NEURAL ARITHMETIC UNITS

Andreas Madsen
Computationally Demanding
amwebdk@gmail.com

Alexander Rosenberg Johansen
Technical University of Denmark
aler@dtu.dk

ABSTRACT

Exact arithmetic operations of real numbers present a unique learning challenge for machine learning models. Neural networks can approximate complex functions by learning from labeled data. However, when extrapolating to out-of-distribution samples neural networks often fail. Learning the underlying logic, as opposed to an approximation, is crucial for applications such as comparing, counting, and inferring physical models. Our proposed Neural Addition Unit (NAU) and Neural Multiplication Unit (NMU) can learn the underlying rules of real number addition, subtraction, and multiplication by observing input-output pairs and using backpropagation.The proposed units controls the arithmetic operation by using a sparse weight matrix, which allows the units to perform exact arithmetic operations. Through theoretical analysis, supported by empirical evidence, we justify how the NAU and NMU improve over previous methods. Our experimental setting, motivated by previous work, includes an arithmetic extrapolation task and multiplication of up to 20 MNIST digits. We show that NAU and NMU have fewer parameters, converges more consistently, learns faster, and have more meaningful discrete values than previous attempts.[1]

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 (**???**). 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 (**???**). However, when using neural networks to learn simple arithmetic problems, such as counting, multiplication, or comparison they systematically fail to extrapolate onto unseen ranges (**???**). This can be a significant drawback when comparing in question answering (**?**) or counting objects in visual data (**??**).

In this paper, we analyze and improve parts of the recently proposed Neural Arithmetic Logic Unit (NALU) (**?**). The NALU is a neural network layer with 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. The parameters, which are computed by a soft weight constraint using a tanh-sigmoid transformation, are learned by observing arithmetic input-output pairs and using backpropagation (**?**).

Our contributions are alternatives to the NAC_+ and $\text{NAC}_\bullet$ units that are more theoretically founded. Our alternatives can support small and negative numbers, are more sparse, and supports a larger hidden size, while using less parameters. We test these properties through a rigid experimental setup with more than 10000 arithmetic tests and recurrent multiplication of up to 20 MNIST digits (**?**).

We motivate our work by an investigation of the NALU components: the soft gating mechanism that binds the subunits; the subunits $\text{NAC}_\bullet$ and NAC_+, and the way parameters are constructed. Our findings are the following: (a) $\text{NAC}_\bullet$ does not work for negative input. (b) $\text{NAC}_\bullet$ cannot model inputs $< \epsilon$. (c) optimal weight initialization for the parameters has a gradient of zero in expectation. (d) the weight design does not enforce sparsity and results suggests that they rarely are, which limits interpretability. (e) $\text{NAC}_\bullet$ has no optimal initialization. (f) The expected mean of $\text{NAC}_\bullet$, at

[1]Implementation is available on GitHub: https://github.com/AndreasMadsen/stable-nalu.

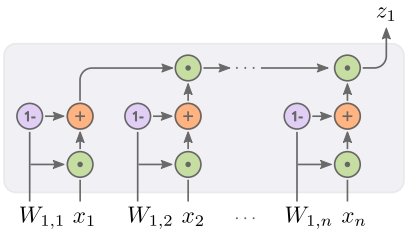


Figure 1: Visualization of NMU for a single output scalar z_1, this construction repeats for every element in the output vector $\mathbf{z}$.

initialization, is exponential w.r.t. the hidden size and has exploding variance. (g) Optimizing NAC$_\bullet$ for division is close to impossible and never converges in practice. (h) The NALU does not converge; we find that learning the gate is cumbersome due to the heterogeneity of the subunits. NAC$_\bullet$ takes orders of magnitude longer to converge than NAC$_+$.

Motivated by these challenges, we attempt to solve (a-f) and leave (g-h) for future work.

Furthermore, we find that the experimental setup of **?** has the following concerns: (i) the dataset parameters for "simple function task" is not defined. (ii) the evaluation metric is based on relative performance to a random baseline. This means that if the random baseline has an MSE of 1e10, then 1e7 would be considered a score of 0.1. (iii) Multiplication is only thoroughly tested in simple function task and not on anything requiring a deep neural network .

We attempt to solve (i-iii) by proposing a much extended arithmetic task, define a successful convergence criteria, and do multiplication of MNIST digits.

(comments on 4.3 in appendix XX)

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 experiemnts for each model, where we attempt to fit this function. Table 1 show that NMU has a higher success rate and converges faster.

Table 1: Shows the success-rate, at what global step the model converged at, and the sparsity error for all weight matrices, with 95% confidence interval. Best result is highlighed.

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 backpropergation.

We develop these layers by analysing the theoretical issues in the Neural Arithmetic Logic Unit (NALU) (**?**).

2.1 INTRODUCING NALU

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

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 CHALLENGES OF GATING BETWEEN NAC_+ AND $\text{NAC}_\bullet$

The sigmoid gating-mechanism in NALU, is problemtic in theory. Its purpose to select either NAC_+ or $\text{NAC}_\bullet$ exclusively, which assumes that the correct sub-unit is selected in first try. As selecting the wrong sub-unit makes it impossible for the correct sub-unit to converge, as it is multiplied by zero.

This assumption rarely holds. The gating-mechanism will converge to select the NAC_+ unit, independently of the desired operation, as this unit is much easier to optimize and will thus be the best estimator early on. NALU works around this issue by sharing the weight matrix between NAC_+ and $\text{NAC}_\bullet$, such that even if NAC_+ is selected the weights can still, by pure chance, converge such that $\text{NAC}_\bullet$ is the best estimator.

To summarize, sharing the weight matrix allows, for a lucky seed, $\text{NAC}_\bullet$ to be selected. However, it also makes NAC_+ much harder to optimize, when addition is the desired operation. We confirm this empirically in appendix **??**, by comparing the NALU model with seperate and shared weights. Solving this problem is a research project on its own, we thus focus on improving the sub-units by assuming that the gating is known.

add appendix

2.2.1 WEIGHT MATRIX CONSTRUCTION

? 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 the context of NALU, this initialization is also 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 show 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 (see table 2) show that equation 1 does not create the desired bias for $\{-1, 0, 1\}$ for the addition and subtraction problem.

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.

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

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

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

2.2.2 Challenges of division

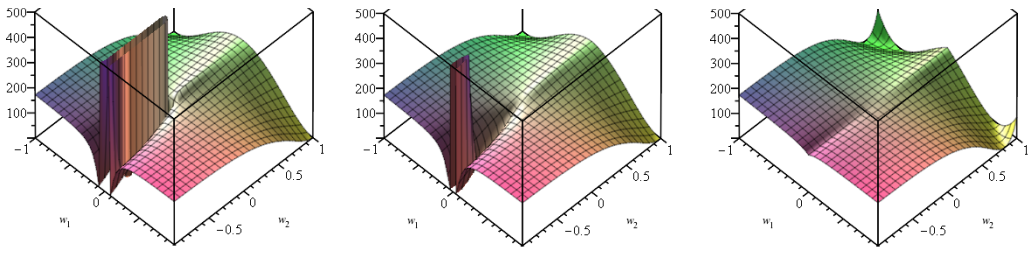
The $\text{NAC}_\bullet$, as formulated in equation 3, has the ability to perform exact division, or more precisely multiplication of the inverse of elements from a vector, when a weight in $W_{h_{\ell-1},h_\ell}$ is -1.

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

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

Equation (8) reveals that if $|z_{h_{\ell-1}}|$ is near zero (which is likely since $E[z_{h_{\ell-1}}] = 0$ is a desired property when initializing (**?**)), $W_{h_{\ell-1},h_\ell}$ 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, is present even when multiplication is the desired operation. Since a multiplication unit should not explode for $z_{h_{\ell-1}}$ near zero, it is likely that supporting division is infeasible. The NALU paper shows the same difficulty, as they are unable to learn division on the interpolation range. Their results on the extrapolation range is an artifact of their evaluation method.



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

Figure 2: RMS loss curvature for a NAC_+ layer followed by either a $\text{NAC}_\bullet$. The weight matrices constrained are to $\mathbf{W}_1 = \left[\begin{smallmatrix} w_1 & w_1 & 0 & 0 \\ w_1 & w_1 & w_1 & w_1 \end{smallmatrix}\right]$, $\mathbf{W}_2 = [w_2 \; w_2]$. 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 desired solution is $w_1 = w_2 = 1$, although this problem have additional undesired unstable solutions.

2.2.3 Initialization of $\text{NAC}_\bullet$

Initialization is important to consider for fast and consistent convergence, one desired property is that weights can be initialized such that $E[z_{h_\ell}] = 0$ (**?**). Using second order Taylor approximation and assuming all $z_{h_{\ell-1}}$ are uncorrelated, the expectation of $\text{NAC}_\bullet$ 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 $\text{NAC}_\bullet$ is likely impossible. The variance can also not be input-independent initialized and is expected to explode (proofs in Appendix B.3).

2.2.4 NEURAL MULTIPLICATION UNIT

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

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

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

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

The NMU is regularized similar to the NAU and has a multiplicative identity when $W_{h_{\ell-1},h_\ell} = 0$. The NMU unit does 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.

2.2.5 MOMENTS AND INITIALIZATION

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

Our proposed 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 units, 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 initialized optimally with $Var[W_{h_{\ell-1},h_\ell}] = \frac{1}{4}$ (proof in Appendix B.4.3).

2.2.6 REGULARIZER SCALING

We use the regularizer scaling as defined in (14). The motivation here, is that the 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, and then a period where the solution is made sparse.

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

3 RELATED WORK

Pure neural models using convolutions, gating, differentiable memory, and/or attention architectures have attempted to learn arithmetic tasks through backpropagation (**???**). Some of the results have close to perfect extrapolation. However, the models are constrained to work only on whole numbers and requires well defined arithmetic setups such as binary representations of numbers for input and output. We do not extend current approximative methods, but instead develop two new units that can learn exact arithmetic operations on real numbers, without requiring a binary representations.

The Neural Arithmetic Expression Calculator (**?**) propose learning real number arithmetic by having neural network subcomponents and repeatedly combine 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 their solution do not generalize much beyond interpolation length.

In our experiments, the NAU is used to do a subset-selection, which is then followed by either a summation or multiplication. An alternative, fully differentiable version, is to use a gumbel-softmax that can perform exact subset-selection (**?**). 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" as shown in (**?**). The goal is to sum two random subsets of a vector and perform a arithmetic operation as defined below

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

where n (default 100), $U[r_{\text{lower}}, r_{\text{upper}}]$ (interpolation default is $U[1, 2]$ and extrapolation default is $U[2, 6]$), the subset size (default 25%), and subset overlap (default 50%) are parameters that we use 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

The goal is to achieve 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, and $\mathbf{W}_1^\epsilon$ is nearly-perfect and $\mathbf{W}_2^*$ is perfect (example in equation 16), the success criteria is $\mathcal{L}_{\mathbf{W}_1,\mathbf{W}_2} < \mathcal{L}_{\mathbf{W}_1^\epsilon,\mathbf{W}_2^*}$, measured on the extrapolation error, for $\epsilon = 10^{-5}$.

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

To measure speed of convergence 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. Only models that managed to solve 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 considered model instances that did solve the task and report the 95% confidence interval.

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

We evaluate each metric every 1000 iterations on the test set that uses the extrapolation range, and choose the best iteration based on the validation dataset that uses the 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. Finally, the NALU model consists of two NALU layers. See explicit definitions 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 converges faster. For addition and subtraction, the NAU model converges faster, given the median, and has a sparser solution. 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 models works better than $NAC_\bullet$ 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 the purpose is to fit an unknown function.

Additionally, the NMU model is unlike the $NAC_\bullet$ model also capable of understanding 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 $\text{U}[-6, -2] \cup \text{U}[2, 6]$.

Table 2: Shows the success-rate, at what global step the model converged at, and the sparsity error for all weight matrices, with 95% confidence interval. Best result is highlighed.

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

Finally, to validate that division and the lack of bias in $NAC_\bullet$ are critical issues but that solving these alone are not enough, two additional models compared with. 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 bias regularization, identically to that in NMU model.

Figure 3 shows that the NMU can both handle a much larger hidden-size, as well as negative inputs, and that solving the division and bias issues alone improves the success rate, but are no enough when the hidden-size is large, as there is no ideal initialization.

4.2 Product of sequential MNIST

To compare how easy it is to backpropergation though the arithmetic layers, the arithmetic layers are applied 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 (**?**)[2], but use multiplication rather than addition.. Each model is trained on sequences of length 2, and then tested on sequences of length 20 MNIST digits.

Success of convergence is determined by comparing the MSE of each model, with a baseline model that directly computes the sequential product. 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 discovered that having an unconstrained "input-network" causes the multiplication-units to learn an undesired solution, such as $(0.1 \cdot 81 + 1 - 0.1) = 9$. 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 $\mathcal{R}_\mathrm{z} = \frac{1}{H_{\ell-1} H_\ell} \sum_{h_\ell}^{H_\ell} \sum_{h_{\ell-1}}^{H_{\ell-1}} (1 - W_{h_{\ell-1}, h_\ell}) \cdot (1 - \bar{z}_{h_{\ell-1}})^2$. This regularizer is applied to the NMU and $\mathrm{NAC}_{\bullet,\mathrm{NMU}}$ models. See appendix D.2 for the results, when this regularizer is not used.

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

[2]Also uses the same CNN, `https://github.com/pytorch/examples/tree/master/mnist`.

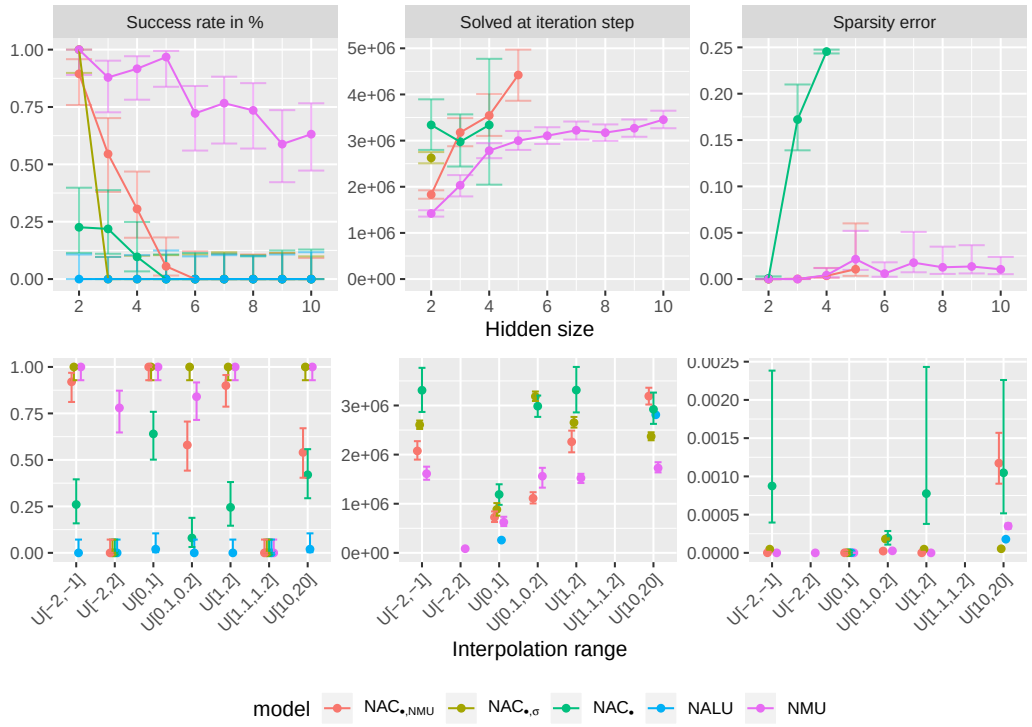


Figure 3: Shows that the NMU can handle a large hidden size, and works when the input contains both positive and negative numbers ($U[-2,-2]$).

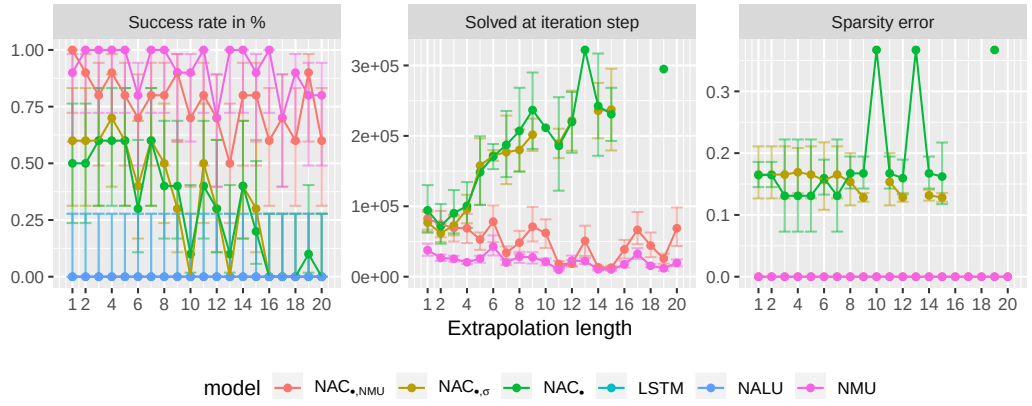


Figure 4: Shows the ability of each model to backpropergation and extrapolate to larger sequence lengths.

5 Conclusion

We have investigated how a function based on constraint weights can extrapolate well on arithmetic operations and if it can learn the weights by stochastic gradient descent. Inspired by the recently proposed Neural Arithmetic Logic Unit (NALU), we propose the Neural Addition Unit (NAU) and Neural Multiplication Unit (NMU), which has more consistent convergence on a range of addition, subtraction, and multiplication tasks than the NALU.4 Beyond convergence capabilities, our modifications makes the NMU capable of extrapolating to numerically small numbers and the negative range, which has previously been impossible. Our analytic analysis highlights that learning division is challenging as division close to zero creates explosions.

ACKNOWLEDGMENTS

We would like to thank Andrew Trask and the other authors of the NALU paper, for highlighting the importance and challenges of extrapolation in Neural Networks. We would also like to thank the students Raja Shan Zaker Kreen and William Frisch Møller from The Technical University of Denmark, who initially showed us that the NALU does not converge consistently.

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 straight forward to derive.

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

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 just a multiplication on $\frac{\partial \mathcal{L}}{\partial W_{h_\ell,h_{\ell-1}}}$.

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

First 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 $W_{h_\ell,h_{\ell-1}}$ is straight forward to derive.

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

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, expect 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, it becomes trivial to derive the gradient 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. (**?**):

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

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 ($Var[W_{h_{\ell-1},h_\ell}] = \frac{2}{H_{\ell-1}+H_\ell}$) (**?**).

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 (**?**). 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 properly unlikely to expect that the expectation 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 does 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, feel free to compute more exact solutions.

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

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

Finally this gives 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 take the second order Taylor approximation of g, around $(E[z_{h_{\ell-1}}], E[W_{h_\ell,h_{\ell-1}}])$.

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

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:

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

This gives the final expectation:

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

As this expectation is of particular interest, 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.

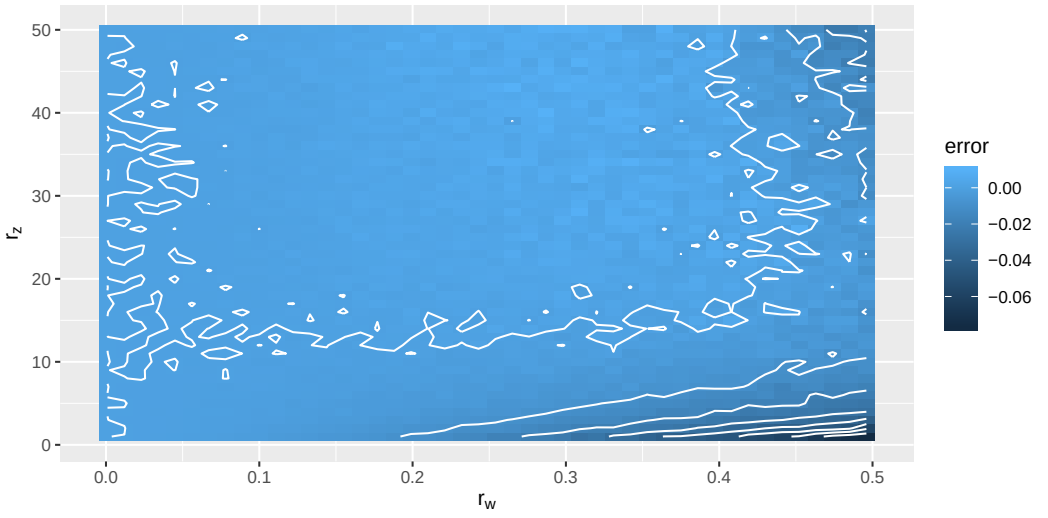


Figure 5: Error between theoretical approximation and the numerical approximation estimated by random sampling of 100000 observations at each combination of r_z and r_w.

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

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

We already have from the expectation result that:

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

By substitution of variable we have that:

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

This gives the variance:

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

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 as:

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

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:

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

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

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

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 again 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 again $E[W_{h_{\ell-1},h_\ell}] = 1/2$, 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 (**?**).

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

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

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

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:

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

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 $E[W_{h_{\ell-1},h_\ell}] = 1/2$.

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

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 $E[W_{h_{\ell-1},h_\ell}] = 1/2$.

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

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:

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

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, 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, one gets from the forward pass:

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

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 (**?**). 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 providing a high degree of flexibility in how the input is shaped, sampled, and the problem complexity.

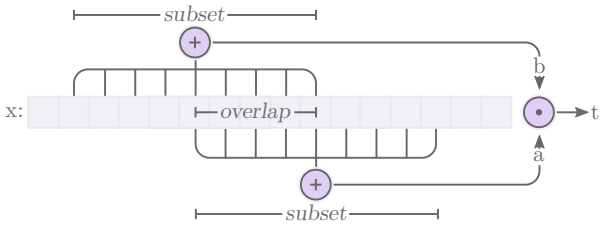


Figure 6: Shows how the dataset is parameterized.

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"

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

C.2 MODEL DEFINTIONS AND SETUP

Models are defined in table 4 and are all optimized with Adam optimization (**?**) using default parameters, and trained over $5 \cdot 10^6$ iterations. Training takes about 6 hours on a single CPU core(8-Core Intel Xeon E5-2665 2.4GHz). We run more than 10000 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.

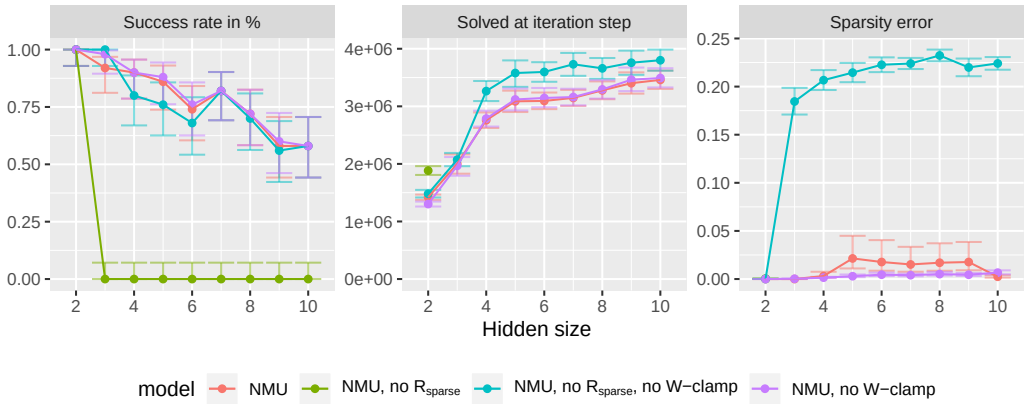


Figure 7: Ablation study where $\mathcal{R}_{sparse}$ is removed and the clamping of W is removed. There are 50 experiments with different seeds, for each configuration.

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.

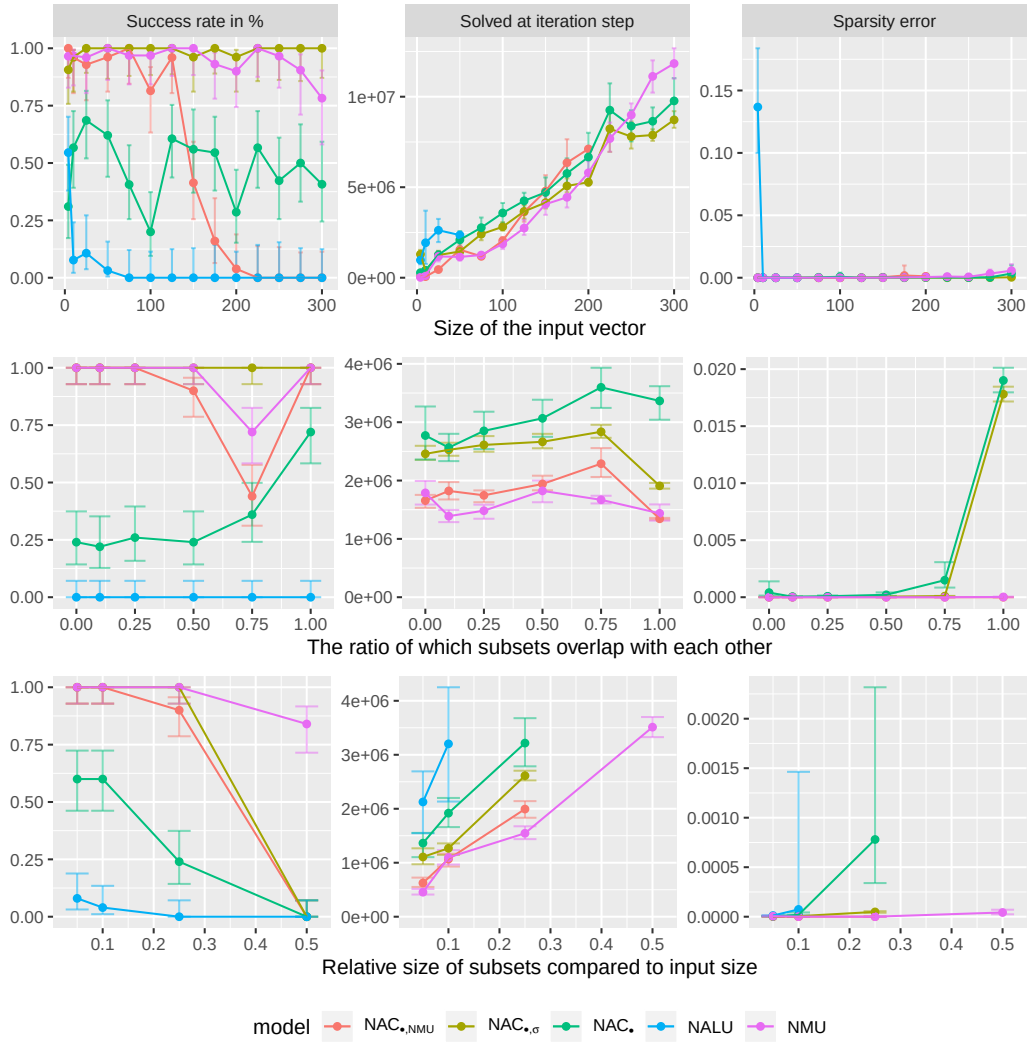


Figure 8: Shows the effect of the dataset parameters.

C.5 NALU GATING EXPERIMENT

In the interrest of adding some understand of what goes wrong in the NALU gate, and the shared weight "hack" that NALU employees to fix 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 just consider them two separate models 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 (**?**). The typical NALU model, is just two NALU layers with shared weights.

These models are trained for 100 different seeds, on the multiplication and addition task. A histogram of the NALU gate is presented in figure 9 and a summary is presented in table 5. Some noteworthy observations:

1. When the weights are separated, far more trials converge to select NAC_+, for both the addition and multiplication task.
2. The performance of the addition task, is strongly correlated with NALU selecting the right operation. In the multiplication task, even if the right gate is selected, NAC_* still struggle to converge consistently, as also seen elsewhere in this paper.
3. Sharing the weights between NAC_+ and $\mathrm{NAC}_\bullet$, makes it harder to learn the addition problem.

These observations validates, that the NALU gate tends to select NAC_+, as NAC_+ is far easier to learn than $\mathrm{NAC}_\bullet$, thus the NAC_+ becomes the best estimator early on which the gate will prefer. Once the gate have selected NAC_+, it is no longer possible to learn $\mathrm{NAC}_\bullet$.

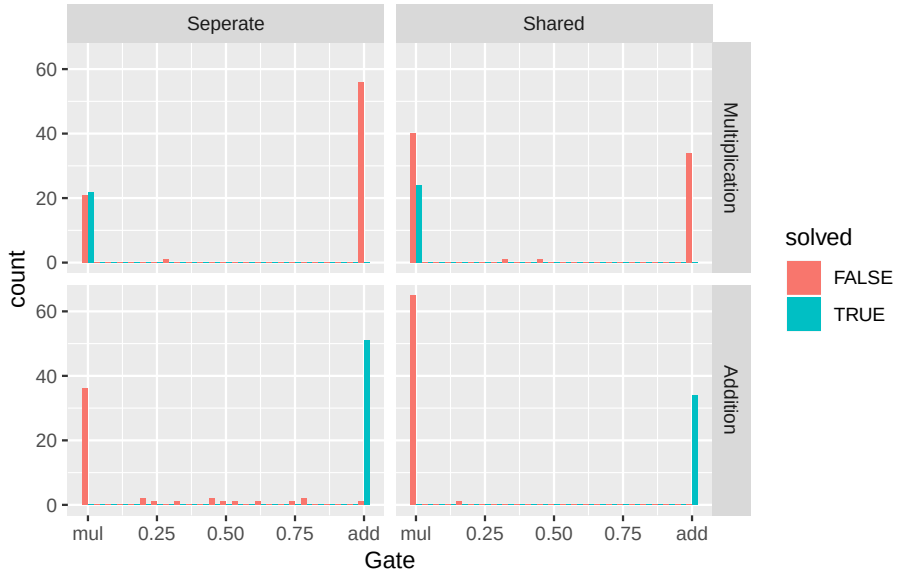


Figure 9: Shows the gating value in the NALU layer followed by a NAC_+ layer, where the weights in NAC_+ and $\mathrm{NAC}_\bullet$ are either shared or separated.

Table 5: Shows the success-rate, when the model converged, and the sparsity error for all weight matrices, with 95% confidence interval. Each value is a summary of 100 different seeds.

Op	Model	Success	Solved at		Sparsity error
		Rate	Median	Mean	Mean
$\times$	NALU (seperate)	$22\%\ ^{+9\%}_{-7\%}$	$\mathbf{2.8 \cdot 10^6}$	$\mathbf{3.3 \cdot 10^6}\ ^{+3.9 \cdot 10^5}_{-3.6 \cdot 10^5}$	$5.8 \cdot 10^{-2}\ ^{+4.1 \cdot 10^{-2}}_{-2.3 \cdot 10^{-2}}$
	NALU (shared)	$\mathbf{24\%}\ ^{+9\%}_{-7\%}$	$2.9 \cdot 10^6$	$3.3 \cdot 10^6\ ^{+3.7 \cdot 10^5}_{-3.6 \cdot 10^5}$	$\mathbf{1.0 \cdot 10^{-3}}\ ^{+1.1 \cdot 10^{-3}}_{-4.5 \cdot 10^{-4}}$
$+$	NALU (seperate)	$\mathbf{51\%}\ ^{+10\%}_{-10\%}$	$\mathbf{1.4 \cdot 10^5}$	$\mathbf{2.9 \cdot 10^5}\ ^{+3.5 \cdot 10^4}_{-4.3 \cdot 10^4}$	$1.8 \cdot 10^{-1}\ ^{+1.4 \cdot 10^{-2}}_{-1.4 \cdot 10^{-2}}$
	NALU (shared)	$34\%\ ^{+10\%}_{-9\%}$	$1.8 \cdot 10^5$	$3.1 \cdot 10^5\ ^{+4.3 \cdot 10^4}_{-5.4 \cdot 10^4}$	$\mathbf{1.8 \cdot 10^{-1}}\ ^{+2.3 \cdot 10^{-2}}_{-2.1 \cdot 10^{-2}}$

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 simply about getting each weight on the right side of ± 0.5.

In these experiments the scaling factor $\hat{\lambda}_{\text{sparse}}$ is optimized. Results are shown in in figure 10, 11, and 12.

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

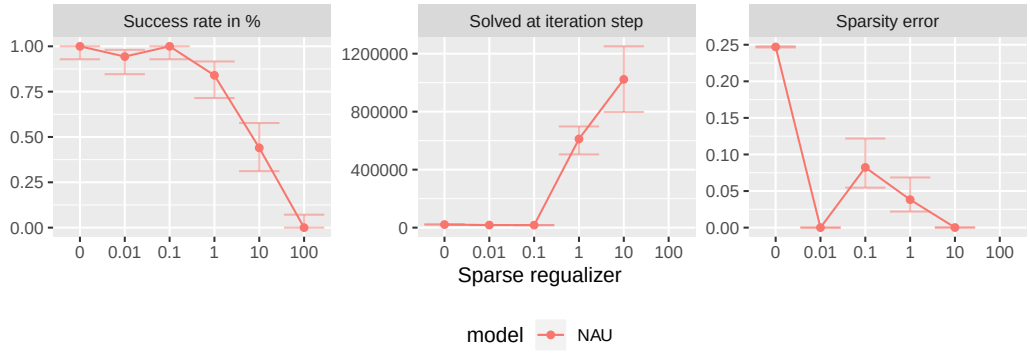


Figure 10: Shows effect of the scaling factor $\hat{\lambda}_{\text{sparse}}$, on the arithmetic dataset for the $+$ operation.

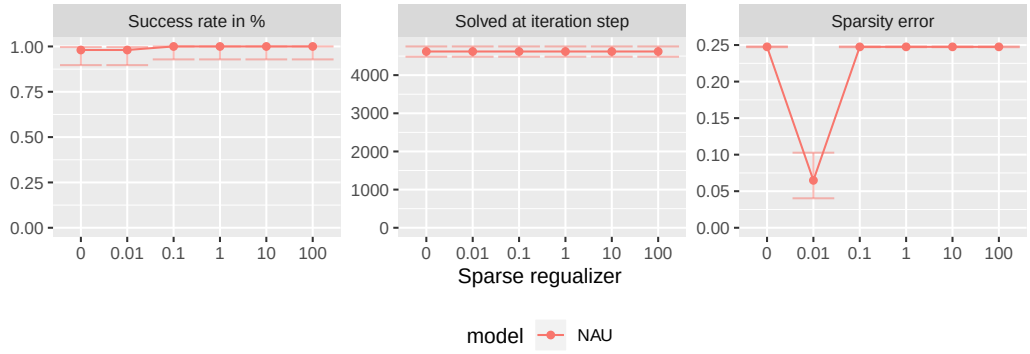


Figure 11: Shows effect of the scaling factor $\hat{\lambda}_{\text{sparse}}$, on the arithmetic dataset for the $-$ operation.

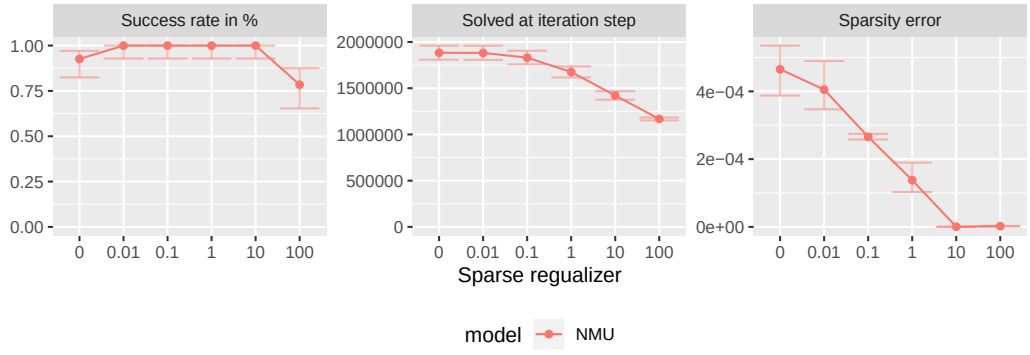


Figure 12: Shows effect of the scaling factor $\hat{\lambda}_{\text{sparse}}$, on the arithmetic dataset for the $\times$ operation.

C.7 COMPARING ALL MODELS

Table 6 compares all models on all operations used in NALU (**?**). All variations of model and operation, are trained for 100 different seeds. Some noteworthy observations are:

1. Division does not work for any model, including the $NAC_\bullet$ and NALU models. This may seam surprising but is actually inline with the results from the NALU paper (**?**, 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. If the NALU model had truly learned division to some degree, then both the interpolation and extrapolation range would yield a small error.
2. $NAC_\bullet$ and NALU are barely able to learn $\sqrt{z}$, with just 2% success-rate for NALU and 7% success-rate for $NAC_\bullet$. Thus we don't belive these models are useful for learning a $\sqrt{z}$ operation.
3. NMU is fully capable of learning z^2. It learns this by learning the subset twice in the NAU layer, this is also how $NAC_\bullet$ learns z^2.

Table 6: Shows the success-rate, at what global step the model converged at, and the sparsity error for all weight matrices, with 95% confidence interval. Best result is highlighed.

Op	Model	Success	Solved at		Sparsity error
		Rate	Median	Mean	Mean
$\times$	$\mathrm{NAC}_{\bullet,\mathrm{NMU}}$	$93\%\ ^{+4\%}_{-7\%}$	$1.8 \cdot 10^6$	$1.9 \cdot 10^6\ ^{+7.7 \cdot 10^4}_{-9.3 \cdot 10^4}$	$9.5 \cdot 10^{-7}\ ^{+4.2 \cdot 10^{-7}}_{-4.2 \cdot 10^{-7}}$
	$\mathrm{NAC}_{\bullet,\sigma}$	$\mathbf{100\%}\ ^{+0\%}_{-4\%}$	$2.5 \cdot 10^6$	$2.6 \cdot 10^6\ ^{+8.8 \cdot 10^4}_{-7.2 \cdot 10^4}$	$4.6 \cdot 10^{-5}\ ^{+5.0 \cdot 10^{-6}}_{-5.6 \cdot 10^{-6}}$
	$\mathrm{NAC}_\bullet$	$31\%\ ^{+10\%}_{-8\%}$	$2.8 \cdot 10^6$	$3.0 \cdot 10^6\ ^{+2.9 \cdot 10^5}_{-2.4 \cdot 10^5}$	$5.8 \cdot 10^{-4}\ ^{+4.8 \cdot 10^{-4}}_{-2.6 \cdot 10^{-4}}$
	NAC_+	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	Linear	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NALU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NAU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NMU	$98\%\ ^{+1\%}_{-5\%}$	$\mathbf{1.4 \cdot 10^6}$	$\mathbf{1.5 \cdot 10^6}\ ^{+4.8 \cdot 10^4}_{-6.5 \cdot 10^4}$	$\mathbf{4.2 \cdot 10^{-7}}\ ^{+2.9 \cdot 10^{-8}}_{-2.9 \cdot 10^{-8}}$
	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_+	$\mathbf{0\%}\ ^{+4\%}_{-0\%}$	—	—	—
	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: Shows the success-rate, at what global step the model converged at, and the sparsity error for all weight matrices, with 95% confidence interval. Best result is highlighed. *(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_{+}	$\mathbf{100\%}\ ^{+0\%}_{-4\%}$	$2.5 \cdot 10^{5}$	$4.9 \cdot 10^{5}\ ^{+5.2 \cdot 10^{4}}_{-4.5 \cdot 10^{4}}$	$2.3 \cdot 10^{-1}\ ^{+6.5 \cdot 10^{-3}}_{-6.5 \cdot 10^{-3}}$
	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	$\mathbf{100\%}\ ^{+0\%}_{-4\%}$	$\mathbf{1.8 \cdot 10^{4}}$	$3.9 \cdot 10^{5}\ ^{+4.4 \cdot 10^{4}}_{-3.7 \cdot 10^{4}}$	$\mathbf{3.2 \cdot 10^{-5}}\ ^{+1.3 \cdot 10^{-5}}_{-1.3 \cdot 10^{-5}}$
	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_{+}	$\mathbf{100\%}\ ^{+0\%}_{-4\%}$	$9.0 \cdot 10^{3}$	$3.7 \cdot 10^{5}\ ^{+3.8 \cdot 10^{4}}_{-3.8 \cdot 10^{4}}$	$2.3 \cdot 10^{-1}\ ^{+5.4 \cdot 10^{-3}}_{-5.4 \cdot 10^{-3}}$
	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_{+}	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	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: Shows the success-rate, at what global step the model converged at, and the sparsity error for all weight matrices, with 95% confidence interval. Best result is highlighed. *(continued)*

Op	Model	Success	Solved at		Sparsity error
		Rate	Median	Mean	Mean
z^2	$\mathrm{NAC}_{\bullet,\mathrm{NMU}}$	$\mathbf{100\%}\ ^{+0\%}_{-4\%}$	$1.4 \cdot 10^6$	$1.5 \cdot 10^6\ ^{+7.1 \cdot 10^4}_{-5.9 \cdot 10^4}$	$\mathbf{2.9 \cdot 10^{-7}}\ ^{+1.4 \cdot 10^{-8}}_{-1.4 \cdot 10^{-8}}$
	$\mathrm{NAC}_{\bullet,\sigma}$	$\mathbf{100\%}\ ^{+0\%}_{-4\%}$	$1.9 \cdot 10^6$	$1.9 \cdot 10^6\ ^{+5.3 \cdot 10^4}_{-6.2 \cdot 10^4}$	$1.8 \cdot 10^{-2}\ ^{+4.3 \cdot 10^{-4}}_{-4.3 \cdot 10^{-4}}$
	$\mathrm{NAC}_{\bullet}$	$77\%\ ^{+7\%}_{-9\%}$	$3.3 \cdot 10^6$	$3.2 \cdot 10^6\ ^{+1.6 \cdot 10^5}_{-2.0 \cdot 10^5}$	$1.8 \cdot 10^{-2}\ ^{+5.8 \cdot 10^{-4}}_{-5.7 \cdot 10^{-4}}$
	NAC_{+}	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	Linear	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NALU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NAU	$0\%\ ^{+4\%}_{-0\%}$	—	—	—
	NMU	$\mathbf{100\%}\ ^{+0\%}_{-4\%}$	$\mathbf{1.2 \cdot 10^6}$	$\mathbf{1.3 \cdot 10^6}\ ^{+3.1 \cdot 10^4}_{-3.6 \cdot 10^4}$	$3.7 \cdot 10^{-5}\ ^{+5.4 \cdot 10^{-5}}_{-3.7 \cdot 10^{-5}}$
	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 doesn't require a deep network. As such it doesn't tests if an arithmetic layer prevents the networks ability to be optimized using gradient decent.

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

As the performance of this task depends on the quality of the image-to-scalar network, as well as the arithmetic layer itself. As threshold has to be determined from 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 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.

A success-criteria is again used, as reporting the MSE is not interpretable, and models that don't 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 issue.

D.2 WITHOUT THE R_z REGULARIZER

As an ablation study of just the R_z regularizer, figure 13 shows the NMU and $\mathrm{NAC}_{\bullet,\mathrm{NMU}}$ models without the R_z regularizer. The success-rate is somewhat similar. However, as seen in the "sparsity error" plot, the solution is quite different.

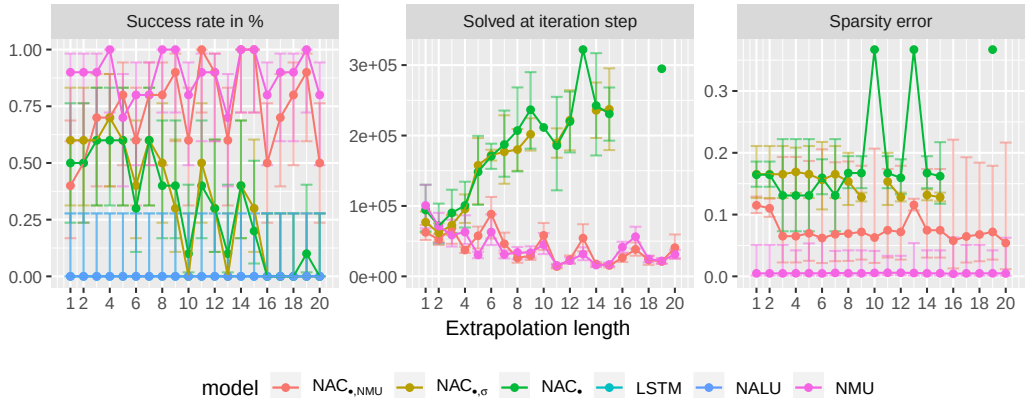


Figure 13: Shows the ability of each model to backpropergation and extrapolate to larger sequence lengths. The NMU and $\mathrm{NAC}_{\bullet,\mathrm{NMU}}$ models does not use the R_z regularizer.

E ADDITIONAL NOTES

E.1 LANGUAGE TO NUMBER TRANSLATION TASKS

In the NALU paper, they include a "Language to Number Translation Tasks" that appears to perform multiplication, by applying a recurrent NALU. However, after contracting the authors this turns out not to be the case.

- Question: In 4.3, how big is the embedding layer output dimensionality?
  Answer: From memory - i think it was 256.
- Question: In 4.3, how big is the LSTM layer output dimensionality?
  Answer: From memory - I think it was 10 (small)
- Question: In 4.3, what output and input activation does the LSTM layer have, tanh?
  Answer: It uses the standard LSTM construction - default in Tensorflow.
- Question: In 4.3, is the Linear/NAC/NALU layer only used for the final element in the sequence? See diagram.
  Answer: Correct.

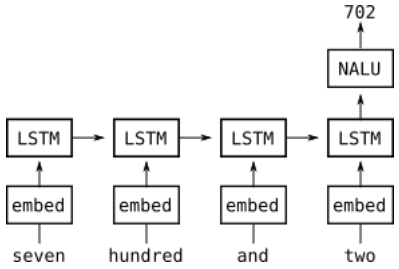


Figure 14: Diagram included in the email.

Based on these answers, we don't believe "Language to Number Translation Tasks" tests multiplication in any meaningful way. There is nothing inherently wrong in that, as **?** don't state this explicitly either, we just wish to clarify this.

Our reasoning is that, an embedding of 256, is more than enough to fully describe every single token, keep in mind there are only 29 tokens in total. As such there is no reason why an LSTM layer wouldn't be able to solve this on its own (a simplified LSTM solving this task is seen in (75)). Although, due to the non-linear activations, that are undesired in this case, the LSTM would need to downscale the values 0-1000 to be in the linear range of $\tanh(\cdot)$, and there would need to be final layer that upscales to 0-1000. We believe that the NALU layer serves this purpose and thus does not have any value in terms of arithmetics.

$$\begin{aligned} h_t = &\quad h_{t-1} \cdot f_t &&+ \quad \tilde{h}_t \cdot i_t \\ h_t = &\begin{bmatrix} h_{t-1} \\ 0 \\ 0 \end{bmatrix} \left(I(x_t = 100) \begin{bmatrix} 100 \\ 1 \\ 1 \end{bmatrix} \right) &&+ \begin{bmatrix} 0 \\ h_{t-1} \\ x_t \end{bmatrix} \left(I(x_t \neq 100) \begin{bmatrix} 100 \\ 1 \\ 1 \end{bmatrix} \right) \end{aligned} \tag{75}$$