On the Abilities of Mathematical Extrapolation with Implicit Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks excel on a variety of different tasks, often surpassing human intelligence. However, when presented with out-of-distribution data, these models tend to break down even on the simplest tasks. In this paper, we compare 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 understand the underlying structure of inputs much 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, 13]. In this work, we investigate the capability of implicitly-defined neural networks [1, 4, 5] to extrapolate.

Implicitly-defined neural networks, such as implicit deep learning [5] 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, ϕ 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, 8, 16]. There has also been emerging work where the equilibrium state is interpreted as a closed-loop feedback system from a neural science's perspective [12]. Inspired by these successes, we seek to explore the generalization capabilities of an implicit model in learning out of distribution shifts.

Submitted to the Workshop on Distribution Shifts, 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

2 Related Work

There is a lot of existing work using specially crafted neural networks to accomplish algorithm learning tasks [6, 7, 10, 11, 14]. Neural Turing Machines (NTM) and its successor, Differentiable Neural Computers (DNC), combine the capabilities of neural networks with external memory resources. This helps them learn very compact internal representations that allow them to generalize well beyond the training data on algorithm learning tasks [6, 7]. Neural GPUs take advantage of highly parallel convolution operations and a less recurrent structure to improve upon the training efficiency of DNCs [10]. Implicit models can provide similar extrapolation advantages over an LSTM without the need to use an external memory source. This is a huge advantage as training deep neural networks is already a memory-intensive process. Other studies propose running the RNN for additional timesteps [14], which is not feasible for many practical applications where high latency is key. Deep Equilibrium Models are another class of variable depth models that show superior out of distribution performance similar to Implicit models. They use a higher number of root-finding iterations before converging for more complex inputs [11].

3 Methodology

We consider three types of robustness tasks: 1) modeling identity function, 2) performing arithmetic operations, and 3) modeling sequential data. We evaluate model robustness on out of distribution shifts in the training mean and standard deviation. For each of the following tasks, we performed little hyperparameter tuning and only compared implicit models with classical deep learning models (MLP and LSTM) that have the same or similar size.

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 [9, 15]. We train on 10,000 data sampled from a uniform distribution with an input dimension of 10, $x_{\text{train}} \in \mathbb{R}^{10} \sim U(-1, 1)$, and test on 1,000 data drawn from an shifted uniform distribution, $x_{\text{test}} \in \mathbb{R}^{10} \sim U(-5, 5)$. For both models, we train for 1,000 epochs 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) and a 2-layered MLP network of size $10 \times 10 \times 10$ (220 parameters including biases).

Arithmetic operations. We compare the performance (measured by testing MAE) of an implicit model and a MLP on two arithmetic operations: addition and subtraction. The models take in 10,000 training arrays of length 10. Replicating the task proposed by Trask et al., we randomly select four numbers $i < j$, $k < l$ from 1 to 100. For each sample, we construct two new numbers from a given array, $\vec{x} := \langle x_1, x_2, \cdots, x_{100} \rangle$. 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 $x_{\text{train}} \in \mathbb{R}^{100} \sim U(1, 2)$ and $x_{\text{test}} \in \mathbb{R}^{100} \sim U(2, 5)$. For all our tasks, we compare an implicit model with $A \in \mathbb{R}^{5\times5}, B \in \mathbb{R}^{5\times100}, C \in \mathbb{R}^{1\times5}, D \in \mathbb{R}^{1\times100}$ and a MLP of size $100 \times 10 \times 10 \times 1$.

Sequence modeling. We perform three sequence modeling tasks: rolling average, rolling argmax and spiky data time series predictions. We compare our implicit model with a LSTM (Long-Short Term Memory), the classical model for sequential data. The rolling average task consists of predicting for each time step the average of the inputs' prior and current time steps. We train on sequences from discrete $X \sim \mathcal{N}(3, 1)$ and test on sequences from discrete $X \sim \mathcal{N}(6, 1)$. We use LSTM and standard implicit models both with hidden layer size 100. The rolling argmax task predicts at each timestep the index of the max value seen by the model so far. We train on sequences $x \sim U(1, 0)$ and test on sequences $x \sim U(1, 10)$. We compare an implicitRNN model with 506 parameters and a LSTM model with 520 parameters. Finally as a more application oriented task, we also perform time series forecasting on linear time series with randomly inserted spikes designed from a combination of sine functions (see more details in the appendix).

4 Experiments

For the modeling identity task, Figure 1 shows the MAE (mean absolute error) on the testing set of the MLP and implicit model. The MLP is unable to replicate the identity function outside of its training distribution, $x < -5$ or $x > 5$, whereas the implicit model achieves MAE = 0 for almost all

testing inputs. In order to reduce testing loss with the MLP, we experimented with different activation functions (see Table 3 in the appendix), the implicit model outperforms the MLP across all activation functions. Not restricted to layers, implicit models simulate infinite depth and therefore diminish the drawbacks of different activation functions on this simple computational task.

For addition and subtraction tasks, Table 1 compares the test MSE for our MLP and implicit model. To increase MLP performance, we experimented with 15 different activation functions (see appendix). For both operations, the implicit model greatly outperforms the MLP regardless of the activation function. Figure 2 shows the training and testing loss of the addition task where the MLP testing error is highly unstable and eventually explodes. Without inputs from a specific distribution, MLPs fail to understand the operation. In comparison, after sufficient training, the testing loss for our out of the box implicit model decreases to 0 and therefore successfully learns addition.

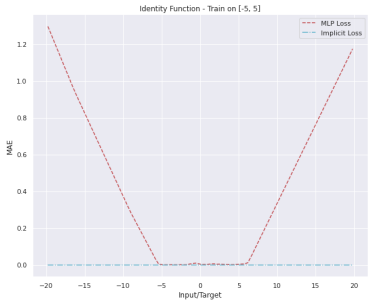

Figure 1: Testing MAE of the MLP and implicit model based on input values for the identity function task. We observe MLP MSE explodes for inputs outside the training distribution.

As suggested by Kaiqu Liang et al. [11], the more selective nature of implicit models may help them generalize better on logical tasks. The implicit deep learning model training only terminates if we find a fixed point representation of an input. Implicit models would have the ability to filter out internal representations that do not help capture the given arithmetic operation. On the other hand, MLP models' forward pass terminates for all inputs in a specific number of steps and therefore has a higher chance of capturing non essential dependencies within the training data.

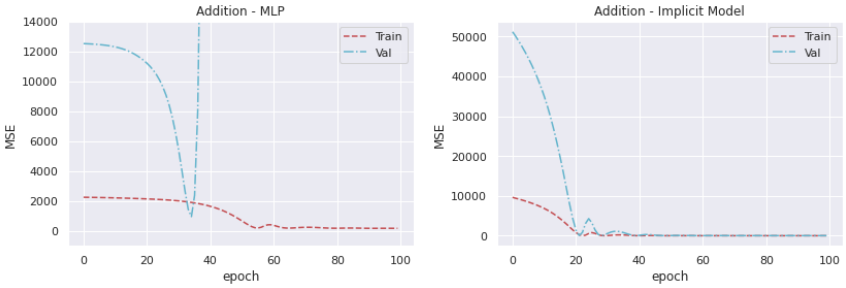

Figure 2: Testing and training MSE plots based on the number of training epochs for the addition and rolling argmax tasks. The MLP test loss bounces from low to high and eventually explodes whereas the implicit model achieves testing loss close to 0.

Table 1: Test MSE table of two MLPs and our implicit model on arithmetic operations. The best MLP for both tasks was with ReLU6 activation. The implicit model greatly outperforms MLP.

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

We summarize our results for the three sequence modeling tasks in Table 2. Figure 4 shows predictions for an example test sequence for rolling average and spiky data tasks. For the rolling average task, the LSTM replicates 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. Finally, Figure 3 depicts the rolling

argmax task comparing the training and testing accuracy for interpolated and extrapolated test set distributions. The implicit model maintains similar accuracy across test sets whereas the LSTM accuracy drops by 50.3 % when the model predicts on out of distribution data. We hypothesize that implicit and implicitRNN models here benefit from greater model flexibility. Specifically for sequential data, implicitRNN may grow as the number of processed time steps When encountering new testing data, implicit models deepen their network to interpret more complex unseen inputs.

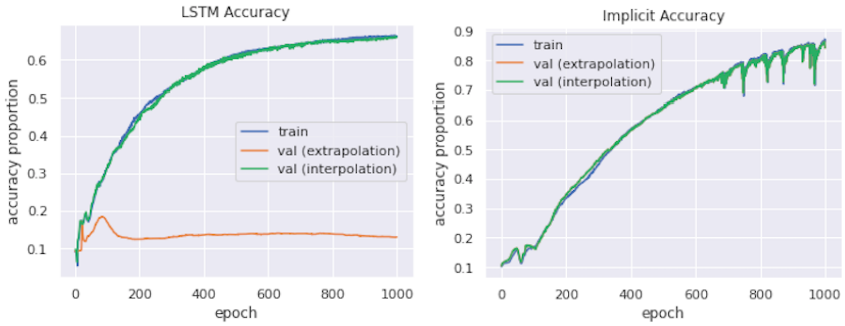

Figure 3: LSTM and implicit models' accuracy for numbers of training epochs. LSTM drops in accuracy for interpolation vs. extrapolation. Implicit model maintains similar accuracy across both.

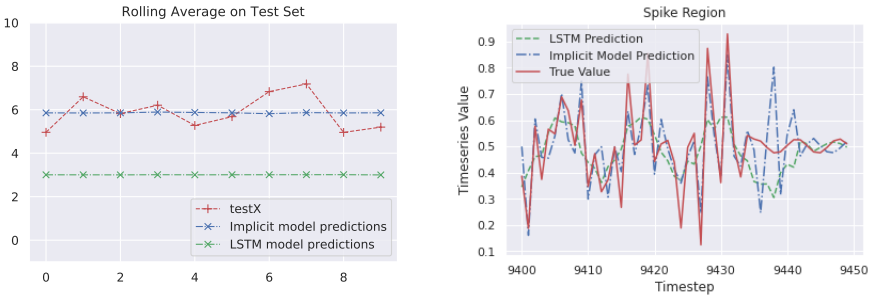

Figure 4: Testing predictions of LSTM and implicit model for rolling average and spiky data. The implicit model learns outside its training distribution whereas the MLP replicates training results.

Table 2: Training and testing error of LSTM and implicit model on sequential data extrapolation tasks. We use specific error metrics for each task: MSE for rolling average, accuracy for rolling argmax and MAE for spiky data predictions.

	Train Error		**Test Error**	
Task	**LSTM**	**Implicit**	**LSTM**	**Implicit**
Rolling Average	0.299	0.181	9.23	0.178
Rolling Argmax	66.3%	86.3%	13.0%	84.2%
Spiky Data Predictions	4.68×10^{-2}	4.95×10^{-3}	5.01×10^{-2}	5.08×10^{-2}

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 algorithm learning and arithmetic tasks. Since implicit models can potentially have infinite depth they have the ability to grow when presented with more complex inputs. Implicit models also require less hyperparameter tuning, specifically regarding activation functions, and therefore appear a better out of the box solution to simple problems as compared to MLPs and LSTMs. 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] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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`.

[16] 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.

A Appendix

A.1 ImplicitRNN.

Implicit recurrent neural networks (ImplicitRNN) receive elements one at a time within a sequence and maintains at least one hidden state to enforce sequence-based memory. We base our ImplicitRN off vanilla RNN to to stay within simple, generalizable models. That is, 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).

A.2 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. Table 3 summarizes the results of 5 of these activation functions on our identity function task as compared to the implicit deep learning model.

Table 3: 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	2.14×10^{-3}	12.4×10^{-1}	21.6	2.16
Leaky ReLU	3.28×10^{-3}	-	22.3	-
Softplus	1.57×10^{-2}	-	17.1	-
Softsign	3.01×10^{-1}	-	47.5	-
Log sigmoid	1.71×10^{-2}	-	17.1	-

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 $y = \sin(x)$.