# On the Abilities of Mathematical Extrapolation with Implicit Models

**Juliette Decugis**$^{1*}$ **Max Emerling**$^{1*}$ **Ashwin Ganesh**$^{1*}$
**Alicia Y. Tsai**$^{1*,2}$ **Laurent El Ghaoui**$^{1,2}$
$^{1}$UC Berkeley $^{2}$VinUniversity
{jdecugis, memerling, aganesh, aliciatsai, elghaoui}@berkeley.edu
{alicia.t, laurent.eg}@vinuni.edu.vn

## Abstract

Deep neural networks excel on a variety of different tasks. However, when presented with extrapolation tasks that consists of out-of-distribution data, these models tend to break down even on the simplest tasks. In this paper, we compare the performance of implicitly-defined and classical deep learning models on a series of mathematical extrapolation tasks, where the models are tested with out-of-distribution samples during inference time. Throughout our experiments, implicit models greatly outperform classical deep learning networks that overfit the training distribution. We showcase implicit models' unique advantages for extrapolation thanks to their flexible and selective framework. Implicit models, with potentially unlimited depth, not only adapt well to out-of-distribution data but also seem to learn mathematical functions better.

## 1 Introduction

Learning to extrapolate – the ability to infer unknown values that extend the application of a method or conclusion beyond the current scope of the known data – is a core ability of human intelligence and an important development towards general machine intelligence. Although contemporary neural networks have demonstrated remarkable success in a myriad of domains, they struggle greatly when faced with data points outside of their training distribution [3, 15]. In this work, we investigate the capability of implicitly-defined neural networks [1, 5, 6] to extrapolate on mathematical tasks.

Implicitly-defined neural networks, such as implicit deep learning [6] or deep equilibrium models (DEQ) [1], are a general class of deep learning models that has been proposed as a potential alternative to classical neural networks. These models do not operate on the premise of explicitly defined layers, but instead have an internal states that are defined via an "equilibrium" (fixed-point) equation, and the outputs are determined only implicitly by the underlying equilibrium equation. Formally, for a given data point $u$, an implicit model solves the equilibrium equation $z = \phi(Az + Bu)$, where $z$ is the equilibrium state for an input $u$, $\phi$ is an non-linear activation such as ReLU, and matrices $A, B$ are model parameters. The prediction is obtained by feeding the equilibrium state $z$ through an affine transformation, $\hat{y}(u) = Cz + Du$, where matrices $C, D$ are also model parameters. Recent results have shown successes of the implicit models [2, 9, 19]. There has also been emerging work where the equilibrium state is interpreted as a closed-loop feedback system from a neural science's perspective [13]. Inspired by these successes, we seek to explore the mathematical generalization capabilities of an implicit model in learning to extrapolate and comparing their capabilities to those of transformers and other architectures specialized for arithmetic computation [18].

## 2 Related Work

Prior work on logical reasoning have largely focused on developing specialized neural network models to accomplish algorithm learning tasks [7, 8, 11, 12, 17]. Some of these papers take advantage of external memory sources while others suggest allowing the network to iterate longer when it comes to more complex inputs. Deep Equilibrium Models, a specific instance of the variable depth implicit models, have shown superior out of distribution performance. They use a higher number of root-finding iterations before converging for more complex inputs [12]. Neural Arithmetic Logic Units (NALU) attempt to answer the extrapolation shortcomings of Deep Neural Network's on arithmetic tasks. NALU is a novel architecture that is able to explicitly represent mathematical relationships using the neurons of the network [18]. However, even with the improved NALU (iNALU), instability is seen in training and random reinitializations are required [16].

Transformers are well-suited for addition and subtraction tasks achieving very high accuracy on interpolation experiments [14]. However, when faced with OOD data, testing on longer numbers than the model was trained on, only the transformer with larger than three billion parameters performed well [14]. The same OOD concerns are also observed when experimenting on arithmetic tasks with BART, a denoising auto-encoder using a transformer-based architecture [20]. On the contrary, when transformers were tested on matrix inversion and eigenvalue decompensation, even with OOD data they provided solutions that were "roughly correct" and demonstrated some mathematical understanding [4].This potentially suggests that transformers are better suited for more complex mathematical tasks where there is some possibility of demonstrating mathematical understanding without explicitly solving the problem correctly.

## 3 Methodology

We consider three types of mathematical extrapolation tasks: 1) modeling the identity function, 2) performing arithmetic operations, and 3) modeling rolling functions over sequential data. We evaluate model robustness on out of distribution shifts in the training mean. We compare implicit models with various classical deep learning models (MLP, LSTM and Transformers) and with Google's Neural Arithmetic Logic Units (NALU) [18] built for out of distribution arithmetic tasks.

**ImplicitRNN.** In this paper, we introduce implicit recurrent neural networks designed to receive elements in a sequence one timestep at a time and maintain at least one hidden state through a linear layer to enforce sequence-based memory. Based on vanilla RNNs, we use a single recurrent layer with input size $I$ and output size $O$, where the output doubles as our recurrent-hidden state. We implement this recurrent layer as a single implicit layer with input size $I + O$, output size $O$, and a variable hidden size $H$. Note that the implicit hidden state refers to $X$ and it is not to be confused with the recurrent-hidden state (which is the same as the recurrent layer's output after the first input of the sequence). We initialize the recurrent-hidden state $h$ to be the zero vector. A single forward pass for input element $i$ of sequence s takes the following steps: (1) concatenate element $i$ and recurrent hidden-state $h : x := [i, h]$; (2) pass $x$ through the implicit layer, producing output $o$; (3) set $h := o$. At any step, $o$ can also be used as the layer's output prediction (i. e., $o$ is the prediction corresponding to input i).

**Identity function.** It has been shown that neural networks struggle to learn the basic task of identity mapping, $f(x) = x$, where models should return the exact input as given [10, 18]. We train on 10,000 data sampled from a uniform distribution with an input dimension of 10, $x_{\text{train}} \in \mathbb{R}^{10} \sim U(-5, 5)$, and test on 1,000 data drawn from multiple shifted uniform distributions, $U(-\kappa, \kappa)$, where $\kappa$ ranges from 10 to 80, for instance $x_{\text{test}} \in \mathbb{R}^{10} \sim U(-10, 10)$. We train for 500 epochs for the MLP and 1,000 epochs for both the implicit models and Transformer encoder, all with a learning rate of 0.01. We compare an implicit model with $A \in \mathbb{R}^{4\times4}, B \in \mathbb{R}^{4\times10}, C \in \mathbb{R}^{10\times4}, D \in \mathbb{R}^{10\times10}$ (196 parameters), a 2-layered MLP network of size $10 \times 10 \times 10$ (220 parameters including biases) and a transformer encoder with a single attention head, the smallest encoder we could generate for this task (43,498 parameters).

**Arithmetic operations.** We focus on two arithmetic operations: addition and subtraction. The models take in 10,000 training arrays of length 50. Replicating the task proposed by Trask et al., we randomly select four numbers $i < j$, $k < l$ from 1 to 50. For each sample, we construct two new numbers from a given array, $\vec{x} := \langle x_1, x_2, \cdots, x_{50} \rangle$. We take $a = \sum_{a=i}^{j} x_a$, $b = \sum_{b=k}^{l} x_b$ and predict $y = a + b$ for addition, and $y = a - b$ for subtraction. The training and testing data follow a uniform distribution where $x_{\text{train}} \in \mathbb{R}^{50} \sim U(-1, 1)$ and we expand or shrink our testing distribution by a factor of t symmetrically such that $x_{\text{test}} \in \mathbb{R}^{50} \sim U(-t/2, t/2)$ . For all our tasks, we compare an implicit model with $A \in \mathbb{R}^{20\times20}, B \in \mathbb{R}^{20\times50}, C \in \mathbb{R}^{1\times20}, D \in \mathbb{R}^{1\times50}$, a MLP and NALU both of size $50 \times 10 \times 10 \times 1$ , and two transformers: a sequential and a depth-wise encoder. The sequential encoder processes each array as a single sequence and has a single layer with ten attention heads. On the other hand, the depth-wise encoder processes each element in a given array as a single sequence and has a single layer encoder with one attention head.

**Sequence modeling.** We perform three sequence modeling tasks: rolling average, rolling argmax and spiky time series predictions. The rolling average task consists of predicting for each time step the average of the sequence up to current time step $j$, $\sum_{i=1}^{j} x_i/j$. We train on sequences drawn from a normal distribution, $x \sim \mathcal{N}(3, 1)$ and test on sequences with a shifted mean such that $x \sim \mathcal{N}(3 + t, 1)$. We compare an LSTM (1×100×100×1, 42,210 parameters), an implicit model ($A \in \mathbb{R}^{200\times200}, B \in \mathbb{R}^{200\times10}, C \in \mathbb{R}^{10\times200}, D \in \mathbb{R}^{10\times10}$, 44,100 parameters) and a transformer encoder (with a single layer and 2 attention heads, 42,210 parameters).

The rolling argmax task predicts at each time step the index of the max value seen by the model so far. We train on sequences $x \sim U(0, 1)$ and test on sequences with extrapolation factor t where $x \sim U(0, t)$. We compare sequential models who process the sequence one timestep at a time: implicitRNN ($A \in \mathbb{R}^{18\times18}, B \in \mathbb{R}^{18\times23}, C \in \mathbb{R}^{22\times18}, D \in \mathbb{R}^{22\times23}$ followed by a single linear layer of size 22×10, 1,870 parameters) and LSTM (1×19×19×10, 1872 parameters). We implement a masked transformer decoder (with a single layer and two attention heads, 1920 parameters) which also only has access to previous inputs at a given timestep. We also compared our results to an unmasked transformer decoder (with a single layer and two attention heads, 1,920 parameters) and a regular implicit deep learning model ($A \in \mathbb{R}^{33\times33}, B \in \mathbb{R}^{33\times10}, C \in \mathbb{R}^{10\times33}, D \in \mathbb{R}^{10\times10}$, 1,849 parameters). In contrast with sequential models, our two transformers and implicit model process the entire sequence at once rather than timestep per timestep.

Finally, for the spiky time series forecasting task, we first generate a time series sequence, then we randomly insert spikes designed from a combination of sine functions (see more details in the appendix). This is not an extrapolation task but rather aims to understand whether our models can predict sudden changes in the data. We compare an implicitRNN ($A \in \mathbb{R}^{20\times20}, B \in \mathbb{R}^{20\times21}, C \in \mathbb{R}^{20\times20}, D \in \mathbb{R}^{20\times21}$ followed by a linear layer 20×1, 1661 parameters), a masked transformer decoder (single layer and 10 attention heads, 43,529 parameters) and a LSTM (1×20×20×1, 1861 parameters).

## 4 Experiments

For the modeling identity task, Figure 1 shows the MSE (mean squarred error) evaluated on the testing set for the MLP, implicit model and transformer. Our implicit model maintains testing MSE $< 5$ for training distribution shifts from 0 to 25. Even for very large distribution shifts of up to 40 where $x_{\text{test}} \in \mathbb{R}^{10} \sim U(-45, 45)$, the implicit model's testing MSE only grows by a factor of 10. In comparison, the MLP and transformer encoder testing errors surpass 10 with distribution shifts of only 10 and 5 respectively. We observe the MLP and transformer fail to model the actual identity function and instead replicate patterns observed in the training distribution which leads to increasing error as our testing set shifts away. Specifically, the transformer encoder's MSE explodes the fastest which may be partly explained by the model's very large size and therefore higher potential to overfit to a small training set. In contrast, our implicit model converged after only 4 iterations when predicting on the testing data. We conclude that implicit models, not restricted to a specific number of layers, can limit overfitting through faster convergence when given simple functions. They can therefore effectively model simple mathematical functions such as the identity.

For addition and subtraction tasks, Figure 2 and Figure 3 compare training and out of distribution testing log MSE loss for our five models. In Figure 3, we observe our implicit model outperforms all other models maintaining the lowest testing loss across distribution shifts for both addition and subtraction tasks.

It successfully seems to learn the operations as demonstrated by low training loss and its ability to replicate the operation on out of distribution inputs with testing loss $< 1$ for distribution shifts $< 100$. In Figure 3, transformers, MLP and implicit models' log MSE loss seem to similarly linearly increase as our log distribution shift factor increases. Further results in our appendix, Table 2 and Figure 6 however demonstrate the extrapolation advantages of implicit models on even small distribution shifts. Furthermore as suggested by their high training MSE in Figure 2, transformers seem to underfit the training data which results in their higher extrapolation testing loss. Implicit models therefore appear a better out of the box model for tasks with fewer training samples. Surprisingly, the NALU model, designed for extrapolation on arithmetic tasks, performs the worst as shown in Figure 3 where its testing loss surpasses $10^{10}$ for an extrapolation shift of only 10. Across our experiments, we weren't able to replicate robust out of distribution predictions with the NALU model from Trask et al. [18]. We suspect the model's performance is inflated by hand-crafted evaluation metrics and therefore does not perform well when evaluated using more traditional metrics.

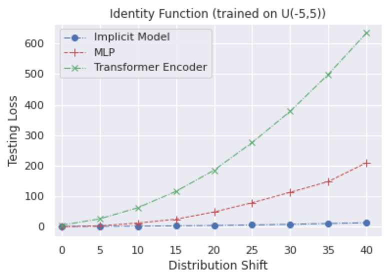

Figure 1: Testing MSE of the MLP, implicit model and transformer encoder evaluated on different testing distribution shifts.

As suggested by Kaiqu Liang et al. [12], the more selective nature of implicit models may help them generalize better on logical tasks. For a specific input X, an implicit model's training only terminates if we find a fixed point representation of X through our equilibrium equation. During training on both arithmetic operations, we observed our model failed to converge for at least 1/3 of our epochs. Implicit models would have the ability to filter out internal representations that do not help capture the given arithmetic operation. On the other hand, MLPs forward pass always terminates in a given number of steps; when the input has gone through each layer. Therefore, the MLP may have a higher chance of overfitting to the training data.

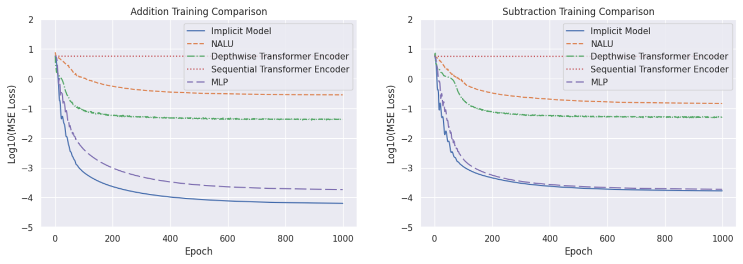

Figure 2: Training Log(MSE) per number of epochs for the five models evaluated on addition and subtraction. We observe that the implicit model achieves the lowest training loss across both tasks.

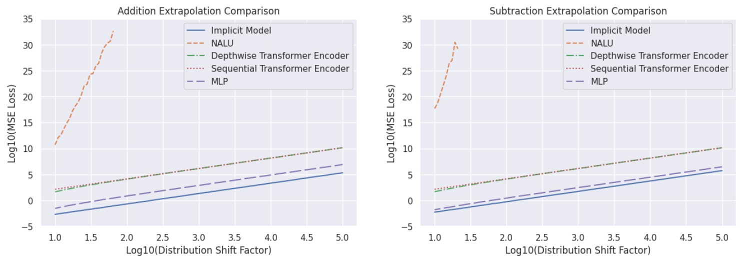

Figure 3: Testing Log(MSE) of our five models evaluated on testing distribution shifts. We observe the implicit model strongly outperforms all other models on OOD data and maintains a linear increase.

We summarize our results on out of distribution inputs for the three sequence modeling tasks in Figure 4 and 5. For the rolling argmax task, we observe in Figure 4 that our implicit models maintain the highest and a very stable testing accuracy across distribution shifts. Both our transformers demonstrate a similar capacity to extrapolate on out of distribution inputs with however a lower accuracy (by at least 5%). In contrast, the LSTM fails to extrapolate as their accuracy drops by almost

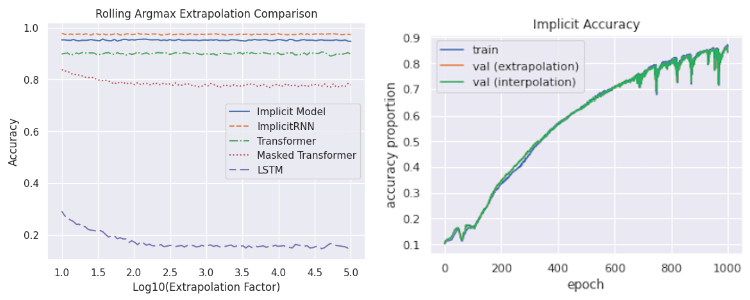

Figure 4: As the extrapolation factor increases the implicit model and implicitRNN are able to maintain superior accuracy on the rolling argmax task. On the left, we observe with extrapolation factor t = 10, our implicitRNN performs similarly on interpolated and extrapolated data.

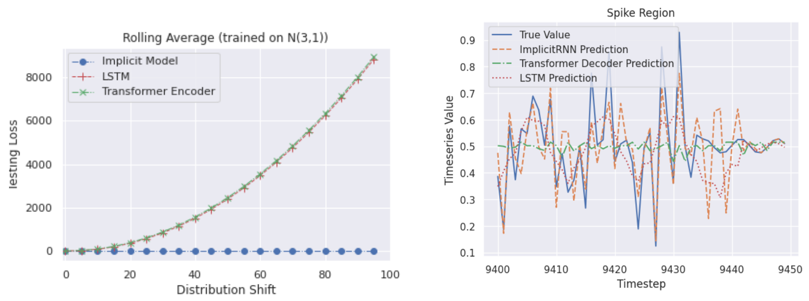

Figure 5: Testing results for the rolling average and spiky time series prediction tasks. For the rolling average, we observe our implicit model maintains close to constant loss across shifts in contrast with the LSTM and transformer. On the right plot, in the spiky regions, the implicit RNN more accurately predicts the magnitude of the spikes.

50% when evaluated on out of distribution inputs. Figure 5 shows test MSE across distribution shifts for the rolling average task and an example test sequence predictions for the spiky data task. For the rolling average task, the LSTM and transformer both replicate the training distribution, predicting averages around 3 even in the test set, whereas the implicit model extrapolates to higher values. For the spiky data predictions, although LSTM and implicit models have similar testing losses, the implicit model seems to have a better understanding of overall sequence structure. It successfully predicts the specific location and magnitudes of spikes. Note that this is not an extrapolation task as the training and testing regimes had a similar proportion of spikes. However, given very few examples of anomalous structure in the data the implicit model performs very well when similar structure appears in the test set. Across these sequential tasks, we hypothesize that implicit and implicitRNN models here benefit from greater model flexibility. Specifically for sequential data, the implicit layers within an implicitRNN can run for more iterations when presented with more complex inputs.

## 5 Conclusion

Our results showcase implicit models' superior performance on out of distribution sample points when compared to traditional deep learning models. This ability is especially apparent on mathematical tasks such as function learning and arithmetic operations. With potentially infinite depth, implicit models have the ability to grow for more complex inputs and adapt to data shifts. In all of our experiments, implicit models showcased improved performance over MLP, LSTM and transformers, which sometimes had 10 times as many model parameters. Implicit models also demonstrate much better OOD generalization abilities than Google's NALU, an architecture specifically designed to perform well on arithmetic tasks. These results motivate the further study of implicit models as a robust framework to excel under distribution shifts.

## References

[1] S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/01386bd6d8e091c2ab4c7c7de644d37b-Paper.pdf.

[2] S. Bai, V. Koltun, and J. Z. Kolter. Multiscale deep equilibrium models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5238–5250. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/3812f9a59b634c2a9c574610eaba5bed-Paper.pdf.

[3] D. Barrett, F. Hill, A. Santoro, A. Morcos, and T. Lillicrap. Measuring abstract reasoning in neural networks. In *International conference on machine learning*, pages 511–520. PMLR, 2018.

[4] F. Charton. What is my math transformer doing? – three results on interpretability and generalization, 2022. URL https://arxiv.org/abs/2211.00170.

[5] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

[6] L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021. doi: 10.1137/20M1358517. URL https://doi.org/10.1137/20M1358517.

[7] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL http://arxiv.org/abs/1410.5401.

[8] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature.*, 538(7626), 2016-10. ISSN 0028-0836.

[9] F. Gu, H. Chang, W. Zhu, S. Sojoudi, and L. El Ghaoui. Implicit graph neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[11] Kaiser and I. Sutskever. Neural gpus learn algorithms, 2015. URL https://arxiv.org/abs/1511.08228.

[12] K. Liang, C. Anil, Y. Wu, and R. Grosse. Out-of-distribution generalization with deep equilibrium models.

[13] Y. Ma, D. Tsao, and H.-Y. Shum. On the principles of parsimony and self-consistency for the emergence of intelligence. *Frontiers of Information Technology & Electronic Engineering*, pages 1–26, 2022.

[14] R. Nogueira, Z. Jiang, and J. Lin. Investigating the limitations of the transformers with simple arithmetic tasks. *CoRR*, abs/2102.13019, 2021. URL https://arxiv.org/abs/2102.13019.

[15] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gR5iR5FX.

[16] D. Schlör, M. Ring, and A. Hotho. inalu: Improved neural arithmetic logic unit. *Frontiers in Artificial Intelligence*, 3, 2020. ISSN 2624-8212. doi: 10.3389/frai.2020.00071. URL `https://www.frontiersin.org/articles/10.3389/frai.2020.00071`.

[17] A. Schwarzschild, E. Borgnia, A. Gupta, F. Huang, U. Vishkin, M. Goldblum, and T. Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6695–6706. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/3501672ebc68a5524629080e3ef60aef-Paper.pdf`.

[18] A. Trask, F. Hill, S. E. Reed, J. Rae, C. Dyer, and P. Blunsom. Neural arithmetic logic units. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/0e64a7b00c83e3d22ce6b3acf2c582b6-Paper.pdf`.

[19] A. Y. Tsai, J. Decugis, L. E. Ghaoui, and A. Atamtürk. State-driven implicit modeling for sparsity and robustness in neural networks. *arXiv preprint arXiv:2209.09389*, 2022.

[20] C. Wang, B. Zheng, Y. Niu, and Y. Zhang. Exploring generalization ability of pretrained language models on arithmetic and logical reasoning. In L. Wang, Y. Feng, Y. Hong, and R. He, editors, *Natural Language Processing and Chinese Computing*, pages 758–769, Cham, 2021. Springer International Publishing. ISBN 978-3-030-88480-2.

## A Appendix

### A.1 Activation Function Experiments

For the identity function and arithmetic operation taks, we experimented with 15 different activation functions on our MLP: hardtanh, sigmoid, reLU6, tanh, tanhshrink, hardshrink, leakyrelu, softshrink, softsign, reLU, preLU, multipreLU, softplus, eLU and seLU. We tried to understand whether specific activations helped the MLP extrapolate as well as our implicit model. Table 1 summarizes the results of 5 of these activation functions on our identity function task as compared to the implicit deep learning model.

Table 1: Testing loss of our implicit model and five MLP models with specific activations on the identity function task. We observe the implicit model outperforms the MLP across activation functions. Description of the activation functions in the appendix.

| | **Train MSE** | | **Test MSE** | |
|---|---|---|---|---|
| **Activation** | **MLP** | **Implicit** | **MLP** | **Implicit** |
| ReLU | $2.14 \times 10^{-3}$ | $12.4 \times 10^{-1}$ | 21.6 | 2.16 |
| Leaky ReLU | $3.28 \times 10^{-3}$ | - | 22.3 | - |
| Softplus | $1.57 \times 10^{-2}$ | - | 17.1 | - |
| Softsign | $3.01 \times 10^{-1}$ | - | 47.5 | - |
| Log sigmoid | $1.71 \times 10^{-2}$ | - | 17.1 | - |

Table 2 compares the test MSE for our MLP with ReLU activation, the best MLP across all 15 activations and our implicit model. We have $x_{\text{train}} \in \mathbb{R}^{100} \sim U(1,2)$ and $x_{\text{test}} \in \mathbb{R}^{100} \sim U(2,5)$. For both operations, the implicit model greatly outperforms the MLP regardless of the activation function.

Table 2: Test MSE table of two MLPs and our implicit model on arithmetic operations. The best MLP for both tasks was with ReLU6 activation.

| **Operation** | **ReLU MLP** | **Best MLP** | **Implicit** |
|---|---|---|---|
| Addition | $6.95 \times 10^{31}$ | $8.50 \times 10^{3}$ | 16.07 |
| Subtraction | $3.69 \times 10^{19}$ | $1.87 \times 10^{4}$ | $3.40 \times 10^{-2}$ |

### A.2 Arithmetic Operations More Results

For more specific results on the OOD generalization capacities of implicit models, we compare in Figure 6 the training and validation loss on the addition task of both implicit and MLP models where $x_{\text{train}} \in \mathbb{R}^{100} \sim U(1,2)$ and $x_{\text{val}} \in \mathbb{R}^{100} \sim U(2,5)$. This is therefore one of our small distribution shifts where t = 3.

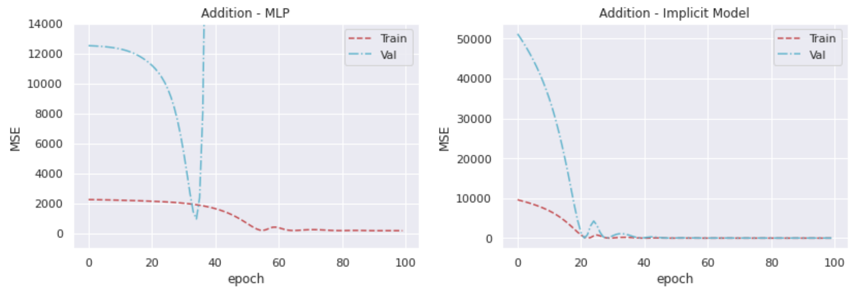

Figure 6: Testing and training MSE plots based on the number of training epochs for the addition and rolling argmax tasks. The MLP test loss bounces from low to high and eventually explodes whereas the implicit model achieves testing loss close to 0.

### A.3 Spiky Data Generation

Both the LSTM and the implicit model were trained on 7000 data points and tested on 3000 data points. The training regime featured 20 spiky regions of 100 data points each. The testing regime featured a proportionate amount of spiky regions. The data points in the spiky regions were sampled from $y = 5 \times (\sin(2x) + \sin(23x) + \sin(78x) + \sin(100x))$. The frequencies were arbitrarily chosen to be between 0 to 100 to generate a sufficiently spiky pattern. The magnitude of the spiky regions is at most 20. For the non-spiky regimes, the data points were sampled from $y = \sin(x)$.