# Neural Arithmetic Units

**Andreas Madsen**[†‡]
amwebdk@gmail.com

**Alexander Rosenberg Johansen**[†]
aler@dtu.dk

**Who else?**

[†]Technical University of Denmark [‡]Computationally Demanding

## Abstract

Arithmetic presents unique challenges to machine learning models as they are solved by inferring rules and axioms as opposed to interpolation of data. Neural networks has, with millions and sometimes billions of parameters, an ability to internalize incredible complex functions. However, when extrapolating to out-of-distribution examples they often fail as the neural network learns an approximation and not the underlying logic. We propose a plug-and-play module for neural networks that trains with stochastic gradient descent and can learn the of addition, subtraction and multiplication. Our proposed Neural Addition Unit (NAU) and Neural Multiplication Unit (NMU) rely on weight constraints to learn rules and extrapolate well beyond current solutions. The proposed NAU and NMU are inspired by the Neural Arithmetic Logic Unit (NALU). We find that replacing the nonlinearities in the weight matrix with a clipped linear function and adding an identity function to the log-exp multiplication unit is crucial for converging consistently. Through analytic and empirical analysis we justify how the NAU and NMU improve the Neural Arithmetic Logic Unit (NALU) and standard multi-layer perceptron (MLP) models. Our empirical results are fewer parameters, more consistent convergence, faster learning and more meaningful discrete values than the NALU.[1]

## 1 Introduction

The ability for neurons to hold numbers and do arithmetic operations have been documented in both humans, non-human primates **?**, as well as newborn chicks **?**, and even bees **?**. In our race to solve intelligence we have put much faith in neural networks, which in turn has provided unparalleled and often superhuman performance in many tasks requiring high cognitive ability**???**. However, when using neural networks to solve simple arithmetic problems, such as counting, they systematically fail to extrapolate**???**.

In this paper, we analyze and extend the recently proposed Neural Arithmetic Logic Unit (NALU)**?**. The NALU is a neural network layer with two modules. The two modules, $\text{NAC}_+$ for addition/subtraction and $\text{NAC}_*$ for multiplication/division, are selected softly between using a using a sigmoid gating mechanism. By using trainable weights, and restricting the weights towards $-1, 0, 1$

[1]In the interrest of scientific integrity, we have made the code for all experiments, and more, available on GitHub: `https://github.com/AndreasMadsen/stable-nalu`.

Preprint. Under review.

the NAC$_+$ is able to approximate addition and subtraction between hidden units in the previous layer. The NAC$_*$ extends this capability with the log-sum-exp trick to achieve multiplication and division. These rules should be learned only by observing arithmetic input-output pairs and using backpropagation**?**.

In practice, training the NALU can be cumbersome. Through an gradient analysis of main components in the NALU; the weight matrix constraint, the log-sum-exp trick and the gating we present the following findings: The weight matrix constraint in the NALU, under zero expectation of the mean layer value**?**, has a gradient of zero. The log-sum-exp trick has a treacherous optimization space with unwanted global minimas(as shown in figure...), exploding/vanishing gradients and variable co-dependencies. In particularly, when performing using the multiplication module, with the gate set to always choose multiplication, we observe that the wanted weight matrix values of $-1, 0, 1$ are rarely found. Furthermore, the gating between the addition/subtraction component and the multiplication/division component has a gradient that, in expectation, bias' towards the NAC$_+$.

Motivated by these convergence and sparsity issue, we propose alternative formulations of the NAC$_+$ and NAC$_*$, which we call the Neural Addition Unit (NAU) and Neural Multiplication Unit (NMU). That is, we will assume that the appropriate operation is already known, or can empirically be found by varying the network architecture (oracle gating). We propose an alternative formulation of the matrix constraint, using a clipped linear activation, and the log-sum-exp, using identity mappings, both which significantly improves upon the NAC$_+$ and NAC$_*$ through extensive testing on synthetic tasks and images.

## 2 Improving NAC and NALU

The NALU **?** is a neural unit capable of doing exact addition or multiplication, controlled by a sigmoid-gating-mechanism. The addition part is trivial, as this is just a matrix multiplication $\mathbf{a} = \mathbf{W}\mathbf{x}$. The only special part is that the weight matrix $\mathbf{W}$ is constrained to be between $-1$ and $1$. This this done using a $\mathbf{W} = \mathrm{tahn}(\hat{\mathbf{W}})\sigma(\hat{\mathbf{M}})$ construction. Meaning that the weight matrix $\mathbf{W}$ is not trained directly, but computed from two auxiliary weight matrices. The core idea is that $\hat{\mathbf{W}}$ controls the sign and $\hat{\mathbf{M}}$ controls if the weight is zero. One of their core claims, is that this weight matrix construction have a sparse bias, which improves extrapolation for cases where a sparse weight is part of the underlying model.

For the multiplication, an exponential-log transformation is used in order to do exact multiplication using a matrix multiplication, $\mathbf{m} = \exp(\mathbf{W} \log(|\mathbf{x}| + \epsilon))$.

The addition unit (originally NAC), and the multiplication unit are in themself theoretically applicable in any neural network as well as being differentiable. The NALU, then combines them using a sigmoid-gating-mechanism $\mathbf{g} = \sigma(\mathbf{G}\mathbf{x})$[2] that choses by interpolation between addition and multiplication $\mathbf{z} = \mathbf{g} \odot \mathbf{a} + (1 - \mathbf{g}) \odot \mathbf{m}$.

In terms of the theory, the Original NALU paper **?** does not discuss anything more than mentioned so-far in this paper. To discuss why particular construction is a bad idea, and also suggests improvements which will be empirically validated later, the NAC and its multiplication variant is first re-formulated using scalar notation.

$$
\begin{aligned}
W_{h_\ell, h_{\ell-1}} &= \tanh(\hat{W}_{h_\ell, h_{\ell-1}})\sigma(\hat{M}_{h_\ell, h_{\ell-1}}) \\
\mathrm{NAC}_+ : z_{h_\ell} &= \sum_{h_{\ell-1}=1}^{H_{\ell-1}} W_{h_\ell, h_{\ell-1}} z_{h_{\ell-1}} \\
\mathrm{NAC}_\bullet : z_{h_\ell} &= \exp\left(\sum_{h_{\ell-1}=1}^{H_{\ell-1}} W_{h_\ell, h_{\ell-1}} \log(|z_{h_{\ell-1}}| + \epsilon)\right)
\end{aligned}
\tag{1}
$$

[2]The lack of bias term here is not a typo. Our preliminary investigations suggests that this is a hack to increase extrapolation of the gate. However in this paper the focus is only arithmetic operators themself.

## 2.1 Weight matrix construction

The weight matrix constructions $\text{tahn}(\hat{\mathbf{W}})\sigma(\hat{\mathbf{M}})$ have a few issues worth mentioning. First, the loss gradient with respect to the weight matrices, can without loss of generality, easily be derived to:

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

This reveals that this construction is particularly problematic, as the $E[\text{tahn}(\hat{W}_{h_{\ell-1},h_\ell})] = 0$ when $E[\hat{W}_{h_{\ell-1},h_\ell}] = 0$. Initializing $\hat{W}_{h_{\ell-1},h_\ell}$ to have zero expectation, is not just common choice but necessary in order to achieve $E[W_{h_{\ell-1},h_\ell}] = 0$, which is always a desired property in linear units such as as the NAC **?**.

The NALU **?** paper also claims that this weight matrix construction, creates a bias for $-1, 0, 1$. However, they provide no empirically or theoretical evidence to support that. In our own empirical investigation as seen in the experiments section, we also find no support for that claim.

To improve on both of these failings, we propose a simple clamped linear construction instead, that is regularize to have the desired bias of $\{-1, 0, 1\}$ and have gradient outside of $[-1, 1]$.

$$\begin{aligned}W_{h_{\ell-1},h_\ell} &= \min(\max(\hat{W}_{h_{\ell-1},h_\ell}, -1), 1), \\ \mathcal{R}_{\ell,\text{bias}} &= \frac{1}{H_\ell + H_{\ell-1}} \sum_{h_\ell=1}^{H_\ell} \sum_{h_{\ell-1}=1}^{H_{\ell-1}} \hat{W}^2_{h_{\ell-1},h_\ell}(1-|\hat{W}_{h_{\ell-1},h_\ell}|)^2 \\ \mathcal{R}_{\ell,\text{oob}} &= \frac{1}{H_\ell + H_{\ell-1}} \sum_{h_\ell=1}^{H_\ell} \sum_{h_{\ell-1}=1}^{H_{\ell-1}} \max(|\hat{W}_{h_{\ell-1},h_\ell}| - 1, 0)^2\end{aligned} \tag{3}$$

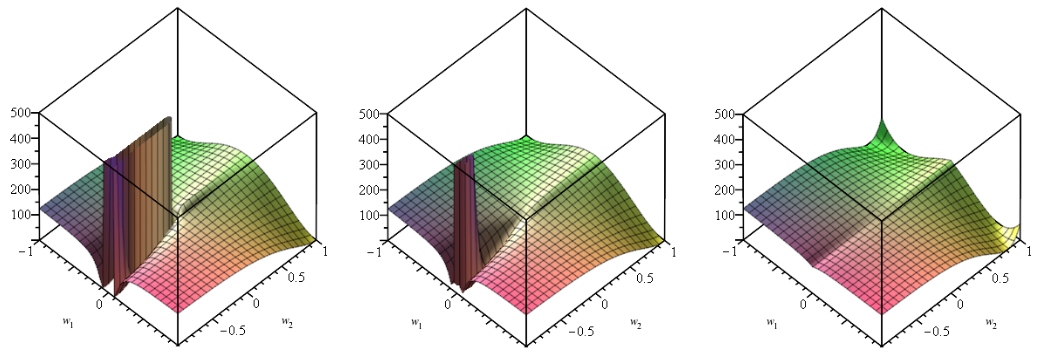
Note that while the bias regularizer $\mathcal{R}_{\ell,\text{bias}}$ also regularize $\hat{W}_{h_{\ell-1},h_\ell}$ to not be outside of $[-1, 1]$, one may choose a small regularization constant for this, or scale it up gradually as done in the experiments later. However, $\mathcal{R}_{\ell,\text{oob}}$ should always be present as it is never desired to have $\hat{W}_{h_{\ell-1},h_\ell} \notin [-1, 1]$.

## 2.2 Multiplication unit

The multiplication unit has its own issues. It should be easy to see that when $|z_{h_{\ell-1}}|$ is near zero and when $\hat{W}_{h_{\ell-1},h_\ell}$ is near $-1$ the $z_{h_\ell}$ value explodes. However, the issue extends beyond a weight near $-1$ as is revealed in the gradients, especially the backpropergation term $({}^{\partial z_{h_\ell}}/_{\partial z_{h_{\ell-1}}})$:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_{h_\ell,h_{\ell-1}}} &= \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} \frac{\partial z_{h_\ell}}{\partial w_{h_\ell,h_{\ell-1}}} = \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} z_{h_\ell} \log(|z_{h_{\ell-1}}| + \epsilon) \\ \frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}} &= \sum_{h_\ell=1}^{H_\ell} \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} \frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}} = \sum_{h_\ell=1}^{H_\ell} \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} z_{h_\ell} W_{h_\ell,h_{\ell-1}} \frac{\text{sign}(z_{h_{\ell-1}})}{|z_{h_{\ell-1}}| + \epsilon}\end{aligned} \tag{4}$$

In should be clear from ${}^{\text{sign}(z_{h_{\ell-1}})}/_{|z_{h_{\ell-1}}| + \epsilon}$ for $z_{h_{\ell-1}}$ near zero, the backpropagation term will not only explode, but can oscillate between a large postive value and large negative value, which is very problematic in optimization **?**. This issue does not only exists for $|z_{h_{\ell-1}}| < \epsilon$, which may have a small probability if $z_{h_{\ell-1}}$ has a wide distribution. But is can also be an issue for values outside of this interval.



(a) $\epsilon = 10^{-7}$ (b) $\epsilon = 0.1$ (c) $\epsilon = 1$

Figure 1: RMS loss curvature for a $\mathrm{NAC}_+$ layer followed by a $\mathrm{NAC}_\bullet$ layer. The weight matrices constrained are to $\mathbf{W}_1 = \left[\begin{smallmatrix} w_1 & w_1 & 0 & 0 \\ w_1 & w_1 & w_1 & 0 \end{smallmatrix}\right]$, $\mathbf{W}_2 = \left[\begin{smallmatrix} w_2 & w_2 \end{smallmatrix}\right]$. The problem is $x = (1, 1.5, 2, 2)\,, t = 11.25$. Desired solution is $w_1 = w_2 = 1$, although this problem have additional undesired solutions.

These observations are particular problematic when considering that $E[z_{h_{\ell-1}}] = 0$ is a desired property when initializing **?**. An alternative multiplication operator must thus be able to not explode for $z_{h_{\ell-1}}$ near zero. To that end we propose a new neural multplication units (NMU):

$$
\begin{aligned}
W_{h_{\ell-1},h_\ell} &= \min(\max(\hat{W}_{h_{\ell-1},h_\ell}, 0), 1), \\
\mathcal{R}_{\ell,\mathrm{bias}} &= \frac{1}{H_\ell + H_{\ell-1}} \sum_{h_\ell=1}^{H_\ell} \sum_{h_{\ell-1}=1}^{H_{\ell-1}} \hat{W}^2_{h_{\ell-1},h_\ell} (1 - \hat{W}_{h_{\ell-1},h_\ell})^2 \\
\mathcal{R}_{\ell,\mathrm{oob}} &= \frac{1}{H_\ell + H_{\ell-1}} \sum_{h_\ell=1}^{H_\ell} \sum_{h_{\ell-1}=1}^{H_{\ell-1}} \max\left( \left| \hat{W}_{h_{\ell-1},h_\ell} - \frac{1}{2} \right| - \frac{1}{2}, 0 \right)^2 \\
\text{NMU} : z_{h_\ell} &= \prod_{h_{\ell-1}=1}^{H_{\ell-1}} \left( W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell} \right)
\end{aligned}
\tag{5}
$$

This units does not support division. But supporting division is likely infeasible if $z_{h_{\ell-1}}$ near zero should not cause explosions. The NALU paper also shows that division doesn't work well for their unit, hence very little is lost here. On the other hand, this unit construction differentiates between negative and positive $z_{h_{\ell-1}}$ values, which should be considered an added bonus.

The gradients weight gradient and backpropagation term of the NMU are:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_{h_\ell,h_{\ell-1}}} &= \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} \frac{\partial z_{h_\ell}}{\partial w_{h_\ell,h_{\ell-1}}} = \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} \frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}} \left( z_{h_{\ell-1}} - 1 \right) \\
\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}} &= \sum_{h_\ell=1}^{H_\ell} \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} \frac{\partial z_{h_\ell}}{\partial z_{h_{\ell-1}}} = \sum_{h_\ell=1}^{H_\ell} \frac{z_{h_\ell}}{W_{h_{\ell-1},h_\ell} z_{h_{\ell-1}} + 1 - W_{h_{\ell-1},h_\ell}} W_{h_{\ell-1},h_\ell}
\end{aligned}
\tag{6}
$$

These is much more well-behaved. Note also that the fraction does not explode for $z_{h_{\ell-1}}$ close to zero, as the denominator simply cancels out a term in $z_{h_\ell}$.

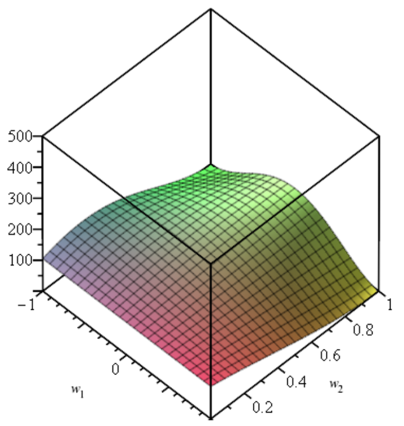


Figure 2: RMS loss curvature (without regularization) for a $\mathrm{NAC}_+$ layer followed by an NMU layer. Otherwise, the setup is identical to that in Figure 1.

### 2.3 Moments and initialization

Initialization is important for fast and consistent convergence. The desired properties are according to Glorot et al. **?**:

$$\begin{aligned} E[z_{h_\ell}] &= 0 & E\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &= 0 \\ Var[z_{h_\ell}] &= Var\left[z_{h_{\ell-1}}\right] & Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &= Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] \end{aligned} \tag{7}$$

The $\mathrm{NAC}_+$ layer is trivial, as this is just a linear layer. Thus the result from Glorot et al. ($Var[W_{h_{\ell-1},h_\ell}] = \frac{2}{H_{\ell-1}+H_\ell}$) can be used **?**.

In the case the modified $\mathrm{NAC}_+$, this condition is easy to satisfy. The original $\mathrm{NAC}_+$ is less trivial as $W_{h_{\ell-1},h_\ell}$ is not sampled directly. But 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 to be:

$$Var[W_{h_{\ell-1},h_\ell}] = \frac{1}{2r}\left(1 - \frac{\tanh(r)}{r}\right)\left(r - \tanh\left(\frac{r}{2}\right)\right) \tag{8}$$

One can the solve for $r$, given the desired variance.

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:

$$\begin{aligned} f(c_1,c_2) &= \left(1 + c_1 \frac{1}{2} Var[W_{h_\ell,h_{\ell-1}}] \log(|E[z_{h_{\ell-1}}]| + \epsilon)^2\right)^{c_2\ H_{\ell-1}} \\ E[z_{h_\ell}] &\approx f(1,1) \\ Var[z_{h_2}] &\approx f(4,1) - f(1,2) \\ E\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &= 0 \\ Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &\approx Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] H_\ell\ f(2,1)\ Var[W_{h_\ell,h_{\ell-1}}] \\ &\quad \cdot \left(\frac{1}{\left(|E[z_{h_{\ell-1}}]| + \epsilon\right)^2} + \frac{3}{\left(|E[z_{h_{\ell-1}}]| + \epsilon\right)^4} Var[z_{h_{\ell-1}}]\right) \end{aligned} \tag{9}$$

This is problematic because $E[z_{h_\ell}] \geq 1$, and the variance explodes for $E[z_{h_{\ell-1}}] = 0$ which is a desired property.

For our proposed NMU layer expectation and variance can be derived using the same assumptions as before, but no Taylor approximation is required:

$$
\begin{aligned}
E[z_{h_\ell}] &\approx \left(\frac{1}{2}\right)^{H_{\ell-1}} \\
E\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &\approx 0 \\
Var[z_{h_\ell}] &\approx \left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right)^{H_{\ell-1}} \left(Var[z_{h_{\ell-1}}] + 1\right)^{H_{\ell-1}} - \left(\frac{1}{4}\right)^{H_{\ell-1}} \\
Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_{\ell-1}}}\right] &\approx Var\left[\frac{\partial \mathcal{L}}{\partial z_{h_\ell}}\right] H_\ell \\
&\quad \cdot \left(\left(Var[W_{h_{\ell-1},h_\ell}] + \frac{1}{4}\right)^{H_{\ell-1}} \left(Var[z_{h_{\ell-1}}] + 1\right)^{H_{\ell-1}-1} - \left(\frac{1}{4}\right)^{H_{\ell-1}}\right)
\end{aligned}
\tag{10}
$$

These expectations are much more well 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 also more well-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 for an input with unit-variance and large $H_{\ell-1}$ and large $H_\ell$. However, feel free to compute more exact solutions.

Be aware that for $Var[z_{h_{\ell-1}}] > 4$ no solutions exists. Thus a third alternative is to normalize the input $z_{h_{\ell-1}}$ by its standard deviation and then upscale the output as $\hat{z}_{h_{\ell-1}} = z_{h_{\ell-1}} Var[z_{h_{\ell-1}}]^{\frac{1}{2} \sum_{h_{\ell-1}=1}^{H_{\ell-1}} W_{h_{\ell-1},h_\ell}}$. This does not maintain constant variance, but bounds how much it can grow and somewhat solves the initialization issue for an input with high variance.

## 3 Results

### 3.1 Simple function

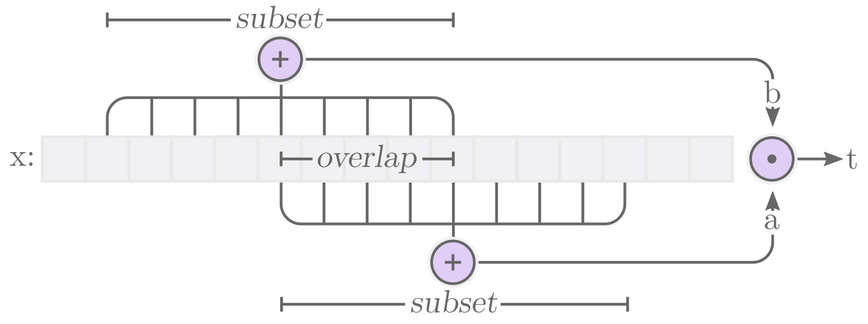


Figure 3: Lorem Ipsum.

#### 3.1.1 Simple multiplication task

Table 1: Shows the sucess-rate for extrapolation < $\epsilon$, at what global step the model converged at, and the sparse error for all weight matrices.

| Operation | Model | Success Rate | Converged at | Sparse error |
|---|---|---|---|---|
| | NAC$_\bullet$ | 13% | 2969 | $7.5 \times 10^{-6}$ |
| $a \cdot b$ | NALU | 26% | 3862 | $9.2 \times 10^{-6}$ |
| | NMU | 94% | 1677 | $3 \times 10^{-6}$ |

### 3.1.2 Static function task - defaults

Table 2: Shows the sucess-rate for extrapolation < $\epsilon$, at what global step the model converged at, and the sparse error for all weight matrices.

| operation | model | success.rate | converged.at | sparse.error |
|---|---|---|---|---|
| $a \cdot b$ | NAC$_\bullet$ | 40% | 3371250 | $4.7 \times 10^{-4}$ |
| | NALU | 0% | — | — |
| | NMU | 100% | 1571900 | $2.5 \times 10^{-3}$ |
| $a - b$ | NAC$_+$ | 100% | 6300 | $4.7 \times 10^{-1}$ |
| | Linear | 100% | 3300 | $3.7 \times 10^{-1}$ |
| | NALU | 40% | 1963250 | $4.3 \times 10^{-1}$ |
| | NAU | 100% | 3700 | $1.7 \times 10^{-3}$ |
| $a + b$ | NAC$_+$ | 100% | 42900 | $4.8 \times 10^{-1}$ |
| | Linear | 100% | 21300 | $6.1 \times 10^{-1}$ |
| | NALU | 10% | 81000 | $4.5 \times 10^{-1}$ |
| | NAU | 100% | 15500 | $2.1 \times 10^{-3}$ |

### 3.1.3 Static function task - boundary

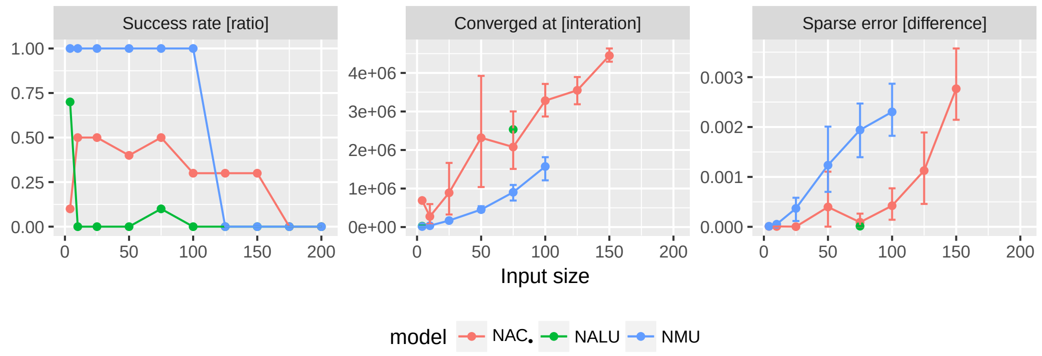


Figure 4: Lorem Ipsum.

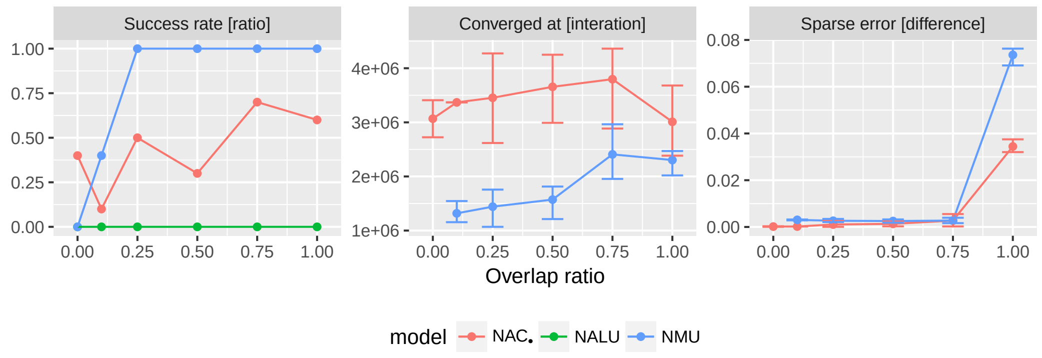


Figure 5: Lorem Ipsum.

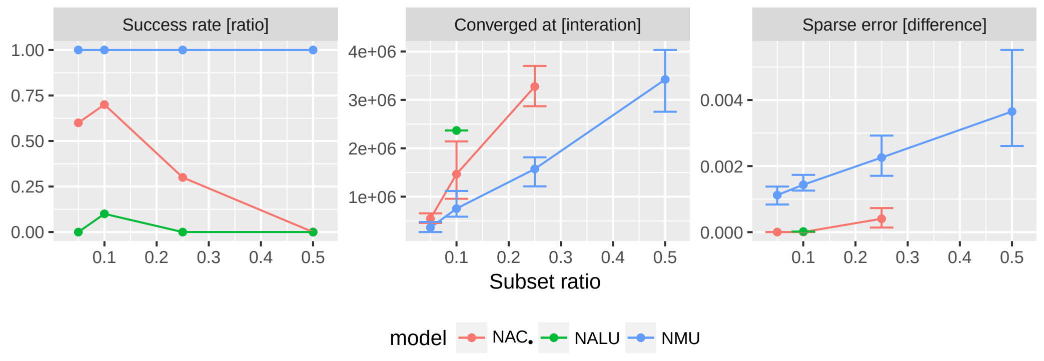


Figure 6: Lorem Ipsum.

## 3.2 sequential MNIST

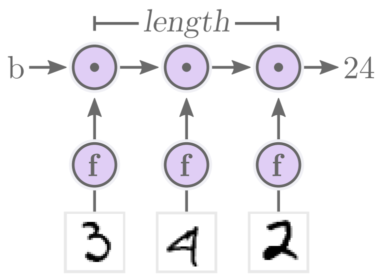


Figure 7: Lorem Ipsum.

### Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments go at the end of the paper. Do not include acknowledgments in the anonymized submission, only in the final paper.

## References