Neural Network Arithmetic Capabilities: a Literature Review

Anton Mukin

May 2025

Contents

Abstract

This is a literature review for the project ”Predictive Calculator”. It showcases why neural networks struggle with arithmetic simple tasks and presents possible previously already successfully implemented solutions, including methods for fusing continuous and discrete representations such as embedding discrete data and using a sequence to grid preprocessor.
The review additionally dives into why the activation function can limit the neural networks arithmetic solving capabilities.
Finally, the paper also discusses which architectures best perform in arithmetic tasks. The front runners being transformers and Recurrent Neural Networks (RNNs).

1 Introduction

1.1 My Project - A Predictive Calculator

This research is dedicated to designing a predictive calculator using various neural network algorithms for final project in Gymnasium (Matura Projekt). Normal calculators compute their answers using a set of mathematical rules programmed into them. I want to find out if it is possible to let a computer solve simple *arithmetic* operations not with a systematic approach, but rather by *predicting* the answers through a neural network, and see if it will be able to generalize beyond the trained data.
Another interesting aspect of this work is understanding the internal architecture of the trained neural network.

My approach involves training a neural network (NN) on a dataset, and then using it to predict the results of various expressions. The NN must be able to extrapolate simple arithmetic operators (i.e. addition) beyond its training range. It is important to consider and evaluate different types of NN architectures based on their performance for this task. While I recognize that there are other more efficient methods available, such as the neural arithmetic logic unit (NALU) (Trask et al., 2018), these methods also have their limitations. for instance, the NALU method does not generalize beyond a certain threshold. But they also have limitations, for example the NALU method does not generalize beyond a certain threshold, due to it’s systematic limitations.

While working on this, I am trying to learn more about neural networks and their architectures.

1.2 Research Question

Under a fixed budget for training-data and parameters, which neural-network architectures demonstrate the best systematic generalization on elementary arithmetic tasks (n-digit addition, subtraction, multiplication, and division) when evaluated on operand lengths and magnitudes that fall outside the training distribution, without relying on hand-coded or symbolic rules or other deterministic systems?

1.3 Keywords

- neural network OR deep learning
- generalization OR extrapolation
- different architectures
- arithmetic tasks

These keywords were used in various combinations to extract important literature from the Google Scholar database. Google Scholar was chosen because my school put forth and promoted it.

2 Neural Network Architecture

Neural networks are computational models within the family of deep learning. They are inspired by the biological neural networks found in our brains. A normal Feed-forward Neural Network (FNN) consists of neurons[1] interlinked together in a complex network via a certain weights. It is arranged so that the neurons can be divided into layers, where each neuron from the previous layer is connected to the next layer. There are mainly three different types of layers. Here is how Mienye and Swart, 2024 explains their functionality as follows: *"**The input layer** receives the raw data, such as images or text, which are then passed through **the hidden layers** where various computations and transformations occur. **The hidden layers** are where the model learns to extract relevant features and patterns from the data. Finally, **the output layer** produces the final prediction or classification based on the processed information."*

The training process of NN is essentially obtaining and adjusting weights between the neurons in the layers in such a way that it minimizes a loss function.

The three layers make up the architecture of a deep neural network[2]. Here, the first hidden layers are responsible for calculating the simpler parts of the problem, while the later ones decipher more intricate relationships (Jones, 2014).

[1] also called nodes

[2] Deep neural networks differ from shallow neural networks by their greater number of hidden layers.

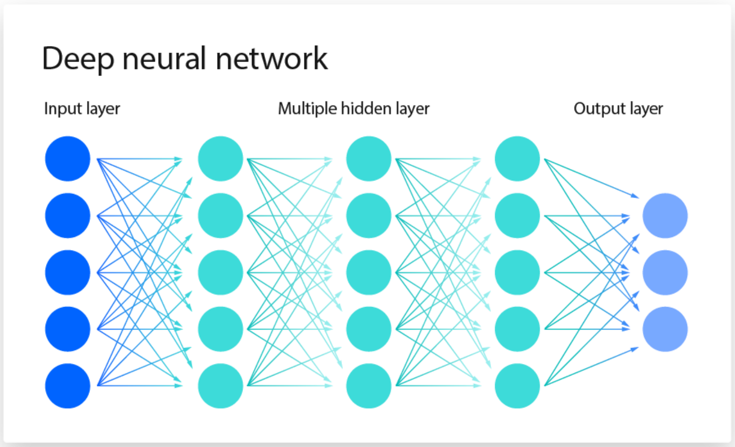

Figure 1: A deep neural network consisting of multiple nodes and layers.
IBM Think, 2024

The neurons in the hidden layers function the following way: they take inputs from the ones of the previous layer, apply an individual *weight* to each of them, sum the weighted inputs, and add some bias. After applying an activation function, which determines if the neuron will be activated, the result (output from the neuron) is calculated.

To make this clear, let j denote a neuron in the hidden layer and i denote a neuron in the preceding layer (e.g., input layer or previous hidden layer). The input received by neuron j from neuron i is x_i. The weight connecting neuron i to neuron j is w_{ji}. The bias for neuron j is b_j.

First, we calculate the weighted sum of inputs for neuron j, often called the "net input" or "pre-activation":

$$z_j = \sum_i (w_{ji} x_i) + b_j$$

Here, the summation $\sum_i$ indicates summing over all neurons i in the preceding layer that are connected to neuron j.

Next, this weighted sum z_j is passed through an activation function, denoted by f. The output of neuron j (which also serves as an input to neurons in the subsequent layer) is:

$$x_j = f(z_j)$$

or, substituting the expression for z_j:

$$x_j = f\left(\sum_i (w_{ji} x_i) + b_j\right)$$

Interestingly this concept was already pioneered in the middle of the 20th century (McCulloch & Pitts, 1943; Rosenblatt, 1958).

By tweaking the weights and biases of neurons, the model can be *trained* on available data. Over multiple *epochs* with the help of a *loss function* the model will adjust its parameters to predict output as accurately as possible.

For more details see the following amazing game and other references: GeeksforGeeks, 2025; Google Developers, 2018; IBM Think, 2024; ScienceDirect Topics, 2025; Smilkov and Carter, 2016.

It is worth mentioning that besides the architecture, there are also other concepts that could help with the predictive calculator project, such as: **activation function, loss function, and optimization algorithms**.

I have chosen to limit myself to architecture only for the sake of simplicity. This does not mean that adjusting the other parts of a neural network would not be effective for arithmetic tasks, as shown by this amazing paper focusing on the optimization algorithms (Abdulkadirov et al., 2023).

3 Why are Neural Networks so bad at Simple Arithmetic Problems?

Neural networks are very intriguing and useful because of their ability to recognize complex patterns in data, similar to a human brain. This allowed them to solve many difficult problems and play a big role everywhere, from our daily lives to the economy. Despite amazing achievements (Mienye & Swart, 2024) neural networks have made in the past decade, they still carry a reputation for being bad at math (Petruzzellis et al., 2023), which would be a key issue to resolve.

There are mainly two big underlying issues that make it difficult to teach a neural network simple arithmetic math:

- Neural networks process continuous domains, not discrete ones like mathematical numbers. See Section 3.1
- Studies have found a direct correlation between the nonlinearity of a NN's activation function and inaccuracy in arithmetic generalization problems. See Section 3.2

3.1 Continuous Domains

Simply put, a NN internally works with a distributed nature and uses float-type numbers; in simple arithmetic operations we use integers. This can lead to rounding errors and precision limitations. The reason for this distributed nature is the continuity of activation functions in a neural network (Dubey et al., 2022).

To be more precise, the output of each neuron and its individual weights all have to be float type numbers in order for the backpropagation process to be able to optimize the network and our model to actually "learn".

Testolin, 2024 put this into words while citing from Cartuyvels et al., 2021: *"Mathematical reasoning poses well-known challenges for connectionist models: the symbols used in mathematical formulas appear as arbitrary tokens that need to be manipulated according to well-defined rules entailing compositionality and systematicity. This is difficult to achieve for neural networks, which struggle to process discrete domains since they inherently represent and process information over continuous manifolds."*

Additionally, the general distributed nature of NNs is very different from a conventional calculator (Cartuyvels et al., 2021).
For example, the storage of the number 3 in a systematic algorithm is localized. In contrast, a neural network performing the same arithmetic task with the number 3 would store it across multiple neurons and layers, rather than a single neuron.

3.2 Nonlinear Activation Functions

Yet another reason lies within the fundamental way how neural networks work. Trask et al., 2018 measured a correlation between the nonlinearity of the activation function, a core component of neural networks, and their inability to generalize mathematical laws outside the training range.

The reason for the NN's inability to extrapolate lies within the NN's tendency to memorize patterns instead of generalizing rules. This is due ti their nonlinear activation functions tending to solve very complex relationships rather than simple arithmetic rules (He et al., 2024). In other words, this means the models tend to be overfitting[3] to random labels (CBMM MIT, 2021).

Which, in turn, means that activation functions, which behave more linearly, are able to deal with this overfitting problem better, as they don't *kill off* neurons as much as nonlinear activation functions do (Singla et al., 2021).

[3] A model that is overfitting has memorized the training data, but wasn't able to generalize a relationship between different parameters. Such models will show a very high training accuracy and a much lower validation accuracy (Mucci, 2024).

4 Architectural Approaches to Improving NN Performance in Arithmetic Tasks

Neural networks struggle with concepts and tasks requiring systematicity, compositionality and extrapolation (Kudo et al., 2023). These are concepts required when solving arithmetical tasks, which makes analyzing the performance of neural networks with these tasks interesting.

To improve the current rather poor performance of neural networks when faced with these tasks, Saxton et al., 2019 suggested that one possible solution would be going straight to the roots of the problem reviewed in section 3.1 and 3.2. Another approach is to consider various NN architectures like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and even attention-based models like transformers, because they are often cited as having better accuracy than FNNs (Mienye & Swart, 2024; Saxton et al., 2019; Testolin, 2024; Trask et al., 2018).

The first part of this section will be tackling the previously discussed fundamental problems of NNs and the second will present more about CNNs, RNNs and tranformers.

4.1 Addressing the Issues from Section 3

Finding a way to *mix* continuity and discreteness in neural networks is a hot topic in the scientific community at the moment. Since neural network optimization works better with continuous domains, a natural thought would be to make the discrete data continuous.
Cartuyvels et al., 2021 and Neurosymbolic Artificial Intelligence Journal, 2025 call for a change in neural networks to fuse discreteness and continuity.

One of the currently available methods is achieved by embedding discrete features while encoding the training data. This ”converts” discrete data (e.g. tabular or even numerical data) into continuous data. Cartuyvels et al., 2021 introduced important concepts for this.

Kim et al., 2021 developed an input preprocessor, implementing a similar idea, by converting the sequence input to a grid. Their so-called *seq2grid* preprocessor works very well, even enabling NNs to generalize arithmetic tasks.

To combat the memorization issue discussed in Section 3.2, Trask et al., 2018 found the Parametric Rectified Linear Unit (PReLU) activation function to be the best performing.

In the image below, neural networks with different activation functions[4] were trained on data with numbers in the interval $(-5, 5)$. Afterwards their respective Mean Absolute Errors (MAEs) were measured on arithmetic tasks with numbers within $(-20, 20)$ and mapped in a graph.

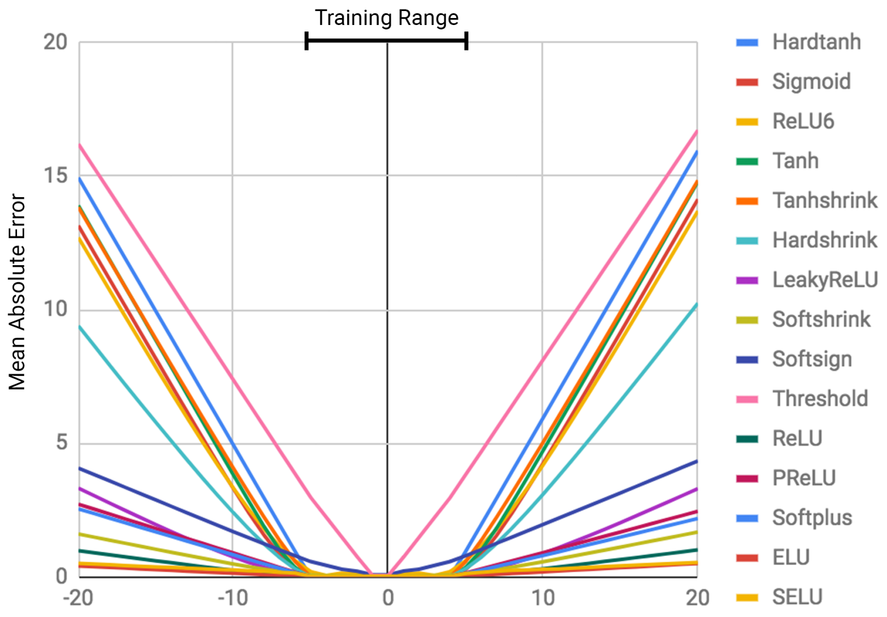

Figure 2: Various activation functions are described in the legend on the right in different color. In the graph, the activation functions with more nonlinear behavior show a higher MAE.
(Trask et al., 2018)

We can observe, that none of the activation functions are properly able to adjust to numbers outside the range they were trained on. Trask et al., 2018 explains this as follows: *"The severity of this failure directly corresponds to the degree of nonlinearity within the chosen activation function. Some activations learn to be highly linear (such as PReLU) which reduces error somewhat, but sharply nonlinear functions such as sigmoid and tanh fail consistently."*

[4]Most of these activation functions and their names are not relevant at the moment. The focus is set on the PReLU, tanh, and sigmoid.

For reference, below are the graphs of the three activation functions mentioned in the quote. (Dubey et al., 2022; Sakshi_Tiwari, 2025) the α parameter of the PReLU function is set to 0.1. In reality the model can adjust this parameter.

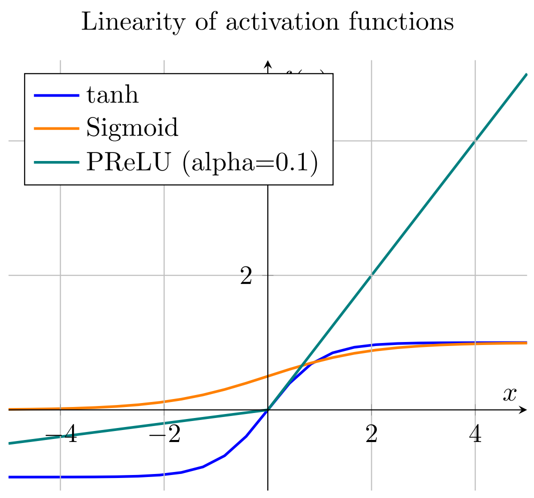

Figure 3: The three previously discussed activation functions are mapped out in a graph. Visibly, the tanh and sigmoid functions are curvy-shaped, while the PReLU function consists of two linear functions.
Trask et al., 2018

4.2 Different Architecture Types

In the last decade, many different types of architectures have been used to try to improve NN performance on arithmetic tasks. Here are some of the most interesting ones that have shown impressive results.

- Neuro-symbolic approaches warrant discussion:
  - Neural Arithmetic Logic Units (NALU) are very efficient, because they utilize little computational power but show high accuracy (Trask et al., 2018).
  - By providing the NN with a virtual abacus, Petruzzellis et al., 2023 utilizes this method to achieve impressive results, capable of generalization.

  Sadly, these neuro-symbolic models do not align with my expectations for this project, because they are only able to generalize up to a certain threshold. This means they will be disregarded for the project.

- A plausible approach would be to extend a NN with external memory, as it was done by Graves et al., 2016; Kaiser and Sutskever, 2016, with great results.
- Possible architectures that showed improvement:
  - Transformers (Vaswani et al., 2023) are the most promising type of architecture as suggested by Saxton et al., 2019 and Nogueira et al., 2021. Lee et al., 2023 was successful in training a small transformer model to accurately solve arithmetic problems. McLeish et al., 2024 recorded incredible accuracy from a transformer with positional embeddings.
  - Recurrent Neural Networks are a plausible architecture, similar to transformers and fit for arithmetic tasks as shown in Bowman et al., 2015.
    More recently Hoedt et al., 2021 outperformed the NALU from Trask et al., 2018 and possibly might also outperform the transformer models evaluated in Saxton et al., 2019.

5 Takeaways for the Predictive Calculator

For the predictive calculator project earlier described in Section 1.1, I will try to utilize as many of the aforementioned architectures as possible, in hopes of finding an architecture capable of solving arithmetic tasks by generalization.

For models with the best accuracies I should use the PReLU activation function, as well as positional embeddings. Additionally some models can be trained with a *seq2grid* preprocessor or similar.

Because of my computational constraints, I have decided to work my way up from simple FNNs to RNNs, to transformers, to pre-trained Language Models.

According to this research, I should expect better results in the later of the aforementioned architectures.

6 Acknowledgments

Massive thanks to Herr Schneider for the help.
Thanks to my Mom, Lidia Mukina, and my Dad, Roman Mukin, for helping with grammatical errors.

References

Abdulkadirov, R., Lyakhov, P., & Nagornov, N. (2023). Survey of optimization algorithms in modern neural networks. *Mathematics*, *11*(11). https://doi.org/10.3390/math11112466

Bowman, S. R., Potts, C., & Manning, C. D. (2015). Recursive neural networks can learn logical semantics. https://arxiv.org/abs/1406.1827

Cartuyvels, R., Spinks, G., & Moens, M.-F. (2021). Discrete and continuous representations and processing in deep learning: Looking forward. *AI Open*, *2*, 143–159. https://doi.org/10.1016/j.aiopen.2021.07.002

CBMM MIT. (2021). Quantifying and Understanding Memorization in Deep Neural Networks [Accessed: 2025-05-30].

Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. https://arxiv.org/abs/2109.14545

GeeksforGeeks. (2025). Feedforward neural network [Last Updated: March 19, 2025; Accessed: 2025-05-24].

Google Developers. (2018). Neural networks — machine learning crash course [Last updated content; Accessed: 2025-05-24].

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., & Hassabis, D. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*, 471–476. https://doi.org/10.1038/nature20101

He, J., Zhao, J., Bai, A., & Hsieh, C.-J. (2024). Embedding space selection for detecting memorization and fingerprinting in generative models. https://arxiv.org/abs/2407.21159

Hoedt, P.-J., Kratzert, F., Klotz, D., Halmich, C., Holzleitner, M., Nearing, G., Hochreiter, S., & Klambauer, G. (2021). Mc-lstm: Mass-conserving lstm. https://arxiv.org/abs/2101.05186

IBM Think. (2024). What are neural networks? [Accessed: 2025-05-24].

Jones, N. (2014). Computer science: The learning machines. *Nature*, *505*(7482), 146–148. https://doi.org/10.1038/505146a

Kaiser, Ł., & Sutskever, I. (2016). Neural gpus learn algorithms. https://arxiv.org/abs/1511.08228

Kim, S., Nam, H., Kim, J., & Jung, K. (2021). Neural sequence-to-grid module for learning symbolic rules. https://arxiv.org/abs/2101.04921

Kudo, K., Aoki, Y., Kuribayashi, T., Brassard, A., Yoshikawa, M., Sakaguchi, K., & Inui, K. (2023, February). *Do deep neural networks capture compositionality in arithmetic reasoning?* https://doi.org/10.48550/arXiv.2302.07866

Lee, N., Sreenivasan, K., Lee, J. D., Lee, K., & Papailiopoulos, D. (2023). Teaching arithmetic to small transformers. https://arxiv.org/abs/2307.03381

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biophysics*, *5*(4), 115–133.

McLeish, S., Bansal, A., Stein, A., Jain, N., Kirchenbauer, J., Bartoldson, B. R., Kailkhura, B., Bhatele, A., Geiping, J., Schwarzschild, A., & Goldstein, T. (2024). Transformers can do arithmetic with the right embeddings. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, & C. Zhang (Eds.), *Advances in neural information processing systems* (pp. 108012–108041, Vol. 37). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2024/file/c35986bc1ee29b31c1011481b77fe540-Paper-Conference.pdf

Mienye, I. D., & Swart, T. G. (2024). A comprehensive review of deep learning: Architectures, recent advances, and applications. *Information*, *15*(12). https://doi.org/10.3390/info15120755

Mucci, T. (2024). What is overfitting vs. underfitting? [Last updated: December 11, 2024; Accessed: June 1, 2025]. *IBM*.

Neurosymbolic Artificial Intelligence Journal. (2025). Call for Papers: Special Issue on Trustworthy Neurosymbolic AI in Regulated Domains: Advances, Challenges, and Applications [Accessed on 29.05.2025]. https://neurosymbolic-ai-journal.com/content/call-papers-special-issue-trustworthy-neurosymbolic-ai-regulated-domains-advances-challenges

Nogueira, R., Jiang, Z., & Lin, J. (2021). Investigating the limitations of transformers with simple arithmetic tasks. https://arxiv.org/abs/2102.13019

Petruzzellis, F., Chen, L. X., & Testolin, A. (2023). Learning to solve arithmetic problems with a virtual abacus. *Proceedings of the Northern Lights Deep Learning Workshop*, *4*. https://doi.org/10.7557/18.6805

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386–408.

Sakshi_Tiwari. (2025). Activation functions in neural networks — geeksforgeeks [Last updated: 05 Apr, 2025].

Saxton, D., Grefenstette, E., Hill, F., & Kohli, P. (2019). Analysing mathematical reasoning abilities of neural models. https://arxiv.org/abs/1904.01557

ScienceDirect Topics. (2025). Feedforward neural network [Accessed: 2025-05-24].

Singla, V., Singla, S., Jacobs, D., & Feizi, S. (2021). Low curvature activations reduce overfitting in adversarial training. https://arxiv.org/abs/2102.07861

Smilkov, D., & Carter, S. (2016). Tensorflow playground [Interactive tool; Accessed: 2025-05-24].

Testolin, A. (2024). Can neural networks do arithmetic? a survey on the elementary numerical skills of state-of-the-art deep learning models. *Applied Sciences*, *14*(2), 744. https://doi.org/10.3390/app14020744

Trask, A., Hill, F., Reed, S., Rae, J., Dyer, C., & Blunsom, P. (2018). Neural arithmetic logic units. https://arxiv.org/abs/1808.00508

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention is all you need. https://arxiv.org/abs/1706.03762