_{\ell-1},h_\ell})\right] = 0 \tag{4}$$

Besides the issue of initialization, our empirical analysis (table 2) shows that this weight construction (1) do not create the desired bias for $\{-1, 0, 1\}$ for the addition and subtraction problem. This bias is desired as it restricts the solution space to exact addition, and in section also exact multiplication, which is an intrinsic property of an underlying arithmetic function. However, it does not necessarily restrict the output space as a plain linear transformation will always be able to scale values accordingly. The bias also adds interpretability which is important for being confident in a model's ability to extrapolate.

To solve these issues, we add a sparsifying regularizer to the loss function ($\mathcal{L} = \hat{\mathcal{L}} + \lambda_{\text{sparse}}\mathcal{R}_{\ell,\text{sparse}}$) and use simple linear weight construction, where $W_{h_{\ell-1},h_\ell}$ is clamped to $[-1, 1]$ in each iteration.

$$W_{h_{\ell-1},h_\ell} = \min(\max(W_{h_{\ell-1},h_\ell}, -1), 1), \tag{5}$$

$$\mathcal{R}_{\ell,\text{sparse}} = \frac{1}{H_\ell \cdot H_{\ell-1}} \sum_{h_\ell=1}^{H_\ell} \sum_{h_{\ell-1}=1}^{H_{\ell-1}} \min\left(|W_{h_{\ell-1},h_\ell}|, 1 - \left|W_{h_{\ell-1},h_\ell}\right|\right) \tag{6}$$

$$\text{NAU} : \; z_{h_\ell} = \sum_{h_{\ell-1}=1}^{H_{\ell-1}} W_{h_\ell,h_{\ell-1}} z_{h_{\ell-1}} \tag{7}$$

2.3 Challenges of division

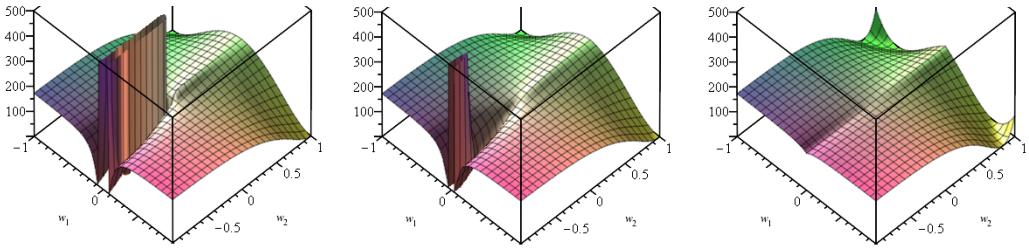
The $\text{NAC}_\bullet$, as formulated in equation 3, has the ability to perform exact multiplication and division, or more precisely multiplication of the inverse of elements from a vector, when a weight in $W_{h_{\ell-1},h_\ell}$ is -1.

However, this flexibility creates critical optimization challenges. Expanding the exp-log-transformation, $\mathrm{NAC}_\bullet$ can be expressed as

$$\mathrm{NAC}_\bullet : \; z_{h_\ell} = \prod_{h_{\ell-1}=1}^{H_{\ell-1}} \left(|z_{h_{\ell-1}}| + \epsilon\right)^{W_{h_\ell,h_{\ell-1}}} . \tag{8}$$

In equation (8), if $|z_{h_{\ell-1}}|$ is near zero ($E[z_{h_{\ell-1}}] = 0$ is a desired property when initializing (Glorot & Bengio, 2010)), $W_{h_{\ell-1},h_\ell}$ is negative, and ϵ is small, then the output will explode. This issue is present even for a reasonably large ϵ value (such as $\epsilon = 0.1$), and just a slightly negative $W_{h_{\ell-1},h_\ell}$, as visualized in figure 2. Also note that the curvature can cause convergence to an unstable area.

This singularity issue in the optimization space also makes multiplication challenging, which further suggests that supporting division is undesirable. These observations are also found empirically in Trask et al. (2018, table 1) and Appendix C.7.



(a) $\mathrm{NAC}_\bullet$ with $\epsilon = 10^{-7}$ (b) $\mathrm{NAC}_\bullet$ with $\epsilon = 0.1$ (c) $\mathrm{NAC}_\bullet$ with $\epsilon = 1$

Figure 2: RMS loss curvature for a NAC_+ layer followed by a $\mathrm{NAC}_\bullet$. The weight matrices are constrained to $\mathbf{W}_1 = \left[\begin{smallmatrix} w_1 & w_1 & 0 & 0 \\ w_1 & w_1 & w_1 & w_1 \end{smallmatrix}\right]$, $\mathbf{W}_2 = \left[\begin{smallmatrix} w_2 & w_2 \end{smallmatrix}\right]$. The problem is $(x_1+x_2)\cdot(x_1+x_2+x_3+x_4)$ for $x = (1, 1.2, 1.8, 2)$. The solution is $w_1 = w_2 = 1$ with many unstable alternatives.

2.4 Initialization of $\mathrm{NAC}_\bullet$

Initialization is important to consider for fast and consistent convergence. One desired property is that weights can be initialized such that $E[z_{h_\ell}] = 0$ (Glorot & Bengio, 2010). Using second order Taylor approximation and assuming all $z_{h_{\ell-1}}$ are uncorrelated; the expectation of $\mathrm{NAC}_\bullet$ can be estimated as

$$E[z_{h_\ell}] \approx \left(1 + \frac{1}{2} Var[W_{h_\ell,h_{\ell-1}}] \log(|E[z_{h_{\ell-1}}]| + \epsilon)^2\right)^{H_{\ell-1}} \Rightarrow E[z_{h_\ell}] > 1. \tag{9}$$

As shown in equation 9, satisfying $E[z_{h_\ell}] = 0$ for $\mathrm{NAC}_\bullet$ is likely impossible. The variance cannot be input-independently initialized and is expected to explode (proofs in Appendix B.3).

2.5 Neural Multiplication Unit

To solve the the gradient and initialization challenges for $\mathrm{NAC}_\bullet$ we propose a new unit for multiplication: the Neural Multiplication Unit (NMU)

$$W_{h_{\ell-1},h_\ell} = \min(\max(W_{h_{\ell-1},h_\ell}, 0), 1), \tag{10}$$

$$\mathcal{R}_{\ell,\text{sparse}} = \frac{1}{H_\ell \cdot H_{\ell-1}} \sum_{h_\ell=1}^{H_\ell} \sum_{h_{\ell-1}=1}^{H_{\ell-1}} \min\left(W_{h_{\ell-1},h_\ell}, 1 - W_{h_{\ell-1},h_\ell}\right) \tag{11}$$

$$\mathrm{NMU} : \; z_{h_\ell} = \prod_{h_{\ell-1}=1}^{H_{\ell-1}} \left(W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}\right) \tag{12}$$

The NMU is regularized similar to the NAU and has a multiplicative identity when $W_{h_{\ell-1},h_\ell} = 0$. The NMU do not support division by design. As opposed to the $\mathrm{NAC}_\bullet$, the NMU can represent input of both negative and positive values and is not ϵ bounded, which allows the NMU to extrapolate to $z_{h_{\ell-1}}$ that are negative or smaller than ϵ. Its gradients are derived in Appendix A.3.

2.6 MOMENTS AND INITIALIZATION

The NAU is a linear layer and can be initialized using Glorot & Bengio (2010). The NAC_+ unit can also achieve an ideal initialization, although it is less trivial (details in Appendix B.2).

The NMU is initialized with $E[W_{h_\ell,h_{\ell-1}}] = 1/2$. Assuming all $z_{h_{\ell-1}}$ are uncorrelated, and $E[z_{h_{\ell-1}}] = 0$, which is the case for most neural units (Glorot & Bengio, 2010), the expectation can be approximated to

$$E[z_{h_\ell}] \approx \left(\frac{1}{2}\right)^{H_{\ell-1}}, \tag{13}$$

which approaches zero for $H_{\ell-1} \to \infty$ (see Appendix B.4). The NMU can, assuming $Var[z_{h_{\ell-1}}] = 1$ and $H_{\ell-1}$ is large, be optimally initialized with $Var[W_{h_{\ell-1},h_\ell}] = \frac{1}{4}$ (proof in Appendix B.4.3).

2.7 REGULARIZER SCALING

We use the regularizer scaling as defined in (14). We motivate this by observing optimization consists of two parts: a warmup period, where $W_{h_{\ell-1},h_\ell}$ should get close to the solution, unhindered by the sparsity regularizer, followed by a period where the solution is made sparse.

$$\lambda_{\text{sparse}} = \hat{\lambda}_{\text{sparse}} \max\left(\min\left(\frac{t - \lambda_{\text{start}}}{\lambda_{\text{end}} - \lambda_{\text{start}}}, 1\right), 0\right) \tag{14}$$

2.8 CHALLENGES OF GATING BETWEEN ADDITION AND MULTIPLICATION

The purpose of the gating-mechanism is to select either NAC_+ or $\mathrm{NAC}_\bullet$ exclusively. This assumes that the correct sub-unit is selected by the NALU, since selecting the wrong sub-unit leaves no gradient signal for the correct sub-unit.

Empirically we find this assumption to be problematic. We observe that both sub-units converge in the beginning of training whereafter the gating-mechanism, seemingly randomly, converge towards either the addition or multiplication unit. Our study shows that gating behaves close to random for both NALU and a gated NMU/NAU variant. However, when the gate correctly selects multiplication our NMU converge much more consistently. (in-depth empirical analysis in appendix C.5, with results for both a shared and non-shared weight matrix between NAC_+ and $\mathrm{NAC}_\bullet$, and a gated version of NAU/NMU).

As the problem size grows, randomly choosing the correct gating value becomes an exponential increasing problem. Because of these challenges we leave solving the problem of sparse gating for future work and focus on improving the sub-units NAC_+ and $\mathrm{NAC}_\bullet$.

3 RELATED WORK

Pure neural models using convolutions, gating, differentiable memory, and/or attention architectures have attempted to learn arithmetic tasks through backpropagation (Kaiser & Sutskever, 2016; Kalchbrenner et al., 2016; Graves et al., 2014; Freivalds & Liepins, 2017). Some of these results have close to perfect extrapolation. However, the models are constrained to only work with well-defined arithmetic setups with no input redundancy, single operation and one-hot representations of numbers for input and output. We propose a model that has the flexibility to learn from hidden representations of a neural network where it has to work around redundancies and learn the underlying function.

The Neural Arithmetic Expression Calculator paper (Chen et al., 2018) propose learning real number arithmetic by having neural network subcomponents and repeatedly combines them through a memory-encoder-decoder architecture learned with hierarchical reinforcement learning. While this model has the ability to dynamically handle a larger variety of expressions compared to our solution they require an explicit definition of the operations.

In our experiments, the NAU is used to do a subset-selection, which is then followed by either a summation or a multiplication. An alternative, fully differentiable version, is to use a gumbel-softmax that can perform exact subset-selection (Xie & Ermon, 2019). However, this is restricted to a predefined subset size, which is a strong assumption that our units are not limited by.

4 EXPERIMENTAL RESULTS

4.1 ARITHMETIC DATASETS

The arithmetic dataset is a replica of the "simple function task" in Trask et al. (2018). The goal is to sum two random contiguous subsets of a vector and apply an arithmetic operation as defined in (15)

$$t = \sum_{i=s_{1,\text{start}}}^{s_{1,\text{end}}} x_i \circ \sum_{i=s_{2,\text{start}}}^{s_{2,\text{end}}} x_i \quad \text{where } \mathbf{x} \in \mathbb{R}^n, x_i \sim \text{Uniform}[r_{\text{lower}}, r_{\text{upper}}], \circ \in \{+, -, \times\} \tag{15}$$

where n (default 100), $U[r_{\text{lower}}, r_{\text{upper}}]$ (interpolation default is $U[1, 2]$ and extrapolation default is $U[2, 6]$), and other dataset parameters are used to assess learning capability (see details in appendix C.1 and the effect of varying the parameters in appendix C.4).

4.1.1 MODEL EVALUATION

We define the success-criterion as a solution that is acceptably close to a perfect solution. To evaluate if a model instance solves the task consistently we compare the MSE to a nearly-perfect solution on the extrapolation range over multiple seeds. If $\mathbf{W}_1, \mathbf{W}_2$ defines the weights of the fitted model, $\mathbf{W}_1^\epsilon$ is nearly-perfect and $\mathbf{W}_2^*$ is perfect (example in equation 16), then the criteria for successful convergence is $\mathcal{L}_{\mathbf{W}_1,\mathbf{W}_2} < \mathcal{L}_{\mathbf{W}_1^\epsilon,\mathbf{W}_2^*}$, measured on the extrapolation error, for $\epsilon = 10^{-5}$. We report a 95% confidence interval using a binomial distribution (Wilson, 1927).

$$\mathbf{W}_1^\epsilon = \begin{bmatrix} 1-\epsilon & 1-\epsilon & 0+\epsilon & 0+\epsilon \\ 1-\epsilon & 1-\epsilon & 1-\epsilon & 1-\epsilon \end{bmatrix}, \mathbf{W}_2^* = [1 \quad 1] \tag{16}$$

To measure speed of convergence of the first iteration for which $\mathcal{L}_{\mathbf{W}_1,\mathbf{W}_2} < \mathcal{L}_{\mathbf{W}_1^\epsilon,\mathbf{W}_2^*}$ is reported with a 95% confidence interval, calculated using a gamma distribution with maximum likelihood profiling. Only models that solved the task are included.

We assume an approximate discrete solution with parameters close to $\{-1, 0, 1\}$ is important for inferring exact arithmetic operations. To measure the sparsity we introduce a sparsity error (defined in equation 17). Similar to the convergence metric we only include model instances that did solve the task and report the 95% confidence interval, which is calculated using a beta distribution with maximum likelihood profiling.

$$E_{\text{sparsity}} = \max_{h_{\ell-1},h_\ell} \min(|W_{h_{\ell-1},h_\ell}|, |1 - |W_{h_{\ell-1},h_\ell}||) \tag{17}$$

We choose the optimal set of parameters based on the validation dataset (interpolation range).

4.1.2 ARITHMETIC OPERATION COMPARISON

We compare models on different arithmetic operation $\circ \in \{+, -, \times\}$. The multiplication models, NMU and $\text{NAC}_\bullet$, have an addition layer first, either NAU or NAC_+, followed by a multiplication layer. The addition models are just two layers of the same unit. The NALU model consists of two NALU layers. See explicit definitions and regularization values in appendix C.7.

Each experiment is trained for $5 \cdot 10^6$ iterations (details in appendix C.7). Results are presented in table 2. For multiplication, the NMU succeeds more often and converge faster. For addition NAU is comparable to a linear transformation in success-rate and convergence speed, but is more sparse. However, for subtraction a linear transformation can not consistently solve the task, while NAU and NAC_+ solves it. A larger comparison is in appendix C.7 and an ablation study is in appendix C.3.

4.1.3 EVALUATING THEORETICAL CLAIMS

To validate our theoretical claim, that the NMU model works better than $\text{NAC}_\bullet$ for larger $H_{\ell-1}$, we increase the hidden size of the network, thereby adding redundant units. Redundant units are very common neural networks, where fitting an unknown function is often done by overparameterization.

Additionally, the NMU model is unlike the $NAC_\bullet$ model also capable of handling inputs that are both negative and positive. To validate this empirically, the training and validation datasets are sampled for $\text{U}[-2, 2]$, and then tested on $\text{U}[-6, -2] \cup \text{U}[2, 6]$. The other ranges are defined in Appendix C.4.

Table 2: Comparison of the success-rate, when the model converged, and the sparsity error, with 95% confidence interval on the "arithmetic datasets" task. Each value is a summary of 100 different seeds.

Op	Model	Success	Solved at		Sparsity error
		Rate	Median	Mean	Mean
$\times$	$\mathrm{NAC}_\bullet$	$31\% \; {}^{+10\%}_{-8\%}$	$2.8 \cdot 10^6$	$3.0 \cdot 10^6 \; {}^{+2.9\cdot10^5}_{-2.4\cdot10^5}$	$5.8 \cdot 10^{-4} \; {}^{+4.8\cdot10^{-4}}_{-2.6\cdot10^{-4}}$
	NALU	$0\% \; {}^{+4\%}_{-0\%}$	—	—	—
	NMU	$\mathbf{98\%} \; {}^{+1\%}_{-5\%}$	$\mathbf{1.4 \cdot 10^6}$	$\mathbf{1.5 \cdot 10^6} \; {}^{+5.0\cdot10^4}_{-6.6\cdot10^4}$	$\mathbf{4.2 \cdot 10^{-7}} \; {}^{+2.9\cdot10^{-8}}_{-2.9\cdot10^{-8}}$
$+$	NAC_+	$\mathbf{100\%} \; {}^{+0\%}_{-4\%}$	$2.5 \cdot 10^5$	$4.9 \cdot 10^5 \; {}^{+5.2\cdot10^4}_{-4.5\cdot10^4}$	$2.3 \cdot 10^{-1} \; {}^{+6.5\cdot10^{-3}}_{-6.5\cdot10^{-3}}$
	Linear	$\mathbf{100\%} \; {}^{+0\%}_{-4\%}$	$6.1 \cdot 10^4$	$\mathbf{6.3 \cdot 10^4} \; {}^{+2.5\cdot10^3}_{-3.3\cdot10^3}$	$2.5 \cdot 10^{-1} \; {}^{+3.6\cdot10^{-4}}_{-3.6\cdot10^{-4}}$
	NALU	$14\% \; {}^{+8\%}_{-5\%}$	$1.5 \cdot 10^6$	$1.6 \cdot 10^6 \; {}^{+3.8\cdot10^5}_{-3.3\cdot10^5}$	$1.7 \cdot 10^{-1} \; {}^{+2.7\cdot10^{-2}}_{-2.5\cdot10^{-2}}$
	NAU	$\mathbf{100\%} \; {}^{+0\%}_{-4\%}$	$\mathbf{1.8 \cdot 10^4}$	$3.9 \cdot 10^5 \; {}^{+4.5\cdot10^4}_{-3.7\cdot10^4}$	$\mathbf{3.2 \cdot 10^{-5}} \; {}^{+1.3\cdot10^{-5}}_{-1.3\cdot10^{-5}}$
$-$	NAC_+	$\mathbf{100\%} \; {}^{+0\%}_{-4\%}$	$9.0 \cdot 10^3$	$3.7 \cdot 10^5 \; {}^{+3.8\cdot10^4}_{-3.8\cdot10^4}$	$2.3 \cdot 10^{-1} \; {}^{+5.4\cdot10^{-3}}_{-5.4\cdot10^{-3}}$
	Linear	$7\% \; {}^{+7\%}_{-4\%}$	$3.3 \cdot 10^6$	$1.4 \cdot 10^6 \; {}^{+7.0\cdot10^5}_{-6.1\cdot10^5}$	$1.8 \cdot 10^{-1} \; {}^{+7.2\cdot10^{-2}}_{-5.8\cdot10^{-2}}$
	NALU	$14\% \; {}^{+8\%}_{-5\%}$	$1.9 \cdot 10^6$	$1.9 \cdot 10^6 \; {}^{+4.4\cdot10^5}_{-4.5\cdot10^5}$	$2.1 \cdot 10^{-1} \; {}^{+2.2\cdot10^{-2}}_{-2.2\cdot10^{-2}}$
	NAU	$\mathbf{100\%} \; {}^{+0\%}_{-4\%}$	$\mathbf{5.0 \cdot 10^3}$	$\mathbf{1.6 \cdot 10^5} \; {}^{+1.7\cdot10^4}_{-1.6\cdot10^4}$	$\mathbf{6.6 \cdot 10^{-2}} \; {}^{+2.5\cdot10^{-2}}_{-1.9\cdot10^{-2}}$

Finally, for a fair comparison we introduce two new units: A variant of $\mathrm{NAC}_\bullet$ called $\mathrm{NAC}_{\bullet,\sigma}$, that only supports multiplication, by constraining the weights with $W = \sigma(\hat{W})$. And a variant, called $\mathrm{NAC}_{\bullet,\mathrm{NMU}}$, that uses linear weights and sparsity regularization, identically to that of the NMU.

Figure 3 shows that the NMU can both handle a much larger hidden-size, negative inputs, and that solving the division and bias issues alone improves the success rate, but are not sufficient when the hidden-size is large, as there is no ideal initialization. Interestingly, none of the models can learn the $\mathrm{U}[-1.1, 1.2]$, suggesting that certain input distributions might be troublesome to learn.

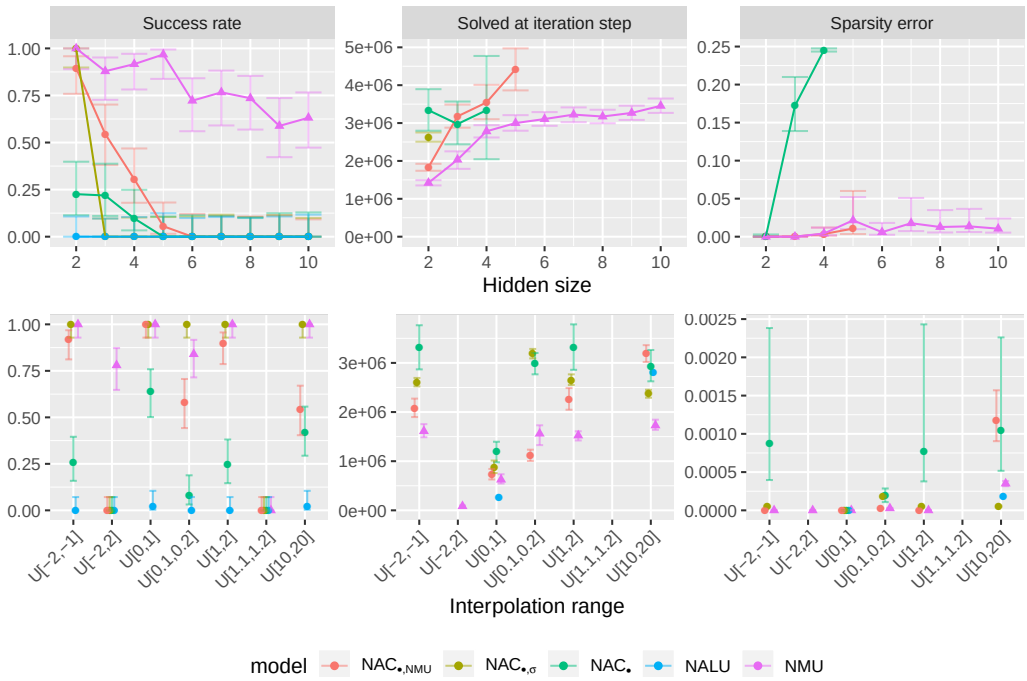


Figure 3: Shows that the NMU can handle a large hidden size, and works when the input contains both positive and negative numbers ($U[-2, 2]$), extrapolation ranges are defined in Appendix C.4.

4.2 PRODUCT OF SEQUENTIAL MNIST

To investigate if a deep neural network can be learned when backpropagating through an arithmetic layer, the arithmetic layers are used as a recurrent-unit to a sequence of MNIST digits, where the target is to fit the cumulative product. This task is similar to "MNIST Counting and Arithmetic Tasks" in Trask et al. (2018)[1], but uses multiplication rather than addition. Each model is trained on sequences of length 2, and then tested on sequences of length up to 20 MNIST digits.

We define the success-criterion by comparing the MSE of each model with a baseline model that has a correct solution for the arithmetic layer. If the MSE of each model, is less than the upper 1% MSE-confidence-interval of the baseline model, then the model is considered successfully converged.

Sparsity and "solved at iteration step" is determined as described in experiment 4.1. The validation set is the last 5000 MNIST digits from the training set, which is used to select the best epoch.

In this experiment we found that having an unconstrained "input-network" can cause the multiplication-units to learn an undesired solution, e.g. $(0.1 \cdot 81 + 1 - 0.1) = 9$. This solves the problem with a similar success-rate, but not in the intended way. To prevent such solution, we regularize the CNN output with $\mathcal{R}_z = \frac{1}{H_{\ell-1} H_\ell} \sum_{h_\ell}^{H_\ell} \sum_{h_{\ell-1}}^{H_{\ell-1}} (1 - W_{h_{\ell-1},h_\ell}) \cdot (1 - \bar{z}_{h_{\ell-1}})^2$. This regularizer is applied to the NMU and $\mathrm{NAC}_{\bullet,\mathrm{NMU}}$ models. See appendix D.4 for the results, when this regularizer is not used.

Figure 4 shows that the NMU does not hinder learning a more complex neural network. And that it can extrapolate to much longer sequences than what it is trained on.

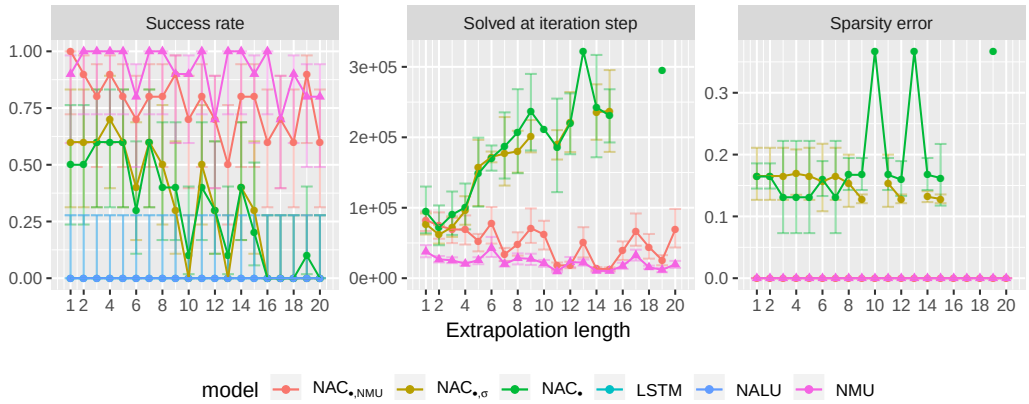


Figure 4: Shows the ability of each model to backpropagate and extrapolate to longer sequences.

5 CONCLUSION

By including theoretical considerations, such as initialization, gradients, and sparsity, we have developed a new neural multiplication unit (NMU) that outperforms the state-of-the-art models on established extrapolation and sequential tasks. Our model converge more consistently, faster, to more sparse solutions than previously proposed models, and supports all input ranges unlike NALU.

A natural next step would be to extend the NMU to support division and add gating between the NMU and NAU, to be comparable in theoretical features with NALU. However we find, both experimentally and theoretically, that learning the division is impractical, because of the singularity when dividing by zero, and that a sigmoid-gate that chooses between two functions with vastly different convergences properties, such as a multiplication unit and an addition unit, cannot be consistently learned.

Finally, when it comes to considering more than just two inputs to the multiplication layer, our model performs significantly better than previously proposed methods and variations of these. The ability for a neural layer to consider more than just two inputs, is critical in neural networks where the desired function is unknown.

[1]The same CNN is used, `https://github.com/pytorch/examples/tree/master/mnist`.

REFERENCES

Anonomous. Measuring arithmetic extrapolation performance. In *Science meets Engineering of Deep Learning at 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, volume abs/1910.01888, Vancouver, Canada, October 2019. URL http://arxiv.org/abs/1910.01888.

Kaiyu Chen, Yihan Dong, Xipeng Qiu, and Zitian Chen. Neural arithmetic expression calculator. *CoRR*, abs/1809.08590, 2018. URL http://arxiv.org/abs/1809.08590.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Karlis Freivalds and Renars Liepins. Improving the neural GPU architecture for algorithm learning. *CoRR*, abs/1702.08727, 2017. URL http://arxiv.org/abs/1702.08727.

C. R. Gallistel. Finding numbers in the brain. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 373(1740):20170119, 2018. doi: 10.1098/rstb.2017.0119. URL https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2017.0119.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9, pp. 249–256, May 2010.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL http://arxiv.org/abs/1410.5401.

Lukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.08228.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1507.01526.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *The 3rd International Conference for Learning Representations, San Diego, 2015*, pp. arXiv:1412.6980, Dec 2014.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 2879–2888, 2018. URL http://proceedings.mlr.press/v80/lake18a.html.

Andreas Nieder. The neuronal code for number. *Nature Reviews Neuroscience*, 17:366 EP –, 05 2016. URL https://doi.org/10.1038/nrn.2016.40.

OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub W. Pachocki, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018. URL http://arxiv.org/abs/1808.00177.

Rosa Rugani, Laura Fontanari, Eleonora Simoni, Lucia Regolin, and Giorgio Vallortigara. Arithmetic in newborn chicks. *Proceedings of the Royal Society B: Biological Sciences*, 276(1666):2451–2460, 2009. doi: 10.1098/rspb.2009.0044. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspb.2009.0044.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap,

Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961. URL https://doi.org/10.1038/nature16961.

Mirac Suzgun, Yonatan Belinkov, and Stuart M. Shieber. On evaluating the generalization of lstm models in formal languages. In *Proceedings of the Society for Computation in Linguistics (SCiL)*, pp. 277–286, January 2019.

Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 8035–8044. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8027-neural-arithmetic-logic-units.pdf.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1927.10502953.

Sang Michael Xie and Stefano Ermon. Differentiable subset sampling. *CoRR*, abs/1901.10517, 2019. URL http://arxiv.org/abs/1901.10517.

A GRADIENT DERIVATIVES

A.1 WEIGHT MATRIX CONSTRUCTION

For clarity the weight matrix construction is defined using scalar notation

$$W_{h_\ell,h_{\ell-1}} = \tanh(\hat{W}_{h_\ell,h_{\ell-1}})\sigma(\hat{M}_{h_\ell,h_{\ell-1}}) \tag{18}$$

The of the loss with respect to $\hat{W}_{h_\ell,h_{\ell-1}}$ and $\hat{M}_{h_\ell,h_{\ell-1}}$ is then derived using backpropagation.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \hat{W}_{h_\ell,h_{\ell-1}}} &= \frac{\partial \mathcal{L}}{\partial W_{h_\ell,h_{\ell-1}}} \frac{\partial W_{h_\ell,h_{\ell-1}}}{\partial \hat{W}_{h_\ell,h_{\ell-1}}} \\
&= \frac{\partial \mathcal{L}}{\partial W_{h_\ell,h_{\ell-1}}} (1 - \tanh^2(\hat{W}_{h_\ell,h_{\ell-1}}))\sigma(\hat{M}_{h_\ell,h_{\ell-1}}) \\
\frac{\partial \mathcal{L}}{\partial \hat{M}_{h_\ell,h_{\ell-1}}} &= \frac{\partial \mathcal{L}}{\partial W_{h_\ell,h_{\ell-1}}} \frac{\partial W_{h_\ell,h_{\ell-1}}}{\partial \hat{M}_{h_\ell,h_{\ell-1}}} \\
&= \frac{\partial \mathcal{L}}{\partial W_{h_\ell,h_{\ell-1}}} \tanh(\hat{W}_{h_\ell,h_{\ell-1}})\sigma(\hat{M}_{h_\ell,h_{\ell-1}})(1 - \sigma(\hat{M}_{h_\ell,h_{\ell-1}}))
\end{aligned} \tag{19}$$

As seen from this result, one only needs to consider $\frac{\partial \mathcal{L}}{\partial W_{h_\ell,h_{\ell-1}}}$ for NAC_+ and $\mathrm{NAC}_\bullet$, as the gradient with respect to $\hat{W}_{h_\ell,h_{\ell-1}}$ and $\hat{M}_{h_\ell,h_{\ell-1}}$ is a multiplication on $\frac{\partial \mathcal{L}}{\partial W_{h_\ell,h_{\ell-1}}}$.

A.2 GRADIENT OF $\mathrm{NAC}_\bullet$

The $\mathrm{NAC}_\bullet$ is defined using scalar notation.

$$z_{h_\ell} = \exp\left(\sum_{h_{\ell-1}=1}^{H_{\ell-1}} W_{h_\ell,h_{\ell-1}} \log(|z_{h_{\ell-1}}| + \epsilon)\right) \tag{20}$$

The gradient of the loss with respect to $W_{h_\ell,h_{\ell-1}}$ can the be derived using backpropagation.

$$\begin{aligned}
\frac{\partial z_{h_\ell}}{\partial W_{h_\ell,h_{\ell-1}}} &= \exp\left(\sum_{h'_{\ell-1}=1}^{H_{\ell-1}} W_{h_\ell,h'_{\ell-1}} \log(|z_{h'_{\ell-1}}| + \epsilon)\right) \log(|z_{h_{\ell-1}}| + \epsilon) \\
&= z_{h_\ell} \log(|z_{h_{\ell-1}}| + \epsilon)
\end{aligned} \tag{21}$$

We now wish to derive the backpropagation term $\delta_{h_\ell} = \frac{\partial \mathcal{L}}{\partial z_{h_\ell}}$, because z_{h_ℓ} affects $\{z_{h_{\ell+1}}\}_{h_{\ell+1}=1}^{H_{\ell+1}}$ this becomes:

$$\delta_{h_\ell} = \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} = \sum_{h_{\ell+1}=1}^{H_{\ell+1}} \frac{\partial \mathcal{L}}{\partial z_{h_{\ell+1}}} \frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}} = \sum_{h_{\ell+1}=1}^{H_{\ell+1}} \delta_{h_{\ell+1}} \frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}} \tag{22}$$

To make it easier to derive $\frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}}$ we re-express the z_{h_ℓ} as $z_{h_{\ell+1}}$.

$$z_{h_{\ell+1}} = \exp\left(\sum_{h_\ell=1}^{H_\ell} W_{h_{\ell+1},h_\ell} \log(|z_{h_\ell}| + \epsilon)\right) \tag{23}$$

The gradient of $\frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}}$ is then:

$$\begin{aligned}
\frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}} &= \exp\left(\sum_{h_\ell=1}^{H_\ell} W_{h_{\ell+1},h_\ell} \log(|z_{h_\ell}| + \epsilon)\right) W_{h_{\ell+1},h_\ell} \frac{\partial \log(|z_{h_\ell}| + \epsilon)}{\partial z_{h_\ell}} \\
&= \exp\left(\sum_{h_\ell=1}^{H_\ell} W_{h_{\ell+1},h_\ell} \log(|z_{h_\ell}| + \epsilon)\right) W_{h_{\ell+1},h_\ell} \frac{\mathrm{abs}'(z_{h_\ell})}{|z_{h_\ell}| + \epsilon} \\
&= z_{h_{\ell+1}} W_{h_{\ell+1},h_\ell} \frac{\mathrm{abs}'(z_{h_\ell})}{|z_{h_\ell}| + \epsilon}
\end{aligned} \tag{24}$$

$\mathrm{abs}'(z_{h_\ell})$ is the gradient of the absolute function. In the paper we denote this as $\mathrm{sign}(z_{h_\ell})$ for brevity. However, depending on the exact definition used there may be a difference for $z_{h_\ell} = 0$, as $\mathrm{abs}'(0)$ is undefined. In practicality this doesn't matter much though, although theoretically it does mean that the expectation of this is theoretically undefined when $E[z_{h_\ell}] = 0$.

A.3 Gradient of NMU

In scalar notation the NMU is defined as:

$$z_{h_\ell} = \prod_{h_{\ell-1}=1}^{H_{\ell-1}} \left(W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}\right) \tag{25}$$

The gradient of the loss with respect to $W_{h_{\ell-1},h_\ell}$ is fairly trivial. Note that every term but the one for $h_{\ell-1}$, is just a constant with respect to $W_{h_{\ell-1},h_\ell}$. The product, except the term for $h_{\ell-1}$ can be expressed as $\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}$. Using this fact, the gradient can be expressed as:

$$\frac{\partial \mathcal{L}}{\partial w_{h_\ell,h_{\ell-1}}} = \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} \frac{\partial z_{h_\ell}}{\partial w_{h_\ell,h_{\ell-1}}} = \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} \frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}} \left(z_{h_{\ell-1}} - 1\right) \tag{26}$$

Similarly, the gradient $\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}$ which is essential in backpropagation can equally easily be derived as:

$$\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}} = \sum_{h_\ell=1}^{H_\ell} \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} \frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}} = \sum_{h_\ell=1}^{H_\ell} \frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}} W_{h_{\ell-1},h_\ell} \tag{27}$$

B MOMENTS

B.1 OVERVIEW

B.1.1 MOMENTS AND INITIALIZATION FOR ADDITION

The desired properties for initialization are according to Glorot et al. (Glorot & Bengio, 2010):

$$\begin{aligned} E[z_{h_\ell}] &= 0 & E\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &= 0 \\ Var[z_{h_\ell}] &= Var\left[z_{h_{\ell-1}}\right] & Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &= Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] \end{aligned} \tag{28}$$

B.1.2 INITIALIZATION FOR ADDITION

Glorot initialization can not be used for NAC_+ as $W_{h_{\ell-1},h_\ell}$ is not sampled directly. Assuming that $\hat{W}_{h_\ell,h_{\ell-1}} \sim \mathrm{Uniform}[-r,r]$ and $\hat{M}_{h_\ell,h_{\ell-1}} \sim \mathrm{Uniform}[-r,r]$, then the variance can be derived (see proof in Appendix B.2) to be:

$$Var[W_{h_{\ell-1},h_\ell}] = \frac{1}{2r}\left(1 - \frac{\tanh(r)}{r}\right)\left(r - \tanh\left(\frac{r}{2}\right)\right) \tag{29}$$

One can then solve for r, given the desired variance ($Var[W_{h_{\ell-1},h_\ell}] = \frac{2}{H_{\ell-1}+H_\ell}$) (Glorot & Bengio, 2010).

B.1.3 MOMENTS AND INITIALIZATION FOR MULTIPLICATION

Using second order multivariate Taylor approximation and some assumptions of uncorrelated stochastic variables, the expectation and variance of the $\mathrm{NAC}_\bullet$ layer can be estimated to:

$$\begin{aligned} f(c_1,c_2) &= \left(1 + c_1 \frac{1}{2} Var[W_{h_\ell,h_{\ell-1}}] \log(|E[z_{h_{\ell-1}}]| + \epsilon)^2\right)^{c_2\ H_{\ell-1}} \\ E[z_{h_\ell}] &\approx f\left(1,1\right) \\ Var[z_{h_\ell}] &\approx f\left(4,1\right) - f\left(1,2\right) \\ E\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &= 0 \\ Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &\approx Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] H_\ell\ f\left(4,1\right)\ Var[W_{h_\ell,h_{\ell-1}}] \\ &\quad \cdot \left(\frac{1}{\left(|E[z_{h_{\ell-1}}]| + \epsilon\right)^2} + \frac{3}{\left(|E[z_{h_{\ell-1}}]| + \epsilon\right)^4} Var[z_{h_{\ell-1}}]\right) \end{aligned} \tag{30}$$

This is problematic because $E[z_{h_\ell}] \geq 1$, and the variance explodes for $E[z_{h_{\ell-1}}] = 0$. $E[z_{h_{\ell-1}}] = 0$ is normally a desired property (Glorot & Bengio, 2010). The variance explodes for $E[z_{h_{\ell-1}}] = 0$, and can thus not be initialized to anything meaningful.

For our proposed NMU, the expectation and variance can be derived (see proof in Appendix B.4) using the same assumptions as before, although no Taylor approximation is required:

$$
\begin{aligned}
E[z_{h_\ell}] &\approx \left(\frac{1}{2}\right)^{H_{\ell-1}} \\
E\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &\approx 0 \\
Var[z_{h_\ell}] &\approx \left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right)^{H_{\ell-1}} \left(Var[z_{h_{\ell-1}}] + 1\right)^{H_{\ell-1}} - \left(\frac{1}{4}\right)^{H_{\ell-1}} \\
Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &\approx Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] H_\ell \\
&\quad \cdot \left(\left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right)^{H_{\ell-1}} \left(Var[z_{h_{\ell-1}}] + 1\right)^{H_{\ell-1}-1}\right)
\end{aligned}
\tag{31}
$$

These expectations are better behaved. It is unlikely that the expectation of a multiplication unit can become zero, since the identity for multiplication is 1. However, for a large $H_{\ell-1}$ it will be near zero.

The variance is also better behaved, but do not provide a input-independent initialization strategy. We propose initializing with $Var[W_{h_{\ell-1},h_\ell}] = \frac{1}{4}$, as this is the solution to $Var[z_{h_\ell}] = Var[z_{h_{\ell-1}}]$ assuming $Var[z_{h_{\ell-1}}] = 1$ and a large $H_{\ell-1}$ (see proof in Appendix B.4.3). However, more exact solutions are possible if the input variance is known.

B.2 Expectation and variance for weight matrix construction in NAC layers

The weight matrix construction in NAC, is defined in scalar notation as:

$$
W_{h_\ell,h_{\ell-1}} = \tanh(\hat{W}_{h_\ell,h_{\ell-1}})\sigma(\hat{M}_{h_\ell,h_{\ell-1}}) \tag{32}
$$

Simplifying the notation of this, and re-expressing it using stochastic variables with uniform distributions this can be written as:

$$
\begin{aligned}
W &\sim \tanh(\hat{W})\sigma(\hat{M}) \\
\hat{W} &\sim\ U[-r, r] \\
\hat{M} &\sim\ U[-r, r]
\end{aligned}
\tag{33}
$$

Since $\tanh(\hat{W})$ is an odd-function and $E[\hat{W}] = 0$, deriving the expectation $E[W]$ is trivial.

$$
\mathrm{E}[W] = \mathrm{E}[\tanh(\hat{W})]\mathrm{E}[\sigma(\hat{M})] = 0 \cdot \mathrm{E}[\sigma(\hat{M})] = 0 \tag{34}
$$

The variance is more complicated, however as $\hat{W}$ and $\hat{M}$ are independent, it can be simplified to:

$$
\mathrm{Var}[W] = \mathrm{E}[\tanh(\hat{W})^2]\mathrm{E}[\sigma(\hat{M})^2] - \mathrm{E}[\tanh(\hat{W})]^2\mathrm{E}[\sigma(\hat{M})]^2 = \mathrm{E}[\tanh(\hat{W})^2]\mathrm{E}[\sigma(\hat{M})^2] \tag{35}
$$

These second moments can be analyzed independently. First for $\mathrm{E}[\tanh(\hat{W})^2]$:

$$
\begin{aligned}
\mathrm{E}[\tanh(\hat{W})^2] &= \int_{-\infty}^{\infty} \tanh(x)^2 f_{U[-r,r]}(x)\, \mathrm{d}x \\
&= \frac{1}{2r}\int_{-r}^{r} \tanh(x)^2\, \mathrm{d}x \\
&= \frac{1}{2r} \cdot 2 \cdot (r - \tanh(r)) \\
&= 1 - \frac{\tanh(r)}{r}
\end{aligned}
\tag{36}
$$

Then for $\mathrm{E}[\tanh(\hat{M})^2]$:

$$\begin{aligned}\mathrm{E}[\sigma(\hat{M})^2] &= \int_{-\infty}^{\infty} \sigma(x)^2 f_{U[-r,r]}(x)\,\mathrm{d}x \\ &= \frac{1}{2r}\int_{-r}^{r} \sigma(x)^2\,\mathrm{d}x \\ &= \frac{1}{2r}\left(r - \tanh\left(\frac{r}{2}\right)\right)\end{aligned} \tag{37}$$

Which results in the variance:

$$\mathrm{Var}[W] = \frac{1}{2r}\left(1 - \frac{\tanh(r)}{r}\right)\left(r - \tanh\left(\frac{r}{2}\right)\right) \tag{38}$$

B.3 Expectation and variance of NAC$_\bullet$

B.3.1 Forward pass

Expectation Assuming that each $z_{h_{\ell-1}}$ are uncorrelated, the expectation can be simplified to:

$$\begin{aligned}E[z_{h_\ell}] &= E\left[\exp\left(\sum_{h_{\ell-1}=1}^{H_{\ell-1}} W_{h_\ell,h_{\ell-1}} \log(|z_{h_{\ell-1}}| + \epsilon)\right)\right] \\ &= E\left[\prod_{h_{\ell-1}=1}^{H_{\ell-1}} \exp(W_{h_\ell,h_{\ell-1}} \log(|z_{h_{\ell-1}}| + \epsilon))\right] \\ &\approx \prod_{h_{\ell-1}=1}^{H_{\ell-1}} E[\exp(W_{h_\ell,h_{\ell-1}} \log(|z_{h_{\ell-1}}| + \epsilon))] \\ &= E[\exp(W_{h_\ell,h_{\ell-1}} \log(|z_{h_{\ell-1}}| + \epsilon))]^{H_{\ell-1}} \\ &= E\left[(|z_{h_{\ell-1}}| + \epsilon)^{W_{h_\ell,h_{\ell-1}}}\right]^{H_{\ell-1}} \\ &= E\left[f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})\right]^{H_{\ell-1}}\end{aligned} \tag{39}$$

Here we define g as a non-linear transformation function of two independent stochastic variables:

$$f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}}) = (|z_{h_{\ell-1}}| + \epsilon)^{W_{h_\ell,h_{\ell-1}}} \tag{40}$$

We then apply second order Taylor approximation of f, around $(E[z_{h_{\ell-1}}], E[W_{h_\ell,h_{\ell-1}}])$.

$$\begin{aligned}&E[f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})] \approx E\Bigg[\\ &f(E[z_{h_{\ell-1}}], E[W_{h_\ell,h_{\ell-1}}]) \\ &+ \begin{bmatrix} z_{h_{\ell-1}} - E[z_{h_{\ell-1}}] \\ W_{h_\ell,h_{\ell-1}} - E[W_{h_\ell,h_{\ell-1}}] \end{bmatrix}^T \begin{bmatrix} \frac{\partial f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})}{\partial z_{h_{\ell-1}}} \\ \frac{\partial f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})}{\partial W_{h_\ell,h_{\ell-1}}} \end{bmatrix} \Bigg|_{\begin{cases} z_{h_{\ell-1}} = E[z_{h_{\ell-1}}] \\ W_{h_\ell,h_{\ell-1}} = E[W_{h_\ell,h_{\ell-1}}] \end{cases}} \\ &+ \frac{1}{2}\begin{bmatrix} z_{h_{\ell-1}} - E[z_{h_{\ell-1}}] \\ W_{h_\ell,h_{\ell-1}} - E[W_{h_\ell,h_{\ell-1}}] \end{bmatrix}^T \\ &\bullet \begin{bmatrix} \frac{\partial^2 f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})}{\partial^2 z_{h_{\ell-1}}} & \frac{\partial^2 f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})}{\partial z_{h_{\ell-1}} \partial W_{h_\ell,h_{\ell-1}}} \\ \frac{\partial^2 f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})}{\partial z_{h_{\ell-1}} \partial W_{h_\ell,h_{\ell-1}}} & \frac{\partial^2 f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})}{\partial^2 W_{h_\ell,h_{\ell-1}}} \end{bmatrix} \Bigg|_{\begin{cases} z_{h_{\ell-1}} = E[z_{h_{\ell-1}}] \\ W_{h_\ell,h_{\ell-1}} = E[W_{h_\ell,h_{\ell-1}}] \end{cases}} \\ &\bullet \begin{bmatrix} z_{h_{\ell-1}} - E[z_{h_{\ell-1}}] \\ W_{h_\ell,h_{\ell-1}} - E[W_{h_\ell,h_{\ell-1}}] \end{bmatrix} \Bigg]\end{aligned} \tag{41}$$

Because $E[z_{h_{\ell-1}} - E[z_{h_{\ell-1}}]] = 0$, $E[W_{h_\ell,h_{\ell-1}} - E[W_{h_\ell,h_{\ell-1}}]] = 0$, and $Cov[z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}}] = 0$. This simplifies to:

$$\begin{aligned} &E[g(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})] \approx g(E[z_{h_{\ell-1}}], E[W_{h_\ell,h_{\ell-1}}]) \\ &+ \frac{1}{2} Var \begin{bmatrix} z_{h_{\ell-1}} \\ W_{h_\ell,h_{\ell-1}} \end{bmatrix}^T \begin{bmatrix} \frac{\partial^2 g(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})}{\partial^2 z_{h_{\ell-1}}} \\ \frac{\partial^2 g(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})}{\partial^2 W_{h_\ell,h_{\ell-1}}} \end{bmatrix} \Bigg|_{\begin{cases} z_{h_{\ell-1}} = E[z_{h_{\ell-1}}] \\ W_{h_\ell,h_{\ell-1}} = E[W_{h_\ell,h_{\ell-1}}] \end{cases}} \end{aligned} \tag{42}$$

Inserting the derivatives and computing the inner products yields:

$$\begin{aligned} &E[f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})] \approx (|E[z_{h_{\ell-1}}]| + \epsilon)^{E[W_{h_\ell,h_{\ell-1}}]} \\ &+ \frac{1}{2} Var[z_{h_{\ell-1}}](|E[z_{h_{\ell-1}}]| + \epsilon)^{E[W_{h_\ell,h_{\ell-1}}]-2} E[W_{h_\ell,h_{\ell-1}}](E[W_{h_\ell,h_{\ell-1}}] - 1) \\ &+ \frac{1}{2} Var[W_{h_\ell,h_{\ell-1}}](|E[z_{h_{\ell-1}}]| + \epsilon)^{E[W_{h_\ell,h_{\ell-1}}]} \log(|E[z_{h_{\ell-1}}]| + \epsilon)^2 \\ &= 1 + \frac{1}{2} Var[W_{h_\ell,h_{\ell-1}}] \log(|E[z_{h_{\ell-1}}]| + \epsilon)^2 \end{aligned} \tag{43}$$

This gives the final expectation:

$$\begin{aligned} E[z_{h_\ell}] &= E\left[g(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})\right]^{H_{\ell-1}} \\ &\approx \left(1 + \frac{1}{2} Var[W_{h_\ell,h_{\ell-1}}] \log(|E[z_{h_{\ell-1}}]| + \epsilon)^2\right)^{H_{\ell-1}} \end{aligned} \tag{44}$$

We evaluate the error of the approximation, where $W_{h_\ell,h_{\ell-1}} \sim U[-r_w, r_w]$ and $z_{h_{\ell-1}} \sim U[0, r_z]$. These distributions are what is used in the arithmetic dataset. The error is plotted in figure 5.

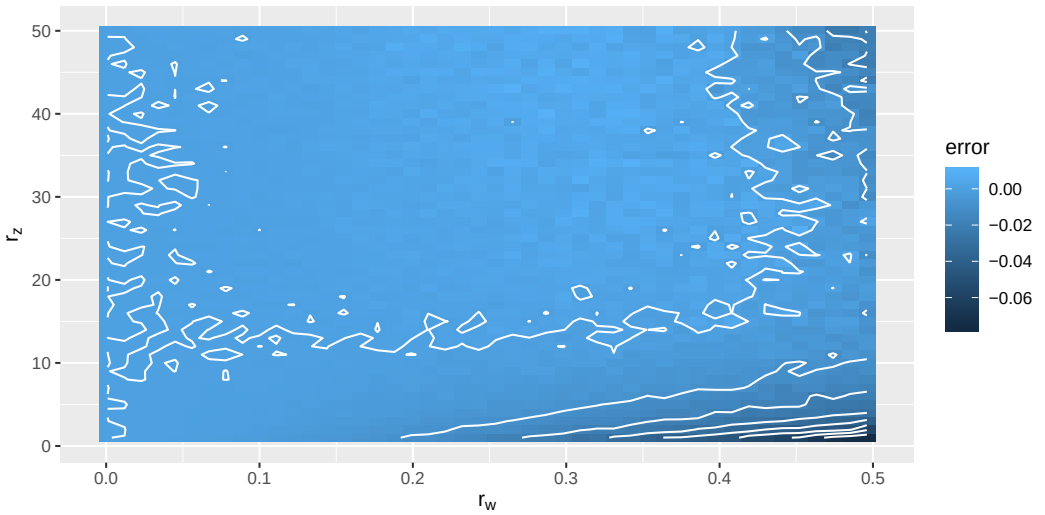


Figure 5: Error between theoretical approximation and the numerical approximation estimated by random sampling of 100000 observations at each combination of r_z and r_w.

Variance The variance can be derived using the same assumptions as used in "expectation", that all $z_{h_{\ell-1}}$ are uncorrelated.

$$\begin{aligned} Var[z_{h_\ell}] &= E[z_{h_\ell}^2] - E[z_{h_\ell}]^2 \\ &= E\left[\prod_{h_{\ell-1}=1}^{H_{\ell-1}} (|z_{h_{\ell-1}}| + \epsilon)^{2 \cdot W_{h_\ell,h_{\ell-1}}}\right] - E\left[\prod_{h_{\ell-1}=1}^{H_{\ell-1}} (|z_{h_{\ell-1}}| + \epsilon)^{W_{h_\ell,h_{\ell-1}}}\right]^2 \\ &= E\left[f(z_{h_{\ell-1}}, 2 \cdot W_{h_\ell,h_{\ell-1}})\right]^{H_{\ell-1}} - E\left[f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})\right]^{2 \cdot H_{\ell-1}} \end{aligned} \tag{45}$$

We already have from the expectation result in (43) that:

$$E\left[f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})\right] \approx 1 + \frac{1}{2}Var[W_{h_\ell,h_{\ell-1}}]\log(|E[z_{h_{\ell-1}}]| + \epsilon)^2 \tag{46}$$

By substitution of variable we have that:

$$\begin{aligned} E\left[f(z_{h_{\ell-1}}, 2\cdot W_{h_\ell,h_{\ell-1}})\right] &\approx 1 + \frac{1}{2}Var[2\cdot W_{h_\ell,h_{\ell-1}}]\log(|E[z_{h_{\ell-1}}]| + \epsilon)^2 \\ &\approx 1 + 2\cdot Var[W_{h_\ell,h_{\ell-1}}]\log(|E[z_{h_{\ell-1}}]| + \epsilon)^2 \end{aligned} \tag{47}$$

This gives the variance:

$$\begin{aligned} Var[z_{h_\ell}] &= E\left[g(z_{h_{\ell-1}}, 2\cdot W_{h_\ell,h_{\ell-1}})\right]^{H_{\ell-1}} - E\left[f(z_{h_{\ell-1}}, W_{h_\ell,h_{\ell-1}})\right]^{2\cdot H_{\ell-1}} \\ &\approx \left(1 + 2\cdot Var[W_{h_\ell,h_{\ell-1}}]\log(|E[z_{h_{\ell-1}}]| + \epsilon)^2\right)^{H_{\ell-1}} \\ &- \left(1 + \frac{1}{2}\cdot Var[W_{h_\ell,h_{\ell-1}}]\log(|E[z_{h_{\ell-1}}]| + \epsilon)^2\right)^{2\cdot H_{\ell-1}} \end{aligned} \tag{48}$$

B.3.2 Backward pass

Expectation The expectation of the back-propagation term assuming that $\delta_{h_{\ell+1}}$ and $\frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}}$ are mutually uncorrelated:

$$E[\delta_{h_\ell}] = E\left[\sum_{h_{\ell+1}=1}^{H_{\ell+1}} \delta_{h_{\ell+1}} \frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}}\right] \approx H_{\ell+1}E[\delta_{h_{\ell+1}}]E\left[\frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}}\right] \tag{49}$$

Assuming that $z_{h_{\ell+1}}$, $W_{h_{\ell+1},h_\ell}$, and z_{h_ℓ} are uncorrelated:

$$E\left[\frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}}\right] \approx E[z_{h_{\ell+1}}]E[W_{h_{\ell+1},h_\ell}]E\left[\frac{\text{abs}'(z_{h_\ell})}{|z_{h_\ell}| + \epsilon}\right] = E[z_{h_{\ell+1}}]\cdot 0\cdot E\left[\frac{\text{abs}'(z_{h_\ell})}{|z_{h_\ell}| + \epsilon}\right] = 0 \tag{50}$$

Variance Deriving the variance is more complicated:

$$Var\left[\frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}}\right] = Var\left[z_{h_{\ell+1}} W_{h_{\ell+1},h_\ell}\frac{\text{abs}'(z_{h_\ell})}{|z_{h_\ell}| + \epsilon}\right] \tag{51}$$

Assuming again that $z_{h_{\ell+1}}$, $W_{h_{\ell+1},h_\ell}$, and z_{h_ℓ} are uncorrelated, and likewise for their second moment:

$$\begin{aligned} Var\left[\frac{\partial z_{h_{\ell+1}}}{\partial z_{h_\ell}}\right] &\approx E[z^2_{h_{\ell+1}}]E[W^2_{h_{\ell+1},h_\ell}]E\left[\left(\frac{\text{abs}'(z_{h_\ell})}{|z_{h_\ell}| + \epsilon}\right)^2\right] \\ &- E[z_{h_{\ell+1}}]^2 E[W_{h_{\ell+1},h_\ell}]^2 E\left[\frac{\text{abs}'(z_{h_\ell})}{|z_{h_\ell}| + \epsilon}\right]^2 \\ &= E[z^2_{h_{\ell+1}}]Var[W_{h_{\ell+1},h_\ell}]E\left[\left(\frac{\text{abs}'(z_{h_\ell})}{|z_{h_\ell}| + \epsilon}\right)^2\right] \\ &- E[z_{h_{\ell+1}}]^2\cdot 0\cdot E\left[\frac{\text{abs}'(z_{h_\ell})}{|z_{h_\ell}| + \epsilon}\right]^2 \\ &= E[z^2_{h_{\ell+1}}]Var[W_{h_{\ell+1},h_\ell}]E\left[\left(\frac{\text{abs}'(z_{h_\ell})}{|z_{h_\ell}| + \epsilon}\right)^2\right] \end{aligned} \tag{52}$$

Using Taylor approximation around $E[z_{h_\ell}]$ we have:

$$\begin{aligned} E\left[\left(\frac{\text{abs}'(z_{h_\ell})}{|z| + \epsilon}\right)^2\right] &\approx \frac{1}{(|E[z_{h_\ell}]| + \epsilon)^2} + \frac{1}{2}\frac{6}{(|E[z_{h_\ell}]| + \epsilon)^4}Var[z_{h_\ell}] \\ &= \frac{1}{(|E[z_{h_\ell}]| + \epsilon)^2} + \frac{3}{(|E[z_{h_\ell}]| + \epsilon)^4}Var[z_{h_\ell}] \end{aligned} \tag{53}$$

Finally, by reusing the result for $E[z_{h_\ell}^2]$ from earlier the variance can be expressed as:

$$\begin{aligned} Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] \approx Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] H_\ell \ \left(1 + 2 \cdot Var[W_{h_\ell,h_{\ell-1}}] \log(|E[z_{h_{\ell-1}}]| + \epsilon)^2\right)^{H_{\ell-1}} \\ \cdot Var[W_{h_\ell,h_{\ell-1}}] \left(\frac{1}{\left(|E[z_{h_{\ell-1}}]| + \epsilon\right)^2} + \frac{3}{\left(|E[z_{h_{\ell-1}}]| + \epsilon\right)^4} Var[z_{h_{\ell-1}}]\right) \end{aligned} \tag{54}$$

B.4 Expectation and variance of NMU

B.4.1 Forward pass

Expectation Assuming that all $z_{h_{\ell-1}}$ are independent:

$$\begin{aligned} E[z_{h_\ell}] &= E\left[\prod_{h_{\ell-1}=1}^{H_{\ell-1}} \left(W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}\right)\right] \\ &\approx E\left[W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}\right]^{H_{\ell-1}} \\ &\approx \left(E[W_{h_{\ell-1},h_\ell}] E[z_{h_{\ell-1}}] + 1 - E[W_{h_{\ell-1},h_\ell}]\right)^{H_{\ell-1}} \end{aligned} \tag{55}$$

Assuming that $E[z_{h_{\ell-1}}] = 0$ which is a desired property and initializing $E[W_{h_{\ell-1},h_\ell}] = 1/2$, the expectation is:

$$\begin{aligned} E[z_{h_\ell}] &\approx \left(E[W_{h_{\ell-1},h_\ell}] E[z_{h_{\ell-1}}] + 1 - E[W_{h_{\ell-1},h_\ell}]\right)^{H_{\ell-1}} \\ &\approx \left(\frac{1}{2} \cdot 0 + 1 - \frac{1}{2}\right)^{H_{\ell-1}} \\ &= \left(\frac{1}{2}\right)^{H_{\ell-1}} \end{aligned} \tag{56}$$

Variance Reusing the result for the expectation, assuming again that all $z_{h_{\ell-1}}$ are uncorrelated, and using the fact that $W_{h_{\ell-1},h_\ell}$ is initially independent from $z_{h_{\ell-1}}$:

$$\begin{aligned} Var[z_{h_\ell}] &= E[z_{h_\ell}^2] - E[z_{h_\ell}]^2 \\ &\approx E[z_{h_\ell}^2] - \left(\frac{1}{2}\right)^{2 \cdot H_{\ell-1}} \\ &= E\left[\prod_{h_{\ell-1}=1}^{H_{\ell-1}} \left(W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}\right)^2\right] - \left(\frac{1}{2}\right)^{2 \cdot H_{\ell-1}} \\ &\approx E[\left(W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}\right)^2]^{H_{\ell-1}} - \left(\frac{1}{2}\right)^{2 \cdot H_{\ell-1}} \\ &= \Big(E[W_{h_{\ell-1},h_\ell}^2] E[z_{h_{\ell-1}}^2] - 2E[W_{h_{\ell-1},h_\ell}^2] E[z_{h_{\ell-1}}] + E[W_{h_{\ell-1},h_\ell}^2] \\ &\quad + 2E[W_{h_{\ell-1},h_\ell}] E[z_{h_{\ell-1}}] - 2E[W_{h_{\ell-1},h_\ell}] + 1\Big)^{H_{\ell-1}} - \left(\frac{1}{2}\right)^{2 \cdot H_{\ell-1}} \end{aligned} \tag{57}$$

Assuming that $E[z_{h_{\ell-1}}] = 0$, which is a desired property and initializing $E[W_{h_{\ell-1},h_\ell}] = 1/2$, the variance becomes:

$$\begin{aligned} Var[z_{h_\ell}] &\approx \left(E[W_{h_{\ell-1},h_\ell}^2] \left(E[z_{h_{\ell-1}}^2] + 1\right)\right)^{H_{\ell-1}} - \left(\frac{1}{2}\right)^{2 \cdot H_{\ell-1}} \\ &\approx \left(\left(Var[W_{h_{\ell-1},h_\ell}] + E[W_{h_{\ell-1},h_\ell}]^2\right) \left(Var[z_{h_{\ell-1}}] + 1\right)\right)^{H_{\ell-1}} - \left(\frac{1}{2}\right)^{2 \cdot H_{\ell-1}} \\ &= \left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right)^{H_{\ell-1}} \left(Var[z_{h_{\ell-1}}] + 1\right)^{H_{\ell-1}} - \left(\frac{1}{2}\right)^{2 \cdot H_{\ell-1}} \end{aligned} \tag{58}$$

B.4.2 BACKWARD PASS

Expectation For the backward pass the expectation can, assuming that $\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}$ and $\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}$ are uncorrelated, be derived to:

$$\begin{aligned} E\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &= H_\ell E\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right] \\ &\approx H_\ell E\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] E\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right] \\ &= H_\ell E\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] E\left[\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}} W_{h_{\ell-1},h_\ell}\right] \\ &= H_\ell E\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] E\left[\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}\right] E\left[W_{h_{\ell-1},h_\ell}\right] \end{aligned} \tag{59}$$

Initializing $E[W_{h_{\ell-1},h_\ell}] = 1/2$, and inserting the result for the expectation $E\left[\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}\right]$.

$$\begin{aligned} E\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &\approx H_\ell E\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right]\left(\frac{1}{2}\right)^{H_{\ell-1}-1}\frac{1}{2} \\ &= E\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] H_\ell \left(\frac{1}{2}\right)^{H_{\ell-1}} \end{aligned} \tag{60}$$

Assuming that $E\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] = 0$, which is a desired property (Glorot & Bengio, 2010).

$$\begin{aligned} E\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &\approx 0 \cdot H_\ell \cdot \left(\frac{1}{2}\right)^{H_{\ell-1}} \\ &= 0 \end{aligned} \tag{61}$$

Variance For the variance of the backpropagation term, we assume that $\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}$ is uncorrelated with $\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}$.

$$\begin{aligned} Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &= H_\ell Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right] \\ &\approx H_\ell \Bigg(Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] E\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right]^2 + E\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right]^2 Var\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right] \\ &+ Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] Var\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right]\Bigg) \end{aligned} \tag{62}$$

Assuming again that $E\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] = 0$, and reusing the result $E\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right] = \left(\frac{1}{2}\right)^{H_{\ell-1}}$.

$$Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] \approx Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] H_\ell \left(\left(\frac{1}{2}\right)^{2\cdot H_{\ell-1}} + Var\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right]\right) \tag{63}$$

Focusing now on $Var\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right]$, we have:

$$\begin{aligned} Var\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right] &= E\left[\left(\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}\right)^2\right] E[W^2_{h_{\ell-1},h_\ell}] \\ &- E\left[\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}\right]^2 E[W_{h_{\ell-1},h_\ell}]^2 \end{aligned} \tag{64}$$

Inserting the result for the expectation $E\left[\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}\right]$ and Initializing again $E[W_{h_{\ell-1},h_\ell}] = 1/2$.

$$\begin{aligned} Var\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right] &\approx E\left[\left(\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}\right)^2\right] E[W^2_{h_{\ell-1},h_\ell}] \\ &- \left(\frac{1}{2}\right)^{2\cdot(H_{\ell-1}-1)} \left(\frac{1}{2}\right)^2 \\ &= E\left[\left(\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}\right)^2\right] E[W^2_{h_{\ell-1},h_\ell}] \\ &- \left(\frac{1}{2}\right)^{2\cdot H_{\ell-1}} \end{aligned} \tag{65}$$

Using the identity that $E[W^2_{h_{\ell-1},h_\ell}] = Var[W_{h_{\ell-1},h_\ell}] + E[W_{h_{\ell-1},h_\ell}]^2$, and again using $E[W_{h_{\ell-1},h_\ell}] = 1/2$.

$$\begin{aligned} Var\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right] &\approx E\left[\left(\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}\right)^2\right] \left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right) \\ &- \left(\frac{1}{2}\right)^{2\cdot H_{\ell-1}} \end{aligned} \tag{66}$$

To derive $E\left[\left(\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}\right)^2\right]$ the result for $Var[z_{h_\ell}]$ can be used, but for $\hat{H}_{\ell-1} = H_{\ell-1} - 1$, because there is one less term. Inserting $E\left[\left(\frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}}\right)^2\right] = \left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right)^{H_{\ell-1}-1} \left(Var[z_{h_{\ell-1}}] + 1\right)^{H_{\ell-1}-1}$, we have:

$$\begin{aligned} Var\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right] &\approx \left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right)^{H_{\ell-1}-1} \left(Var[z_{h_{\ell-1}}] + 1\right)^{H_{\ell-1}-1} \\ &\cdot \left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right) - \left(\frac{1}{2}\right)^{2\cdot H_{\ell-1}} \\ &= \left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right)^{H_{\ell-1}} \left(Var[z_{h_{\ell-1}}] + 1\right)^{H_{\ell-1}-1} - \left(\frac{1}{2}\right)^{2\cdot H_{\ell-1}} \end{aligned} \tag{67}$$

Inserting the result for $Var\left[\frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}}\right]$ into the result for $Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right]$:

$$\begin{aligned} Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &\approx Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] H_\ell \Bigg(\left(\frac{1}{2}\right)^{2\cdot H_{\ell-1}} \\ &+ \left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right)^{H_{\ell-1}} \left(Var[z_{h_{\ell-1}}] + 1\right)^{H_{\ell-1}-1} - \left(\frac{1}{2}\right)^{2\cdot H_{\ell-1}} \Bigg) \\ &= Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] H_\ell \\ &\cdot \left(\left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right)^{H_{\ell-1}} \left(Var[z_{h_{\ell-1}}] + 1\right)^{H_{\ell-1}-1}\right) \end{aligned} \tag{68}$$

B.4.3 INITIALIZATION

The $W_{h_{\ell-1},h_\ell}$ should be initialized with $E[W_{h_{\ell-1},h_\ell}] = \frac{1}{2}$, in order to not bias towards inclusion or exclusion of $z_{h_{\ell-1}}$. Using the derived variance approximations (68), the variance should be according to the forward pass:

$$Var[W_{h_{\ell-1},h_\ell}] = \left((1 + Var[z_{h_\ell}])^{-H_{\ell-1}} Var[z_{h_\ell}] + (4 + 4Var[z_{h_\ell}])^{-H_{\ell-1}}\right)^{\frac{1}{H_{\ell-1}}} - \frac{1}{4} \tag{69}$$

And according to the backward pass it should be:

$$Var[W_{h_{\ell-1},h_\ell}] = \left(\frac{(Var[z_{h_\ell}] + 1)^{1-H_{\ell-1}}}{H_\ell}\right)^{\frac{1}{H_{\ell-1}}} - \frac{1}{4} \tag{70}$$

Both criteria are dependent on the input variance. If the input variance is know then optimal initialization is possible. However, as this is often not the case one can perhaps assume that $Var[z_{h_{\ell-1}}] = 1$. This is not an unreasonable assumption in many cases, as there may either be a normalization layer somewhere or the input is normalized. If unit variance is assumed, the variance for the forward pass becomes:

$$Var[W_{h_{\ell-1},h_\ell}] = \left(2^{-H_{\ell-1}} + 8^{-H_{\ell-1}}\right)^{\frac{1}{H_{\ell-1}}} - \frac{1}{4} = \frac{1}{8}\left(\left(4^{H_{\ell-1}} + 1\right)^{H_{\ell-1}} - 2\right) \tag{71}$$

And from the backward pass:

$$Var[W_{h_{\ell-1},h_\ell}] = \left(\frac{2^{1-H_{\ell-1}}}{H_\ell}\right)^{\frac{1}{H_{\ell-1}}} - \frac{1}{4} \tag{72}$$

The variance requirement for both the forward and backward pass can be satisfied with $Var[W_{h_{\ell-1},h_\ell}] = \frac{1}{4}$ for a large $H_{\ell-1}$.

C ARITHMETIC TASK

Our "arithmetic task" is identical to the "simple function task" in the NALU paper (Trask et al., 2018). However, as they do not describe their dataset generation, dataset parameters, and model evaluation in details we elaborate on that here.

The aim of the "Arithmetic task" is to directly test arithmetic models ability to extrapolate beyond the training range. Additionally, our generalized version provides a high degree of flexibility in how the input is shaped, sampled, and the problem complexity.

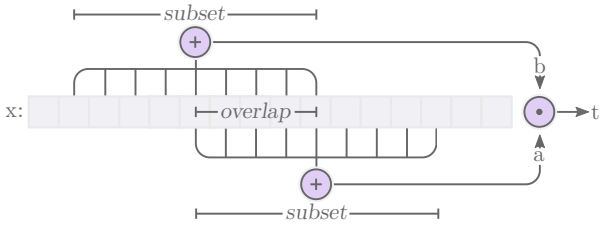


Figure 6: Shows how the dataset is parameterized.

C.1 DATASET GENERATION

The goal is to sum two random subsets of a vector $\mathbf{x}$ (a and b), and perform an arithmetic operation on these ($a \circ b$).

$$a = \sum_{i=s_{1,\text{start}}}^{s_{1,\text{end}}} x_i, \quad b = \sum_{i=s_{2,\text{start}}}^{s_{2,\text{end}}} x_i, \quad t = a \circ b \tag{73}$$

Algorithm 1 defines the exact procedure to generate the data, where an interpolation range will be used for training and validation and an extrapolation range will be used for testing. Default values are defined in table 3.

Table 3: Default dataset parameters for "Arithmetic task"

Parameter name	Default value
Input size	100
Subset ratio	0.25
Overlap ratio	0.5

Parameter name	Default value
Interpolation range	$U[1, 2]$
Extrapolation range	$U[2, 6]$

Algorithm 1 Dataset generation algorithm for "Arithmetic task"

1: **function** DATASET(OP$(\cdot,\cdot)$: Operation, i : InputSize, s : SubsetRatio, o : OverlapRatio, R : Range)
2: $\mathbf{x} \leftarrow$ UNIFORM$(R_{lower}, R_{upper}, i)$ ▷ Sample i elements uniformly
3: $k \leftarrow$ UNIFORM$(0, 1 - 2s - o)$ ▷ Sample offset
4: $a \leftarrow$ SUM$(\mathbf{x}[ik : i(k+s)])$ ▷ Create sum a from subset
5: $b \leftarrow$ SUM$(\mathbf{x}[i(k+s-o) : i(k+2s-0)])$ ▷ Create sum b from subset
6: $t \leftarrow$ OP(a, b) ▷ Perform operation on a and b
7: **return** x, t

C.2 MODEL DEFINTIONS AND SETUP

Models are defined in table 4 and are all optimized with Adam optimization (Kingma & Ba, 2014) using default parameters, and trained over $5 \cdot 10^6$ iterations. Training takes about 8 hours on a single CPU core(8-Core Intel Xeon E5-2665 2.4GHz). We run 19150 experiments on a HPC cluster.

The training dataset is continuously sampled from the interpolation range where a different seed is used for each experiment, all experiments use a mini-batch size of 128 observations, a fixed validation dataset with $1 \cdot 10^4$ observations sampled from the interpolation range, and a fixed test dataset with $1 \cdot 10^4$ observations sampled from the extrapolation range.

Table 4: Model definitions

Model	Layer 1	Layer 2	$\hat{\lambda}_{\text{sparse}}$	λ_{start}	λ_{end}
NMU	NAU	NMU	10	10^6	$2 \cdot 10^6$
NAU	NAU	NAU	0.01	$5 \cdot 10^3$	$5 \cdot 10^4$
$\text{NAC}_\bullet$	NAC_+	$\text{NAC}_\bullet$	–	–	–
$\text{NAC}_{\bullet,\sigma}$	NAC_+	$\text{NAC}_{\bullet,\sigma}$	–	–	–
$\text{NAC}_{\bullet,\text{NMU}}$	NAC_+	$\text{NAC}_{\bullet,\text{NMU}}$	10	10^6	$2 \cdot 10^6$
NAC_+	NAC_+	NAC_+	–	–	–
NALU	NALU	NALU	–	–	–
Linear	Linear	Linear	–	–	–
ReLU	ReLU	ReLU	–	–	–
ReLU6	ReLU6	ReLU6	–	–	–

C.3 Ablation Study

To validate our model, we perform an ablation on the multiplication problem. Some noteworthy observations:

1. None of the W constraints, such as $\mathcal{R}_{sparse}$ and clamping W to be in $[0, 1]$, are necessary when the hidden size is just 2.
2. Removing the $\mathcal{R}_{sparse}$ causes the NMU to immediately fail for larger hidden sizes.
3. Removing the clamping of W does not cause much difference. This is because $\mathcal{R}_{sparse}$ also constrains W outside of $[0, 1]$. The regularizer used here is $\mathcal{R}_{sparse} = \min(|W|, |1 - W|)$, which is identical to the one used in other experiments in $[0, 1]$, but is also valid outside $[0, 1]$. Doing this gives only a slightly slower convergence. Although, this can not be guaranteed in general, as the regularizer is omitted during the initial optimization.
4. Removing both constraints, gives a somewhat satisfying solution, but with a lower success-rate, slower convergence, and higher sparsity error.

In conclusion both constraints are valuable, as they provide faster convergence and a sparser solution, but they are not critical to the success-rate of the NMU.

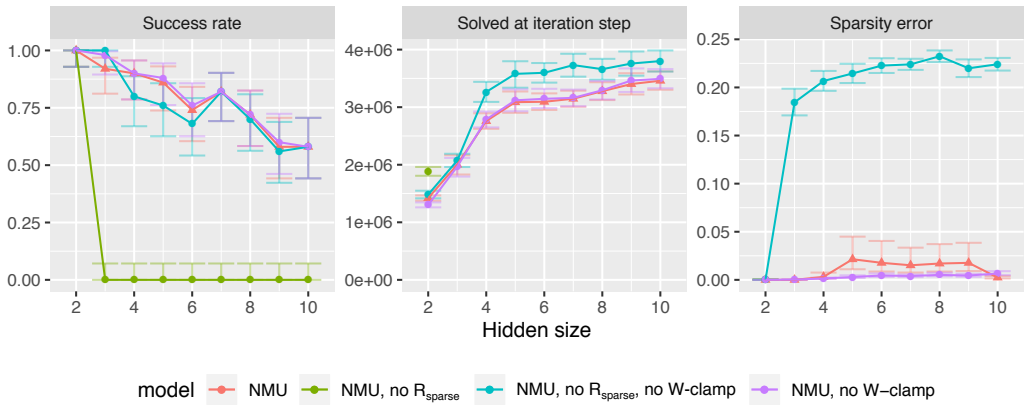


Figure 7: Ablation study where $\mathcal{R}_{sparse}$ is removed and the clamping of W is removed. There are 50 experiments with different seeds, for each configuration.

C.4 EFFECT OF DATASET PARAMETER

To stress test the models on the multiplication task, we vary the dataset parameters one at a time while keeping the others at their default value (default values in table 3). Each runs for 50 experiments with different seeds. The results, are visualized in figure 8.

In figure 3, the interpolation-range is changed, therefore the extrapolation-range needs to be changed such it doesn't overlap. For each interpolation-range the following extrapolation-range is used: $\mathrm{U}[-2,-1]$ uses $\mathrm{U}[-6,-2]$, $\mathrm{U}[-2,2]$ uses $\mathrm{U}[-6,-2] \cup \mathrm{U}[2,6]$, $\mathrm{U}[0,1]$ uses $\mathrm{U}[1,5]$, $\mathrm{U}[0.1,0.2]$ uses $\mathrm{U}[0.2,2]$, $\mathrm{U}[1.1,1.2]$ uses $\mathrm{U}[1.2,6]$, $\mathrm{U}[1,2]$ uses $\mathrm{U}[2,6]$, $\mathrm{U}[10,20]$ uses $\mathrm{U}[20,40]$.

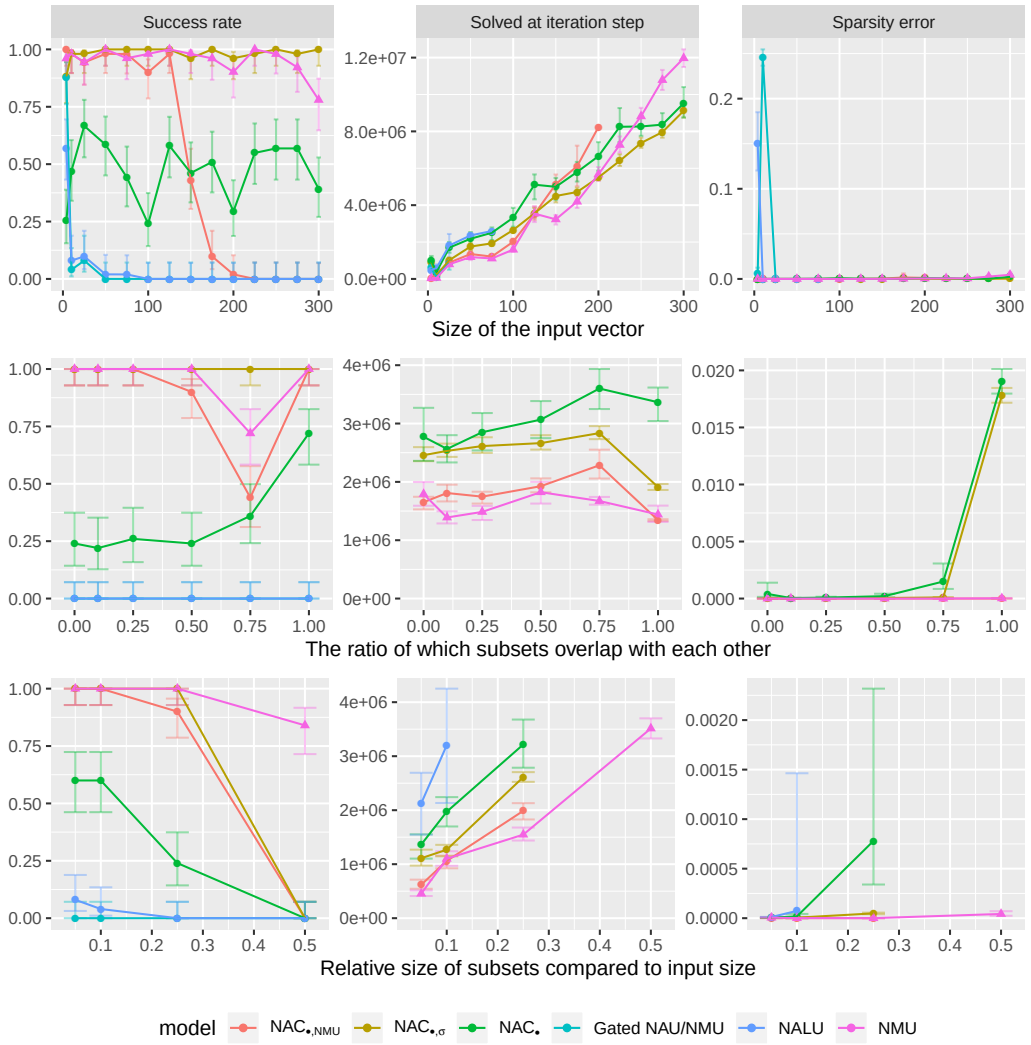


Figure 8: Shows the effect of the dataset parameters.

C.5 GATING CONVERGENCE EXPERIMENT

In the interest of adding some understand of what goes wrong in the NALU gate, and the shared weight choice that NALU employs to remedy this, we introduce the following experiment.

We train two models to fit the arithmetic task. Both uses the NAC_+ in the first layer and NALU in the second layer. The only difference is that one model shares the weight between NAC_+ and $\mathrm{NAC}_\bullet$ in the NALU, and the other treat them as two separate units with separate weights. In both cases NALU should gate between NAC_+ and $\mathrm{NAC}_\bullet$ and choose the appropriate operation. Note that this NALU

model is different from the one presented elsewhere in this paper, including the original NALU paper (Trask et al., 2018). The typical NALU model is just two NALU layers with shared weights.

Furthermore, we also introduce a new gated unit that simply gates between our proposed NMU and NAU, using the same sigmoid gating-mechanism as in the NALU. This combination is done with seperate weights, as NMU and NAU uses different weight constrains and can therefore not be shared.

The models are trained and evaluated over 100 different seeds on the multiplication and addition task. A histogram of the gate-value for all seeds is presented in figure 9 and table 5 contains a summary. Some noteworthy observations:

1. When the NALU weights are separated far more trials converge to select NAC_+ for both the addition and multiplication task. Sharing the weights between NAC_+ and $\mathrm{NAC}_\bullet$ makes the gating less likely to converge for addition.
2. The performance of the addition task is dependent on NALU selecting the right operation. In the multiplication task, when the right gate is selected, $\mathrm{NAC}_\bullet$ do not converge consistently, unlike our NMU that converges more consistently.
3. Which operation the gate converges to appears to be mostly random and independent of the task. This issues caused by the sigmoid gating-mechanism and thus exists independent of the used sub-units.

These observations validates that the NALU gating-mechanism does not converge as intended. This becomes a critical issues when more gates are present, as is normally the case. E.g. when stacking multiple NALU layers together.

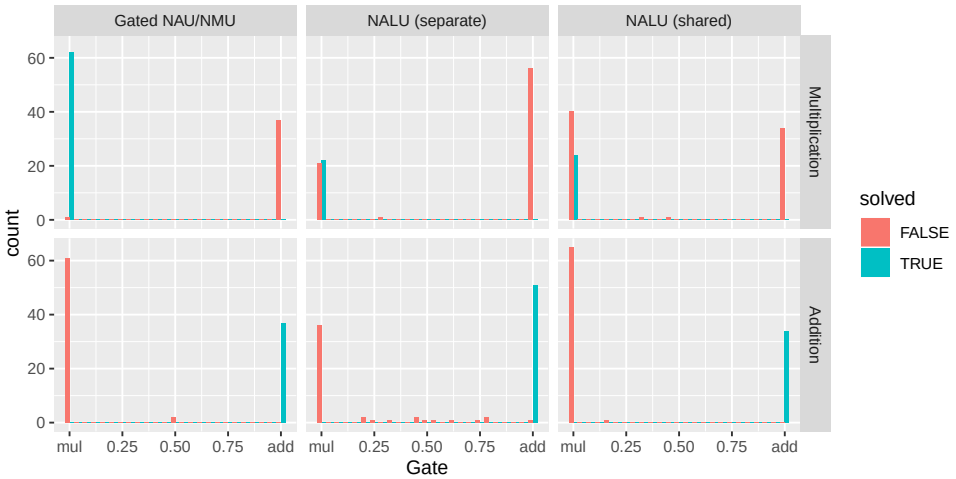


Figure 9: Shows the gating-value in the NALU layer and a variant that uses NAU/NMU instead of NAC_+/$\mathrm{NAC}_\bullet$. Separate/shared refers to the weights in NAC_+/$\mathrm{NAC}_\bullet$ used in NALU.

Table 5: Comparison of the success-rate, when the model converged, and the sparsity error, with 95% confidence interval on the "arithmetic datasets" task. Each value is a summary of 100 different seeds.

Op	Model	Success	Solved at		Sparsity error
		Rate	Median	Mean	Mean
$\times$	Gated NAU/NMU	$\mathbf{62\%}\ ^{+9\%}_{-10\%}$	$\mathbf{1.5 \cdot 10^6}$	$\mathbf{1.5 \cdot 10^6}\ ^{+3.9 \cdot 10^4}_{-3.8 \cdot 10^4}$	$\mathbf{5.0 \cdot 10^{-5}}\ ^{+2.3 \cdot 10^{-5}}_{-1.8 \cdot 10^{-5}}$
	NALU (separate)	$22\%\ ^{+9\%}_{-7\%}$	$2.8 \cdot 10^6$	$3.3 \cdot 10^6\ ^{+3.9 \cdot 10^5}_{-3.6 \cdot 10^5}$	$5.8 \cdot 10^{-2}\ ^{+4.1 \cdot 10^{-2}}_{-2.3 \cdot 10^{-2}}$
	NALU (shared)	$24\%\ ^{+9\%}_{-7\%}$	$2.9 \cdot 10^6$	$3.3 \cdot 10^6\ ^{+3.7 \cdot 10^5}_{-3.6 \cdot 10^5}$	$1.0 \cdot 10^{-3}\ ^{+1.1 \cdot 10^{-3}}_{-4.5 \cdot 10^{-4}}$
$+$	Gated NAU/NMU	$37\%\ ^{+10\%}_{-9\%}$	$\mathbf{1.9 \cdot 10^4}$	$4.2 \cdot 10^5\ ^{+7.3 \cdot 10^4}_{-6.7 \cdot 10^4}$	$\mathbf{1.7 \cdot 10^{-1}}\ ^{+4.6 \cdot 10^{-2}}_{-4.0 \cdot 10^{-2}}$
	NALU (separate)	$\mathbf{51\%}\ ^{+10\%}_{-10\%}$	$1.4 \cdot 10^5$	$\mathbf{2.9 \cdot 10^5}\ ^{+3.5 \cdot 10^4}_{-4.3 \cdot 10^4}$	$1.8 \cdot 10^{-1}\ ^{+1.4 \cdot 10^{-2}}_{-1.4 \cdot 10^{-2}}$
	NALU (shared)	$34\%\ ^{+10\%}_{-9\%}$	$1.8 \cdot 10^5$	$3.1 \cdot 10^5\ ^{+4.3 \cdot 10^4}_{-5.4 \cdot 10^4}$	$1.8 \cdot 10^{-1}\ ^{+2.3 \cdot 10^{-2}}_{-2.1 \cdot 10^{-2}}$

C.6 REGULARIZATION

The λ_{start} and λ_{end} are simply selected based on how much time it takes for the model to converge. The sparsity regularizer should not be used during early optimization as this part of the optimization is exploratory and concerns finding the right solution by getting each weight on the right side of ± 0.5.

In figure 10, 11 and 12 the scaling factor $\hat{\lambda}_{\text{sparse}}$ is optimized.

$$\lambda_{\text{sparse}} = \hat{\lambda}_{\text{sparse}} \max(\min(\frac{t - \lambda_{\text{start}}}{\lambda_{\text{end}} - \lambda_{\text{start}}}, 1), 0) \tag{74}$$

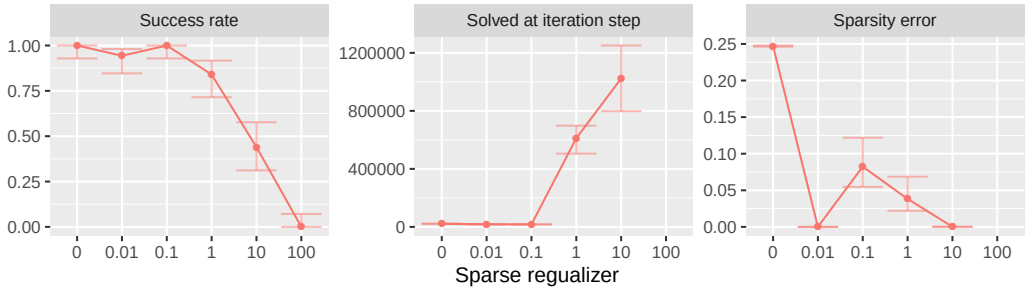


Figure 10: Shows effect of $\hat{\lambda}_{\text{sparse}}$ in NAU on the arithmetic dataset for the $+$ operation.

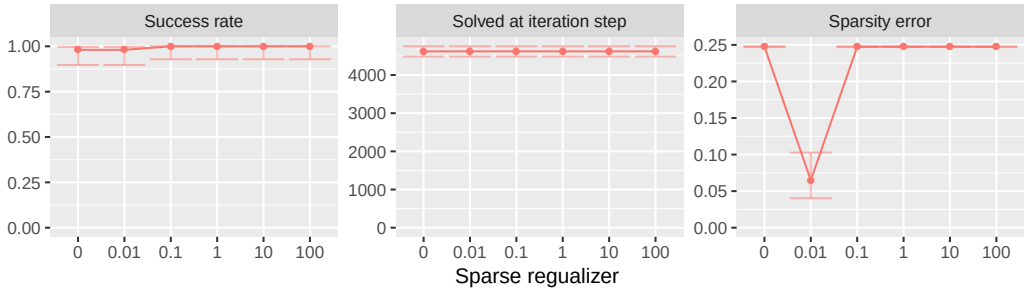


Figure 11: Shows effect of $\hat{\lambda}_{\text{sparse}}$ in NAU on the arithmetic dataset for the $-$ operation.

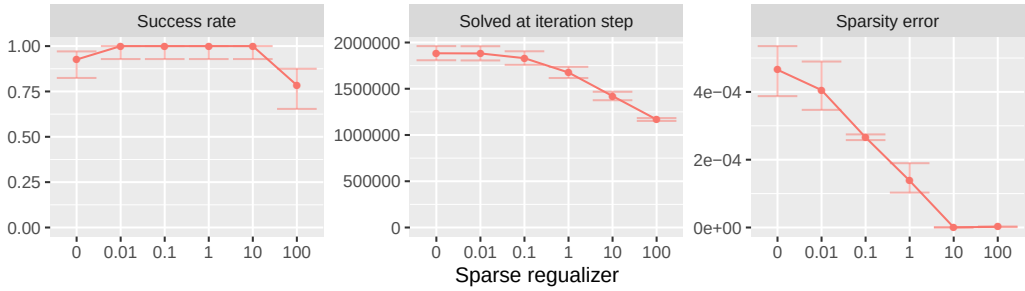


Figure 12: Shows effect of $\hat{\lambda}_{\text{sparse}}$ in NMU on the arithmetic dataset for the $\times$ operation.

C.7 COMPARING ALL MODELS

Table 6 compares all models on all operations used in NALU (Trask et al., 2018). All variations of models and operations are trained for 100 different seeds to build confidence intervals. Some noteworthy observations are:

1. Division does not work for any model, including the $\mathrm{NAC}_\bullet$ and NALU models. This may seem surprising but is actually in line with the results from the NALU paper (Trask et al. (2018), table 1) where there is a large error given the interpolation range. The extrapolation range has a smaller error, but this is an artifact of their evaluation method where they normalize with a random baseline. Since a random baseline with have a higher error for the extrapolation range, a similar error will appear to be smaller. A correct solution to division should have both a small interpolation and extrapolation error.
2. $\mathrm{NAC}_\bullet$ and NALU are barely able to learn $\sqrt{z}$, with just 2% success-rate for NALU and 7% success-rate for $\mathrm{NAC}_\bullet$.
3. NMU is fully capable of learning z^2. It learn this by learning the same subset twice in the NAU layer, this is also how $\mathrm{NAC}_\bullet$ learn z^2.
4. The Gated NAU/NMU (discussed in section C.5) works very poorly, because the NMU initialization assumes that $E[z_{h_{\ell-1}}] = 0$. This is usually true, as discussed in section 2.6, but not in this case for the first layer. In the recommended NMU model, the NMU layer appears after NAU, which causes that assumption to be satisfied.

Table 6: Comparison of the success-rate, when the model converged, and the sparsity error, with 95% confidence interval on the "arithmetic datasets" task. Each value is a summary of 100 different seeds.

Op	Model	Success	Solved at		Sparsity error
		Rate	Median	Mean	Mean
	$\mathrm{NAC}_{\bullet,\mathrm{NMU}}$	$93\% \,^{+4\%}_{-7\%}$	$1.8 \cdot 10^6$	$2.0 \cdot 10^6 \,^{+1.0\cdot10^5}_{-9.7\cdot10^4}$	$9.5 \cdot 10^{-7} \,^{+4.2\cdot10^{-7}}_{-4.2\cdot10^{-7}}$
	$\mathrm{NAC}_{\bullet,\sigma}$	$\mathbf{100}\% \,^{+0\%}_{-4\%}$	$2.5 \cdot 10^6$	$2.6 \cdot 10^6 \,^{+8.8\cdot10^4}_{-7.2\cdot10^4}$	$4.6 \cdot 10^{-5} \,^{+5.0\cdot10^{-6}}_{-5.6\cdot10^{-6}}$
	$\mathrm{NAC}_\bullet$	$31\% \,^{+10\%}_{-8\%}$	$2.8 \cdot 10^6$	$3.0 \cdot 10^6 \,^{+2.9\cdot10^5}_{-2.4\cdot10^5}$	$5.8 \cdot 10^{-4} \,^{+4.8\cdot10^{-4}}_{-2.6\cdot10^{-4}}$
	NAC_+	$0\% \,^{+4\%}_{-0\%}$	—	—	—
	Gated $^{\mathrm{NAU}}_{\mathrm{NMU}}$	$0\% \,^{+4\%}_{-0\%}$	—	—	—
	Linear	$0\% \,^{+4\%}_{-0\%}$	—	—	—
$\times$	NALU	$0\% \,^{+4\%}_{-0\%}$	—	—	—
	NAU	$0\% \,^{+4\%}_{-0\%}$	—	—	—
	NMU	$98\% \,^{+1\%}_{-5\%}$	$\mathbf{1.4 \cdot 10^6}$	$\mathbf{1.5 \cdot 10^6} \,^{+5.0\cdot10^4}_{-6.6\cdot10^4}$	$\mathbf{4.2 \cdot 10^{-7}} \,^{+2.9\cdot10^{-8}}_{-2.9\cdot10^{-8}}$
	ReLU	$0\% \,^{+4\%}_{-0\%}$	—	—	—
	ReLU6	$0\% \,^{+4\%}_{-0\%}$	—	—	—
	$\mathrm{NAC}_{\bullet,\mathrm{NMU}}$	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—
	$\mathrm{NAC}_{\bullet,\sigma}$	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—
	$\mathrm{NAC}_\bullet$	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—
	NAC_+	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—
	Gated $^{\mathrm{NAU}}_{\mathrm{NMU}}$	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—
	Linear	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—
/	NALU	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—
	NAU	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—
	NMU	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—
	ReLU	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—
	ReLU6	$\mathbf{0}\% \,^{+4\%}_{-0\%}$	—	—	—

Table 6: Comparison of the success-rate, when the model converged, and the sparsity error, with 95% confidence interval on the "arithmetic datasets" task. Each value is a summary of 100 different seeds. *(continued)*

Op	Model	Success	Solved at		Sparsity error
		Rate	Median	Mean	Mean
$+$	$\mathrm{NAC}_{\bullet,\mathrm{NMU}}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	$\mathrm{NAC}_{\bullet,\sigma}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	$\mathrm{NAC}_{\bullet}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NAC_{+}	$\mathbf{100}\%\ ^{+0\%}_{-4\%}$	$2.5 \cdot 10^{5}$	$4.9 \cdot 10^{5}\ ^{+5.2\cdot 10^{4}}_{-4.5\cdot 10^{4}}$	$2.3 \cdot 10^{-1}\ ^{+6.5\cdot 10^{-3}}_{-6.5\cdot 10^{-3}}$
	Gated $^{\mathrm{NAU}}_{\mathrm{NMU}}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	Linear	$\mathbf{100}\%\ ^{+0\%}_{-4\%}$	$6.1 \cdot 10^{4}$	$\mathbf{6.3 \cdot 10^{4}}\ ^{+2.5\cdot 10^{3}}_{-3.3\cdot 10^{3}}$	$2.5 \cdot 10^{-1}\ ^{+3.6\cdot 10^{-4}}_{-3.6\cdot 10^{-4}}$
	NALU	$14\%\ ^{+8\%}_{-5\%}$	$1.5 \cdot 10^{6}$	$1.6 \cdot 10^{6}\ ^{+3.8\cdot 10^{5}}_{-3.3\cdot 10^{5}}$	$1.7 \cdot 10^{-1}\ ^{+2.7\cdot 10^{-2}}_{-2.5\cdot 10^{-2}}$
	NAU	$\mathbf{100}\%\ ^{+0\%}_{-4\%}$	$\mathbf{1.8 \cdot 10^{4}}$	$3.9 \cdot 10^{5}\ ^{+4.5\cdot 10^{4}}_{-3.7\cdot 10^{4}}$	$\mathbf{3.2 \cdot 10^{-5}}\ ^{+1.3\cdot 10^{-5}}_{-1.3\cdot 10^{-5}}$
	NMU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	ReLU	$62\%\ ^{+9\%}_{-10\%}$	$6.2 \cdot 10^{4}$	$7.6 \cdot 10^{4}\ ^{+8.3\cdot 10^{3}}_{-7.0\cdot 10^{3}}$	$2.5 \cdot 10^{-1}\ ^{+2.4\cdot 10^{-3}}_{-2.4\cdot 10^{-3}}$
	ReLU6	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
$-$	$\mathrm{NAC}_{\bullet,\mathrm{NMU}}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	$\mathrm{NAC}_{\bullet,\sigma}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	$\mathrm{NAC}_{\bullet}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NAC_{+}	$\mathbf{100}\%\ ^{+0\%}_{-4\%}$	$9.0 \cdot 10^{3}$	$3.7 \cdot 10^{5}\ ^{+3.8\cdot 10^{4}}_{-3.8\cdot 10^{4}}$	$2.3 \cdot 10^{-1}\ ^{+5.4\cdot 10^{-3}}_{-5.4\cdot 10^{-3}}$
	Gated $^{\mathrm{NAU}}_{\mathrm{NMU}}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	Linear	$7\%\ ^{+7\%}_{-4\%}$	$3.3 \cdot 10^{6}$	$1.4 \cdot 10^{6}\ ^{+7.0\cdot 10^{5}}_{-6.1\cdot 10^{5}}$	$1.8 \cdot 10^{-1}\ ^{+7.2\cdot 10^{-2}}_{-5.8\cdot 10^{-2}}$
	NALU	$14\%\ ^{+8\%}_{-5\%}$	$1.9 \cdot 10^{6}$	$1.9 \cdot 10^{6}\ ^{+4.4\cdot 10^{5}}_{-4.5\cdot 10^{5}}$	$2.1 \cdot 10^{-1}\ ^{+2.2\cdot 10^{-2}}_{-2.2\cdot 10^{-2}}$
	NAU	$\mathbf{100}\%\ ^{+0\%}_{-4\%}$	$\mathbf{5.0 \cdot 10^{3}}$	$\mathbf{1.6 \cdot 10^{5}}\ ^{+1.7\cdot 10^{4}}_{-1.6\cdot 10^{4}}$	$6.6 \cdot 10^{-2}\ ^{+2.5\cdot 10^{-2}}_{-1.9\cdot 10^{-2}}$
	NMU	$56\%\ ^{+9\%}_{-10\%}$	$1.0 \cdot 10^{6}$	$1.0 \cdot 10^{6}\ ^{+5.8\cdot 10^{2}}_{-5.8\cdot 10^{2}}$	$\mathbf{3.4 \cdot 10^{-4}}\ ^{+3.2\cdot 10^{-5}}_{-2.6\cdot 10^{-5}}$
	ReLU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	ReLU6	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
$\sqrt{z}$	$\mathrm{NAC}_{\bullet,\mathrm{NMU}}$	$3\%\ ^{+5\%}_{-2\%}$	$1.0 \cdot 10^{6}$	$\mathbf{1.0 \cdot 10^{6}}\ ^{+NaN\cdot 10^{-Inf}}_{-NaN\cdot 10^{-Inf}}$	$\mathbf{1.7 \cdot 10^{-1}}\ ^{+8.3\cdot 10^{-3}}_{-8.1\cdot 10^{-3}}$
	$\mathrm{NAC}_{\bullet,\sigma}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	$\mathrm{NAC}_{\bullet}$	$\mathbf{7}\%\ ^{+7\%}_{-4\%}$	$\mathbf{4.0 \cdot 10^{5}}$	$1.5 \cdot 10^{6}\ ^{+6.0\cdot 10^{5}}_{-5.6\cdot 10^{5}}$	$2.4 \cdot 10^{-1}\ ^{+1.7\cdot 10^{-2}}_{-1.7\cdot 10^{-2}}$
	NAC_{+}	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	Gated $^{\mathrm{NAU}}_{\mathrm{NMU}}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	Linear	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NALU	$2\%\ ^{+5\%}_{-1\%}$	$2.6 \cdot 10^{6}$	$3.3 \cdot 10^{6}\ ^{+1.8\cdot 10^{6}}_{-2.2\cdot 10^{6}}$	$5.0 \cdot 10^{-1}\ ^{+2.5\cdot 10^{-6}}_{-8.0\cdot 10^{-6}}$
	NAU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NMU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	ReLU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	ReLU6	$0\%\ ^{+4\%}_{-0\%}$	—	—	—

Table 6: Comparison of the success-rate, when the model converged, and the sparsity error, with 95% confidence interval on the "arithmetic datasets" task. Each value is a summary of 100 different seeds. *(continued)*

Op	Model	Success	Solved at		Sparsity error
		Rate	Median	Mean	Mean
z^2	$\mathrm{NAC}_{\bullet,\mathrm{NMU}}$	$\mathbf{100\%}\ ^{+0\%}_{-4\%}$	$1.4 \cdot 10^6$	$1.5 \cdot 10^6\ ^{+8.4\cdot10^4}_{-7.9\cdot10^4}$	$\mathbf{2.9 \cdot 10^{-7}}\ ^{+1.4\cdot10^{-8}}_{-1.4\cdot10^{-8}}$
	$\mathrm{NAC}_{\bullet,\sigma}$	$\mathbf{100\%}\ ^{+0\%}_{-4\%}$	$1.9 \cdot 10^6$	$1.9 \cdot 10^6\ ^{+5.3\cdot10^4}_{-6.2\cdot10^4}$	$1.8 \cdot 10^{-2}\ ^{+4.3\cdot10^{-4}}_{-4.3\cdot10^{-4}}$
	$\mathrm{NAC}_{\bullet}$	$77\%\ ^{+7\%}_{-9\%}$	$3.3 \cdot 10^6$	$3.2 \cdot 10^6\ ^{+1.6\cdot10^5}_{-2.0\cdot10^5}$	$1.8 \cdot 10^{-2}\ ^{+5.8\cdot10^{-4}}_{-5.7\cdot10^{-4}}$
	NAC_{+}	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	Gated $^{\mathrm{NAU}}_{\mathrm{NMU}}$	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	Linear	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NALU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NAU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NMU	$\mathbf{100\%}\ ^{+0\%}_{-4\%}$	$\mathbf{1.2 \cdot 10^6}$	$\mathbf{1.3 \cdot 10^6}\ ^{+3.1\cdot10^4}_{-3.6\cdot10^4}$	$3.7 \cdot 10^{-5}\ ^{+5.4\cdot10^{-5}}_{-3.7\cdot10^{-5}}$
	ReLU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	ReLU6	$0\%\ ^{+4\%}_{-0\%}$	—	—	—

D SEQUENTIAL MNIST

D.1 TASK AND EVALUATION CRITERIA

The simple function task is a purely synthetic task, that does not require a deep network. As such it does not test if an arithmetic layer inhibits the networks ability to be optimized using gradient decent.

The sequential MNIST task takes the numerical value of a sequence of MNIST digits and applies a binary operation recursively. Such that $t_i = Op(t_{i-1}, z_t)$, where z_t is the MNIST digit's numerical value.

The performance of this task depends on the quality of the image-to-scalar network and the arithmetic layer's ability to model the scalar. We use mean-square-error (MSE) to evaluate joint image-to-scalar and arithmetic layer model performance. To determine an MSE threshold from the correct prediction we use an empirical baseline. This is done by letting the arithmetic layer be solved, such that only the image-to-scalar is learned. By learning this over multiple seeds an upper bound for an MSE threshold can be set. In our experiment we use the 1% one-sided upper confidence-interval, assuming a student-t distribution.

Similar to the simple function task we use a success-criteria as reporting the MSE is not interpretable and models that do not converge will obscure the mean. Furthermore, because the operation is applied recursively, natural error from the dataset will accumulate over time, thus exponentially increasing the MSE. Using a baseline model and reporting the successfulness solves this interpretation challenge.

D.2 ADDITION OF SEQUENTIAL MNIST

Figure shows results for sequential addition of MNIST digits. This experiment is identical to the MNIST Digit Addition Test from Trask et al. (2018, section 4.2). The models are trained on a sequence of 10 digits and evaluated on sequences between 1 and 1000 MNIST digits.

Note that the NAU model includes the R_z regularizer, similarly to the "Multiplication of sequential MNIST" experiment in section 4.2. However, because the weights are in $[-1, 1]$, and not $[0, 1]$, and the idendity of addition is 0, and not 1, R_z is

$$\mathcal{R}_z = \frac{1}{H_{\ell-1}H_\ell} \sum_{h_\ell}^{H_\ell} \sum_{h_{\ell-1}}^{H_{\ell-1}} (1 - |W_{h_{\ell-1},h_\ell}|) \cdot \tilde{z}^2_{h_{\ell-1}} \,. \tag{75}$$

To provide a fair comparison, a variant of NAC_+ that also uses this regularizer is included, this variant is called NAC_{+,R_z}. Section D.3 provides an ablation study of the R_z regularizer.

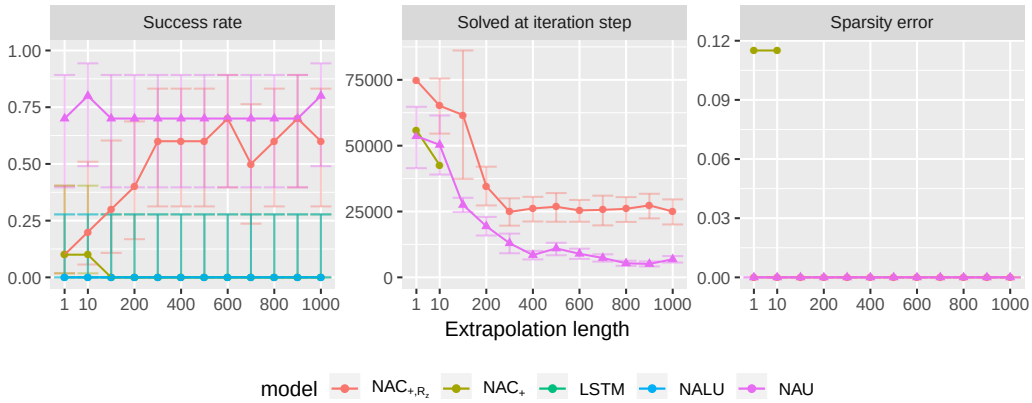


Figure 13: Shows the ability of each model to learn the arithmetic operation of addition and back-propagate through the arithmetic layer in order to learn an image-to-scalar value for MNIST digits. The model is tested by extrapolating to larger sequence lengths than what it has been trained on. The NAU and NAC_{+,R_z} models use the R_z regularizer from section 4.2.

D.3 SEQUENTIAL ADDTION WITHOUT THE R_z REGULARIZER

As an ablation study of the R_z regularizer, figure 14 shows the NAU model without the R_z regularizer. Removing the regularizer causes a reduction in the success-rate. The reduction is likely larger, as compared to sequential multiplication, because the sequence length used for training is longer. The loss function is most sensitive to the 10th output in the sequence, as this has the largest scale. This causes some of the model instances to just learn the mean, which becomes passable for very long sequences, which is why the success-rate increases for longer sequences. However, this is not a valid solution. A well-behavior model should be successful independent of the sequence length.

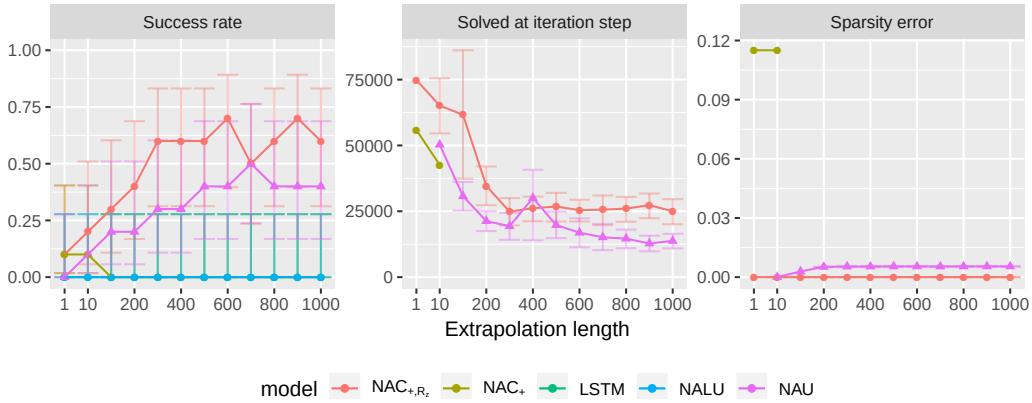


Figure 14: Same as figure 13, but where the NAU model do not use the R_z regularizer.

D.4 SEQUENTIAL MULTIPLICATION WITHOUT THE R_z REGULARIZER

As an ablation study of the R_z regularizer figure 15 shows the NMU and $\mathrm{NAC}_{\bullet,\mathrm{NMU}}$ models without the R_z regularizer. The success-rate is somewhat similar to figure 4. However, as seen in the "sparsity error" plot, the solution is quite different.

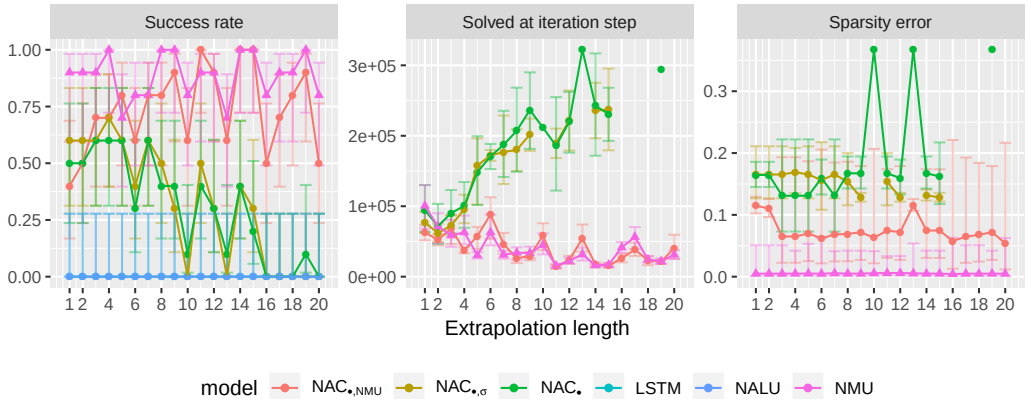


Figure 15: Shows the ability of each model to learn the arithmetic operation of addition and back-propagate through the arithmetic layer in order to learn an image-to-scalar value for MNIST digits. The model is tested by extrapolating to larger sequence lengths than what it has been trained on. The NMU and $\mathrm{NAC}_{\bullet,\mathrm{NMU}}$ models do not use the R_z regularizer.