

# Towards Efficient and Robust Out-of-Distribution Deep Learning with Implicit Models

*Ashwin Ganesh  
Laurent El Ghaoui, Ed.  
Alper Atamtürk, Ed.*



Electrical Engineering and Computer Sciences  
University of California, Berkeley

Technical Report No. UCB/EECS-2023-150

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-150.html>

May 12, 2023

Copyright © 2023, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

---

# Towards Efficient and Robust Out-of-Distribution Deep Learning with Implicit Models

Ashwin Ganesh

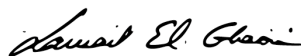
---

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,  
University of California at Berkeley, in partial satisfaction of the requirements for  
the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

### Committee



---

Professor Laurent El Ghaoui  
Research Advisor

5/12/2023

---

(Date)

★ ★ ★ ★ ★ ★ ★



---

Professor Alper Atamtürk  
Second Reader

5/11/2023

---

(Date)

## Acknowledgement

First and foremost, I would like to express my gratitude to my research advisor, Professor Laurent El Ghaoui. He brought me into his team as an undergrad and gave me interesting and high-impact projects to work on from the start. I greatly appreciate him sticking to his commitment to being my research advisor even after getting the exciting opportunity to be the Dean of the College of Engineering and Computer Science of VinUniversity. I would also like to thank Professor Alper Atamtürk for taking time out of his busy schedule to be the second reader for my report.

This paper also would not have been possible without the help of my collaborators. I am deeply grateful to, Alicia Tsai, Prof. El Ghaoui's graduate student who has been an outstanding mentor for me. I could come to her at any time with questions or concerns and she would happily help. She taught me a lot and helped me quell my fears about the paper publishing process. I am also grateful to Juliette Decugis and Max Emerling who worked with me on the mathematical extrapolation project which we published as a NeurIPS workshop paper. They are very brilliant and hard-working and it has been a pleasure working with them over the last few semesters. My paper has intersections with domains such as optimization and seismology that I did not have much knowledge on. I owe my thanks to Wenzhi Gao who provided me with a lot of guidance regarding the theoretical and optimization portions of the feature elimination project. I also owe my thanks to Lindsay Chuang, who went above and beyond to help me on the earthquake location prediction project. She brought me up to speed on the basics of seismology, provided code to generate datasets, and provided lots of feedback on my work.

Finally, I would like to thank my parents, Ganesh and Gayathri, and my sister, Sandhya, for their constant love and support. They have put up with me for so many years and have motivated me to work hard. Last but not least, I would like to thank all my friends, family, and everyone who has supported me throughout my four year journey at UC Berkeley.

---

# Towards Efficient and Robust Out-of-Distribution Deep Learning with Implicit Models

---

Ashwin Ganesh  
UC Berkeley  
aganesh10@berkeley.edu

## Abstract

Deep neural networks excel on a variety of different tasks, often surpassing human abilities. However, when presented with out-of-distribution data, these models tend to break down even on the simplest tasks. This paper compares the robustness of implicitly-defined and classical deep learning models on a series of mathematical tasks and a real-world earthquake location prediction task, where the models are tested with out-of-distribution samples during inference time. Across all experiments, implicit models greatly outperform classical deep learning networks that overfit the training distribution. This paper then shows how to decrease implicit model training time by harnessing the state-driven implicit modeling framework to safely eliminate features while maintaining model accuracy. Safe feature elimination is demonstrated with the FashionMNIST dataset and earthquake location prediction offering a promising avenue for the adoption of implicit models.

## 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, 37]. This work investigates the capability of implicitly-defined neural networks [1, 9, 13] to extrapolate on mathematical tasks.

Implicitly-defined neural networks, such as implicit deep learning [13] 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 state vectors 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  $x = \phi(Ax + Bu)$ , where  $x$  is the equilibrium state for an input  $u$ ,  $\phi$  is a 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, 19, 43]. There has also been emerging work where the equilibrium state is interpreted as a closed-loop feedback system from a neural science perspective [31]. Inspired by these successes, this paper seeks to explore the generalization capabilities of an implicit model when dealing with mathematical and sequential tasks by comparing their capabilities to those of transformers and other architectures specialized for arithmetic computation [42].

It is also important to note that implicit models are slower to train when compared to classical deep learning models. The training time is highly dependent on how many iterations the state-space equation takes to converge for a certain input. The State-driven Implicit Modeling (SIM) paper is able to speed up training time by constraining the state vectors and outputs of the implicit model to that of a pre-trained feed-forward model thus circumnavigating costly implicit differentiation steps in

the backward pass [43]. This work attempts to speed up the training process even further by detailing how one can safely eliminate features when working with the convex and parallelizable training problem outlined by eq. 9 in the SIM paper.

## 2 Related Work

**Out-of-Distribution Extrapolation.** Prior work on using machine learning to solve logical reasoning tasks has primarily focused on developing specialized neural network models to accomplish algorithm learning [17, 18, 25, 30, 40]. Some of these papers take advantage of external memory sources while others allow the network to iterate longer with 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 [30]. Neural Arithmetic Logic Units (NALU) attempt to answer Deep Neural Network’s extrapolation shortcomings on arithmetic tasks. NALU is a novel architecture that is able to explicitly represent mathematical relationships using the neurons of the network [42]. However, even with the improved NALU (iNALU), instability is seen in training and random re-initializations are required [38].

Transformers are well-suited for addition and subtraction tasks achieving very high accuracy on interpolation experiments [34]. However, when faced with Out of Distribution (OOD) data, testing on longer numbers than the model was trained on, only the transformer with larger than three billion parameters performed well [34]. The same OOD concerns are also observed when experimenting on arithmetic tasks with BART, a de-noising auto-encoder using a transformer-based architecture [46]. On the contrary, when transformers were tested on matrix inversion and eigenvalue decomposition, even with OOD data they provided solutions that were “roughly correct” and demonstrated some mathematical understanding [8]. 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.

**Feature Elimination.** The growing size and computational costs of deep neural networks coupled with the push to deploy on edge devices have ushered in a heightened focus on model compression. Many different methods of model compression are being studied such as network pruning [4, 15, 21, 29, 28, 49], lightweight architecture design [35, 27], knowledge distillation [23, 48], and quantization [12]. Amongst these, pruning is the most popular where the goal is to systematically eliminate network structures before [28], during [49], or after training [21] the model.

This paper intends to eliminate features as part of the SIM training problem which can be represented as a constrained LASSO problem [39]. Screening techniques for LASSO have been widely studied. El Ghaoui et. al.’s seminal work describes a pre-training algorithm that allows one to conclusively determine the non-zero entries in the LASSO solution vector thus reducing the problem size and speeding up convergence [16]. These same principles have been extended to many LASSO variants such as group [6], sparse group [32], and generalized [36]. More dynamic variants have since arose, working within iterative solving approaches by leveraging the additional information learned about the optimal solution at each iteration [5, 33]. Screening rules have also been developed for the elastic net formulation [50] which combines the  $\ell_1$  and  $\ell_2$  penalty terms allowing for the combined benefits of group variable selection and shrinkage [20, 47].

## 3 Mathematical Extrapolation

### 3.1 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 from 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) [42] built for out of distribution arithmetic tasks. See Table 1 in the Appendix for model architecture specifics for all tasks.

**ImplicitRNN.** The implicit model may be seen as having an unfair advantage when compared to a traditional recurrent neural networks (RNN) that is only able to look at the input sequence one element at a time. To make a fair comparison, this paper introduces an implicitRNN designed to process sequence elements one at a time and maintain hidden state(s) to enforce sequence-based memory. The linear recurrent layer seen in vanilla RNNs is replaced with an implicit layer with input size  $I + O$  and output size  $O$ , where the output doubles as the recurrent-hidden state.  $I$  will typically be equal to 1. The initial recurrent-hidden state,  $h_0$ , is initialized to be the zero vector. A single forward pass for the  $i$ th sequence element,  $s_i$ , takes the following steps: (1) concatenate  $s_i$  and previous time-step recurrent hidden-state  $y_{i-1} : x_i := [s_i, y_{i-1}]$ ; (2) pass  $x_i$  into the implicit layer, producing output  $y_i$ ; (3) repeat the procedure with  $y_i$  and  $s_{i+1}$ . At any time-step  $i$ ,  $y_i$  can also be used as the output prediction corresponding to the input  $s_i$ .

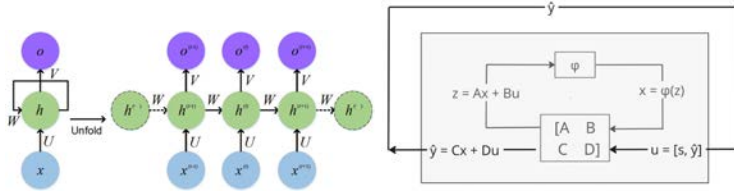


Figure 1: The left panel shows a block diagram of a vanilla RNN (adapted from [14]) and the right panel shows a block diagram of an implicitRNN. This paper’s construction replaces the linear cell present in a vanilla RNN with an implicit layer. Additionally, the implicitRNN does not make a distinction between the output and the recurrent hidden state.

**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 [22, 42]. We train on 10,000 data points 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 points 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.

**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,  $\vec{x} := \langle x_1, x_2, \dots, x_{50} \rangle$ . 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  $t$  ranging from 10 to  $10^5$  symmetrically such that  $x_{\text{test}} \in \mathbb{R}^{50} \sim U(-t/2, t/2)$ .

**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  $x \sim \mathcal{N}(3+t, 1)$ .

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 that process the sequence one timestep at a time: implicitRNN and LSTM. We implement a masked transformer decoder which also only has access to previous inputs at a given timestep. We also compared our results to an unmasked transformer decoder and a regular implicit deep learning model. 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.

### 3.2 Results

For the modeling identity task, Figure 2 shows the MSE (mean squared error) evaluated on the testing set for the MLP, implicit model and transformer. The 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. 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 the 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, the implicit model converged after only 4 iterations when predicting on each test point. It can be concluded 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 3 and Figure 4 compare training and out of distribution testing log MSE loss for the five models. In Figure 4, it is observed that the 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 4, transformers, MLP and implicit models’ log MSE loss seem to similarly linearly increase as the log distribution shift factor increases. Further results in the appendix, Table 3 and Figure 10 however demonstrate the extrapolation advantages of implicit models on even small distribution shifts. Furthermore as suggested by their high training MSE in Figure 3, 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 4 where its testing loss surpasses  $10^{10}$  for an extrapolation shift of only 10. The robust out-of-distribution predictions with the NALU model seen in [42] were not able to be replicated across all the experiments. It is suspected the model’s performance is inflated by hand-crafted evaluation metrics and therefore does not perform well when evaluated using more traditional metrics.

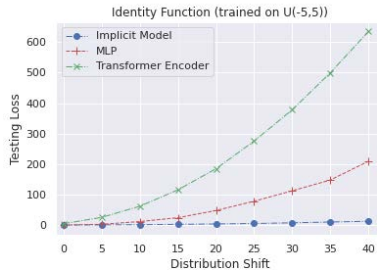


Figure 2: Testing MSE of the MLP, implicit model and transformer encoder evaluated on different testing distribution shifts.

As suggested by Kaiqu Liang et al. [30], 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 it finds a fixed point representation of  $X$  through the equilibrium equation. During training on both arithmetic operations, it was observed that the model failed to converge for at least 1/3 of the 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.

The results on out of distribution inputs for the three sequence modeling tasks are summarized in Figure 5 and 6. For the rolling argmax task, as seen in Figure 5, it is observed that the implicit models maintain the highest and a very stable testing accuracy across distribution shifts. Both the 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 its accuracy drops by almost 50% when evaluated on out of distribution inputs. Figure 6 shows test MSE across distribution shifts for the rolling average task and an example test sequence prediction 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



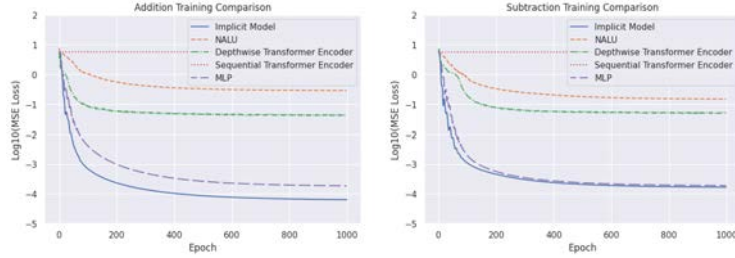


Figure 3: Training Log(MSE) per number of epochs for the five models evaluated on addition and subtraction. The implicit model achieves the lowest training loss across both tasks.

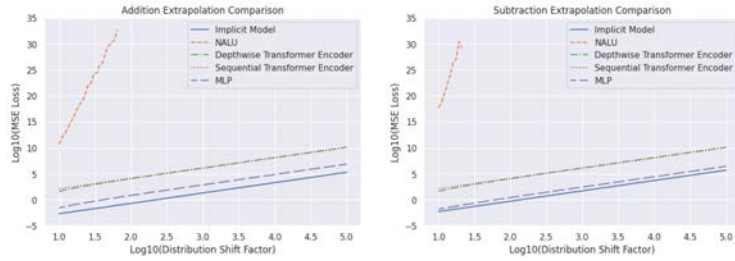


Figure 4: Testing Log(MSE) of the five models evaluated on testing distribution shifts. The implicit model strongly outperforms all other models on OOD data and maintains a linear increase.

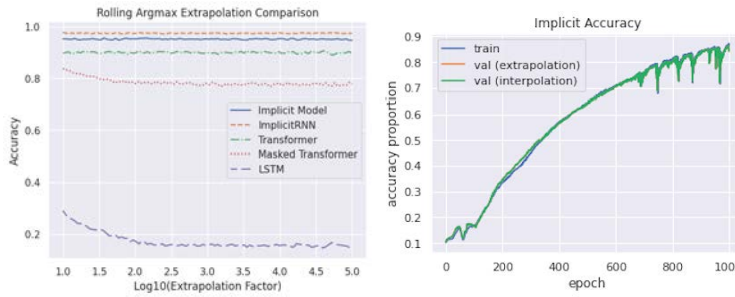


Figure 5: As the extrapolation factor increases the implicit model and implicitRNN are able to maintain superior accuracy on the rolling argmax task. On the right, with extrapolation factor  $t = 10$ , the implicitRNN performs similarly on interpolated and extrapolated data.

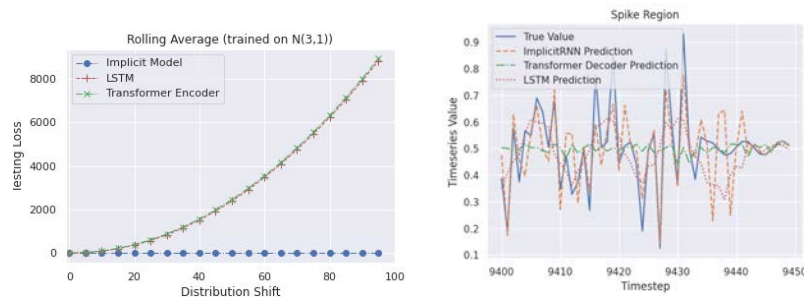


Figure 6: Testing results for the rolling average and spiky time series prediction tasks. For the rolling average, the implicit model maintains close to constant loss across shifts in contrast with the LSTM and transformer. On the right plot, in the spiky regions, the implicit RNN more accurately predicts the magnitude of the spikes.

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, this paper hypothesizes 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.

## 4 Earthquake Location Prediction Extrapolation

### 4.1 Background

The prior experiments solely focused on examining the OOD generalization capabilities of various models when faced with learning mathematical behaviors. The experiments in this section illustrate that the aforementioned OOD extrapolation advantages of implicit models extend to more complex and relevant tasks.

Location prediction is a long-studied problem in the field of seismology. It boils down to predicting the location (longitude, latitude, and depth) and origin time of earthquakes given the observations of stations recording seismic waves. Solving this problem accurately has tremendous implications for early warning systems that use the non-destructive primary (p) waves to predict the origin/impact region of earthquakes. Ascertaining this information even 60-90 seconds before the destructive secondary (s) waves arrive has huge humanitarian implications [11]. These same early warning systems can be used beyond seismic domains for surface explosion monitoring when presented with additional infrasound signals [26]. Location prediction also has huge implications for insurance underwriting, building codes, and policy writing. Seismology, as a field, is embracing deep learning methods which are outperforming traditional methods in many seismological tasks. However, deep learning methods struggle when presented with a sparse number of observations as is the case with early warning system applications [11]. Effective in-distribution location prediction neural networks, such as EikoNet [41], struggle with OOD generalization which will prove detrimental due to the spatial heterogeneity of earthquake data (the majority of earthquakes occur in the Pacific Ring of Fire) [10]. The following experiment results show that general-purpose implicit models can outperform EikoNet when faced with an OOD location prediction task.

### 4.2 Methodology

The data generation and organization follow the methods outlined in [11]. These experiments use synthetically generated data to avoid the bias that creeps in due to the spatial heterogeneity of real-world data. Each sample (subnet) comprises data relating to five stations that record seismic waves. Each subnet contains a randomly selected anchor station which serves as a reference point for all other p-wave arrival times in the subnet. The p-wave travel times were generated using the 1-D velocity model Ak135. The  $x$ ,  $y$ , and  $z$  coordinates of each station, event-station back-azimuths ( $\theta$ ), and relative p-travel times ( $p$ ) w.r.t. to the anchor station are known. All features are linearly scaled to be between -1 and 1 and packed into a feature vector of length 30. The label is the  $X$ ,  $Y$ , and  $Z$  coordinates of the source and the p-wave travel time from the source to the anchor station ( $T$ ). The label is of length 4 and all its components are linearly scaled to be between -1 and 1.

The in-distribution training + validation data is composed of 900,720 samples: 1,295 ( $X$ ,  $Y$ ) source pairs occurring at 695 different depths ( $Z$ ). Training source points were synthetically generated between  $90^\circ\text{E}$  and  $-90^\circ\text{E}$  which roughly corresponds to the boundaries of the Pacific Ring of Fire. Each testing region comprises two 10-degree bands symmetrically located on either side of the training boundary. The first two testing bands border the training boundary. More explicitly, each pair of testing bands takes the following form:  $(90 - 10k, 100 - 10k) \cup (-100 + 10k, -90 + 10k)$  where  $k$ , the extrapolation factor, ranges from 1 to 9. For this experiment, the implicit model was compared to the EikoNet model, a six-layer MLP containing one residual block.

### 4.3 Results

After training, the MLP finished off with a slightly lower in-distribution test loss of  $1.73\text{e}-3$  whereas the implicit model ended with a test loss of  $1.98\text{e}-3$ . As seen in the left panel of Figure 8 as  $k$  increases, the implicit model performs increasingly better when compared to EikoNet in terms of the

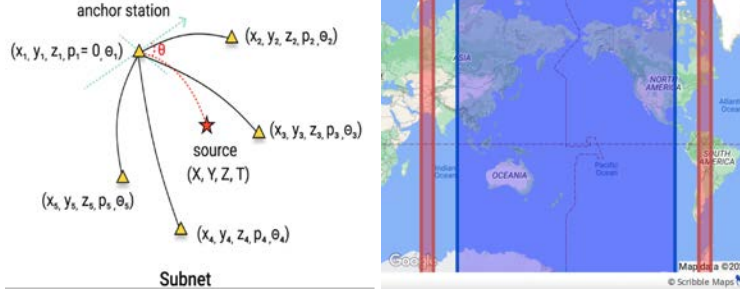


Figure 7: The panel on the left shows a geometric visualization of one set of training features  $(x_i, y_i, z_i, p_i, \theta_i)$  and its corresponding labels  $(X, Y, Z, T)$ . The triangles correspond to stations and the star corresponds to a source. This figure is modified after [11]. The map on the right shows the training set region colored in blue, roughly corresponding to the Pacific Ring of Fire. The area colored in red corresponds to the testing set region when  $k = 3$ .

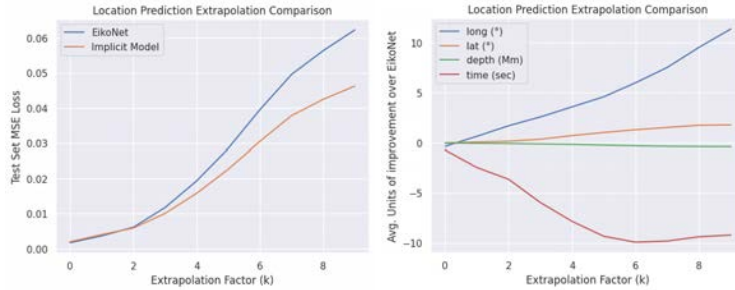


Figure 8: Extrapolation comparison between EikoNet and implicit model on the location prediction task as the extrapolation factor increases. The implicit model has better performance in terms of MSE loss. However, upon closer inspection, it is apparent that the implicit model only outperforms EikoNet in two out of the four categories that make up the MSE loss metric (longitude and latitude).

test set MSE loss. By the time  $k = 2$ , the implicit model has overtaken EikoNet, and when  $k = 9$ , the implicit model’s test set loss is better than that of EikoNet by  $1.59e-2$ . This translates to an average improvement of  $11^\circ$  longitude and  $2^\circ$  of latitude. Future work needs to be done to see whether restricting the source latitude in addition to longitude during training leads to a greater latitude extrapolation improvement. As seen in the right panel of Figure 8, the implicit model struggles a lot more with time and depth prediction performing increasingly worse as  $k$  increases. When  $k = 9$ , the implicit model performs 9.2 seconds and 409 km worse on average. Future work needs to be done to see whether training an implicit model exclusively on time and depth labels improves its performance in those categories. This two pass approach is seen with traditional location prediction software as depth and time are much harder to constrain [44].

## 5 Feature Elimination with State-driven Implicit Models

### 5.1 Background

The relaxed version of the first of the two parallelizable training sub-problems seen in the SIM paper is as follows.  $X$  is the post-activation state matrix of the feed-forward baseline model.  $a^T$  represents a row of  $A$ , one of the implicit model parameter matrices.  $z^T$  is the corresponding row of the pre-activation state matrix [43]. This becomes a simple  $\ell_1$ -norm ball-constrained least squares problem.

$$\min_{\|a\|_1 \leq \kappa} \frac{1}{2} \|Xa - z\|^2 \quad (1)$$

To invoke a modified version of the elastic net formulation, an  $\ell_2$  penalty is added on. Elastic net is known to have performance benefits over LASSO and adds strong convexity to this problem [50].

$$\min_{\|a\|_1 \leq \kappa} \frac{1}{2} \|Xa - z\|^2 + \frac{\gamma}{2} \|a\|^2 \quad (2)$$

After introducing the auxiliary variable  $y = Xa - z$ , the following QP formulation results

$$\begin{aligned} \min_{a, y} \quad & \frac{1}{2} \|y\|^2 + \frac{\gamma}{2} \|a\|^2 \\ \text{subject to} \quad & e^\top t \leq \kappa \\ & t \geq a \geq -t \\ & Xa - z - y = 0. \end{aligned} \quad (3)$$

This paper proceeds by writing the Lagrangian dual. The dual problem is being considered for two reasons. First, the dual problem is of size  $m$ , where  $m$  is the number of features. The primal problem is of size  $n$ , where  $n$  is the number of samples. For most deep learning problems  $m \gg n$ ; therefore, considering the dual reduces the problem size. Additionally due to complementary slackness, solving the dual problem provides more information about redundancy present in the primal variable than solving the primal problem [7].

$$\begin{aligned} L(a, y, t, \mu_1, \mu_2, \nu, \delta) &= \frac{1}{2} \|y\|^2 + \frac{\gamma}{2} \|a\|^2 + \delta(e^\top t - \kappa) \\ &\quad + \mu_1^\top (a - t) - \mu_2^\top (a + t) + \nu^\top (Xa - z - y) \\ &= \frac{1}{2} \|y\|^2 - \nu^\top y + \frac{\gamma}{2} \|a\|^2 \\ &\quad + (\mu_1 - \mu_2 + X^\top \nu)^\top a + (\delta e - \mu_1 - \mu_2)^\top t - z^\top \nu - \kappa \delta \end{aligned}$$

The dual problem follows naturally

$$\begin{aligned} \min_{\mu_1, \mu_2, \nu, \delta} \quad & \frac{1}{2} \|\nu\|^2 + \frac{1}{2\gamma} \|\mu_1 - \mu_2 + X^\top \nu\|^2 + z^\top \nu + \kappa \delta \\ \text{subject to} \quad & \mu_1 + \mu_2 = \delta e \\ & \mu_1, \mu_2 \geq 0. \end{aligned} \quad (4)$$

Assume that  $\mu_1^*$  and  $\mu_2^*$  are known, then by complementarity it is known that

$$\mu_{1,i}^* (t_i^* - a_i^*) = \mu_{2,i}^* (t_i^* + a_i^*) = 0$$

The safe feature elimination condition follows naturally

$$\text{If } \mu_{1,i}^*, \mu_{2,i}^* > 0, \quad \text{then } t_i^* - a_i^* = t_i^* + a_i^* \Rightarrow a_i^* = 0.$$

*Remark 1.* Specifically if  $\gamma = 0$ , then it can be seen that

$$\begin{aligned} \min_{\delta, \mu_1, \mu_2, \nu} \quad & \frac{1}{2} \|\nu\|^2 + z^\top \nu + \kappa \delta \\ \text{subject to} \quad & X^\top \nu + \mu_1 - \mu_2 = 0 \\ & \mu_1 + \mu_2 = \delta e \\ & \mu_1, \mu_2 \geq 0. \end{aligned}$$

and further simplification can be done to obtain an unconstrained dual problem in  $\mathbb{R}^m$ .

$$\min_{\nu} \quad \frac{1}{2} \|\nu\|^2 + z^\top \nu + \kappa \|X^\top \nu\|_\infty \quad (5)$$

## 5.2 Methodology and Results

When solving the dual problem, the non-zero indices of  $\mu_1^*$  and  $\mu_2^*$  correspond to the inactive indices of the primal optimizer ( $a^*$ ). Therefore, these indices also correspond to columns of  $X$  and rows of  $Z$  that can be removed prior to training. This same procedure can be repeated with a reduced size

$X$  and  $Z$  until a pre-defined stopping point. For the purposes of the following experiments, feature elimination halts when  $X$  only has two columns left.

There are multiple methods that can be used to solve the dual problem. For proof of concept, this paper directly solves the dual using the optimization software CVXPY. Unlike the dual problem of LASSO, the dual problems in (4) and (5) are less friendly to solve exactly. Directly solving this dual problem for larger-scale datasets is very time-consuming. However, since the dual solution provides us with more useful information about primal redundancy, it is worthwhile to collect this information and reuse it for solving subsequent problems. A potential solution that balances efficiency and accuracy could solve the first few dual problems exactly using optimization software and then proceed to solve the remaining problems using a first-order heuristic method such as proximal gradient descent.

**FashionMNIST.** Following along with [43], a 4-layer fully-connected network of size  $784 \times 64 \times 32 \times 16 \times 10$  is chosen for constructing the state matrices  $X, Z$  and the outputs  $\hat{Y}$ . The baseline model achieved 82% test performance. The corresponding SIM trained on an  $\ell_1$ -norm objective achieved a test accuracy of 81% using 500 training samples. The bound on the  $\ell_1$  norm of  $a, \kappa$ , can be treated as a hyperparameter; after hyperparameter-tuning  $\kappa = 0.3$  was chosen.

As seen in the left panel of Figure 9, this feature elimination method is very effective. After 50% of the 896 features are eliminated an accuracy drop of less than 1% is observed. A 5% accuracy drop-off only occurs after 95% of the features have been eliminated. This shows that feature elimination can greatly speed up SIM training for a marginal trade-off in accuracy.

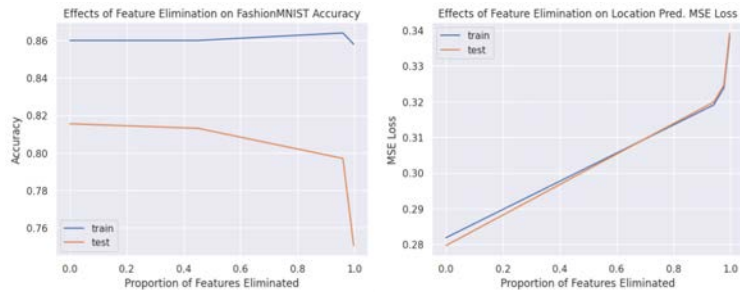


Figure 9: Model performance as a function of the proportion of features eliminated for FashionMNIST classification task and earthquake location prediction. Feature elimination is very effective for the classification task as the elimination of 90% of the model’s features leads to an accuracy drop of approx. 2%. The location prediction results are more modest as 30% feature elimination leads to a  $1.5e-2$  increase in MSE.

**Earthquake Location Prediction.** This section revisits the earthquake location prediction problem from the lens of feature elimination. It considers the scenario where there are observations from many stations and the objective is to decrease training and inference time by only using the observations of a few relevant stations. Synthetic data generation is used to obtain six features each for 45 stations. 4,865 source locations are used in the training set which corresponds to 7  $(X, Y)$  coordinate pairs occurring at 695 depths ( $Z$ ). The baseline EikoNet model achieved a test MSE loss of 0.36 and the baseline implicit model achieved a test MSE loss of 0.27.  $\kappa = 1.0$  is used for this problem and there are 4,750 features prior to feature elimination.

As seen in Figure 9, eliminating 30% of the features leads to a test MSE loss gain of  $1.5e-2$ . As seen earlier, this approximately translates to a loss of  $11^\circ$  longitude and  $2^\circ$  latitude in accuracy. This is a sizeable accuracy drop indicating that this feature elimination method is not as effective for this task. When 90% of the features are eliminated the loss increases by  $4.0e-2$ . Further work needs to be done to see if this procedure can be scaled up to work with a larger training set such as the one used in the prior extrapolation experiments. Currently, it is not possible to chose the number of features that are eliminated with each iteration. Having fine-grained control of this would prove to be very useful for practical applications.

## 6 Conclusion

This paper showcases implicit models' superior performance on out-of-distribution sample points when compared to traditional deep learning models. This ability is apparent in mathematical tasks such as function learning and arithmetic operations but also extends to more challenging applications such as earthquake location prediction. In all experiments conducted, implicit models showcased improved performance over MLPs, LSTMs, transformers, and Google's NALU architecture. The adaptive nature of training implicit models enables them to explore and identify optimal architectures, thereby providing a convenient out-of-the-box solution that offers superior performance on extrapolation tasks. This paper also addresses implicit model training speed concerns by incorporating feature elimination into the state-driven implicit framework. This approach roughly maintains accuracy on small-scale datasets and this paper outlines strategies to extend this approach to larger datasets. These results motivate the further study of implicit models as a robust and practical 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] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Gutttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.
- [5] A. Bonnefoy, V. Emiya, L. Ralaivola, and R. Gribonval. Dynamic screening: Accelerating first-order algorithms for the lasso and group-lasso. *IEEE Transactions on Signal Processing*, 63(19):5121–5132, 2015. doi: 10.1109/TSP.2015.2447503.
- [6] A. Bonnefoy, V. Emiya, L. Ralaivola, and R. Gribonval. Dynamic screening: Accelerating first-order algorithms for the lasso and group-lasso. *IEEE Transactions on Signal Processing*, 63(19):5121–5132, 2015. doi: 10.1109/TSP.2015.2447503.
- [7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511804441.
- [8] F. Charton. What is my math transformer doing? – three results on interpretability and generalization, 2022. URL <https://arxiv.org/abs/2211.00170>.
- [9] 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>.
- [10] L. Y. Chuang, J. Williams, L. Barama, Z. Peng, and A. V. Newman. Teleseismic earthquake phase association via mcmc and deep learning. In *Fall Meeting 2022*. AGU, 2022.
- [11] L. Y. Chuang, J. Williams, L. Barama, Z. Peng, and A. V. Newman. Deep learning and masksembles for sparse network earthquake location and uncertainty estimation. Manuscript in preparation., 2023.
- [12] M. Courbariaux, Y. Bengio, and J. David. Binaryconnect: Training deep neural networks with binary weights during propagations. *CoRR*, abs/1511.00363, 2015. URL <http://arxiv.org/abs/1511.00363>.
- [13] 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>.
- [14] W. Feng, N. Guan, Y. Li, X. Zhang, and Z. Luo. Audio visual speech recognition with multimodal recurrent neural networks. pages 681–688, 05 2017. doi: 10.1109/IJCNN.2017.7965918.
- [15] J. Frankle and M. Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018. URL <http://arxiv.org/abs/1803.03635>.
- [16] L. E. Ghaoui, V. Viallon, and T. Rabbani. Safe feature elimination for the lasso and sparse supervised learning problems, 2011.

- [17] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. URL <http://arxiv.org/abs/1410.5401>.
- [18] 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.
- [19] 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.
- [20] T. Guyard, C. Herzet, and C. Elvira. Screen relax: Accelerating the resolution of elastic-net by safe identification of the solution support, 2022.
- [21] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks, 2015.
- [22] 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.
- [23] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015.
- [24] S. Hochreiter and J. Schmidhuber. Lstm can solve hard long time lag problems. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. URL [https://proceedings.neurips.cc/paper\\_files/paper/1996/file/a4d2f0d23dcc84ce983ff9157f8b7f88-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/1996/file/a4d2f0d23dcc84ce983ff9157f8b7f88-Paper.pdf).
- [25] Kaiser and I. Sutskever. Neural gpu learn algorithms, 2015. URL <https://arxiv.org/abs/1511.08228>.
- [26] C. D. Koch and S. Arrowsmith. Locating surface explosions by combining seismic and infrasound data. *Seismological Research Letters*, 2019.
- [27] B. Koonce. *SqueezeNet*, pages 73–85. Apress, Berkeley, CA, 2021. ISBN 978-1-4842-6168-2. doi: 10.1007/978-1-4842-6168-2\_7. URL [https://doi.org/10.1007/978-1-4842-6168-2\\_7](https://doi.org/10.1007/978-1-4842-6168-2_7).
- [28] N. Lee, T. Ajanthan, and P. H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity, 2019.
- [29] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016. URL <http://arxiv.org/abs/1608.08710>.
- [30] K. Liang, C. Anil, Y. Wu, and R. Grosse. Out-of-distribution generalization with deep equilibrium models.
- [31] 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.
- [32] E. Ndiaye, O. Fercoq, A. Gramfort, and J. Salmon. Gap safe screening rules for sparse-group lasso. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL [https://proceedings.neurips.cc/paper\\_files/paper/2016/file/555d6702c950ecb729a966504af0a635-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2016/file/555d6702c950ecb729a966504af0a635-Paper.pdf).
- [33] E. Ndiaye, O. Fercoq, Alex, re Gramfort, and J. Salmon. Gap safe screening rules for sparsity enforcing penalties. *Journal of Machine Learning Research*, 18(128):1–33, 2017. URL <http://jmlr.org/papers/v18/16-577.html>.
- [34] 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>.



- [35] S. Qian, C. Ning, and Y. Hu. Mobilenetv3 for image classification. In *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, pages 490–497, 2021. doi: 10.1109/ICBAIE52039.2021.9389905.
- [36] S. Ren, S. Huang, J. Ye, and X. Qian. Safe feature screening for generalized lasso. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2992–3006, 2017.
- [37] 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>.
- [38] 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>.
- [39] M. Schmidt. Least squares optimization with  $l_1$  norm regularization. 01 2005.
- [40] 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>.
- [41] J. D. Smith, K. Azizzadenesheli, and Z. E. Ross. EikoNet: Solving the eikonal equation with deep neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 59(12):10685–10696, dec 2021. doi: 10.1109/tgrs.2020.3039165. URL <https://doi.org/10.1109/TGRS.2020.3039165>.
- [42] 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>.
- [43] 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.
- [44] USGS. Hypoinverse earthquake location. URL <https://www.usgs.gov/software/hypoinverse-earthquake-location>.
- [45] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017. URL [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf).
- [46] 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.
- [47] Y. Xu, Y. Tian, X. Pan, and H. Wang. E-endpp: a safe feature selection rule for speeding up elastic net. *Applied Intelligence*, 49:592–604, 2019.
- [48] L. Zhang, C. Bao, and K. Ma. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4388–4403, 2021.
- [49] M. Zhu and S. Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression, 2017.
- [50] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005. ISSN 13697412, 14679868. URL <http://www.jstor.org/stable/3647580>.

## A Appendix

### A.1 Model and Task Specifications

For all the extrapolation experiments, this paper compares the extrapolation capacities of a baseline model (MLP for non-sequential tasks, LSTM [24] for sequential tasks, and NALU for arithmetic operations), implicit models (standard and RNN), and Transformers [45]. The architectures used for each task are summarized in Table 1.

### A.2 Activation Function Experiments

For the identity function and arithmetic operation tasks, this paper experiments with 15 different activation functions on the MLP: hardtanh, sigmoid, reLU6, tanh, tanhshrink, hardshrink, leakyrelu, softshrink, softsign, reLU, preLU, multipreLU, softplus, eLU and seLU. The goal is to understand whether specific activations helped the MLP extrapolate as well as the implicit model. Table 2 summarizes the results of 5 of these activation functions on the identity function task as compared to the implicit deep learning model.

Table 3 compares the test MSE for the MLP with ReLU activation, the best MLP across all 15 activations and the implicit model. For this experiment,  $x_{\text{train}} \in \mathbb{R}^{100} \sim U(1, 2)$  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.

### A.3 Arithmetic Operations More Results

For more specific results on the OOD generalization capacities of implicit models, Figure 10 compares 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 a small distribution shifts since  $t = 3$ .

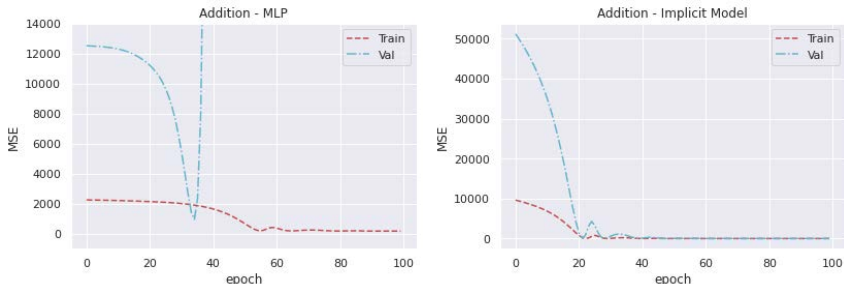


Figure 10: Testing and training MSE plots based on the number of training epochs for the addition and rolling argmax tasks. The MLP test loss bounces from low to high and eventually explodes whereas the implicit model achieves testing loss close to 0.

### A.4 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  $y = \sin(x)$ .

Table 1: Details of the network architectures used in this paper’s experiments and applications. The number of parameters in each model is in parentheses next to its architecture info.

Task	Dataset/Task summary	Baseline model(s)	Implicit model(s)	Transformer(s)
Identity Function	$x_{\text{train}} \in \mathbb{R}^{10} \sim U(-5, 5)$  $x_{\text{test}} \in \mathbb{R}^{10} \sim U(-\kappa, \kappa)$ , where $\kappa$ ranges from 10 to 80	MLP: $10 \times 10 \times 10$ (220)	Regular: $A \in \mathbb{R}^{4 \times 4}, B \in \mathbb{R}^{4 \times 10}, C \in \mathbb{R}^{10 \times 4}, D \in \mathbb{R}^{10 \times 10}$ (196)	Encoder: Single attention head (43,498)
Arithmetic Operations	$x_{\text{train}} \in \mathbb{R}^{50} \sim U(-1, 1)$  $x_{\text{test}} \in \mathbb{R}^{50} \sim U(-t/2, t/2)$ , $t$ ranges from 10 to $10^5$  Predict the sum or difference of 8 random numbers out of 50	MLP: $50 \times 10 \times 10 \times 1$ (1,497)  NALU: $50 \times 10 \times 10 \times 1$ (1,530)	Regular: $A \in \mathbb{R}^{20 \times 20}, B \in \mathbb{R}^{20 \times 50}, C \in \mathbb{R}^{1 \times 20}, D \in \mathbb{R}^{1 \times 50}$ (1,470)	Sequential encoder: 1 layer, 10 attention heads - processes each array as a single sequence (6,208)  Depth-wise encoder: 1 layer, 1 attention head - processes each element in a given array as a single sequence (217,349)
Rolling Average	$x_{\text{train}} \sim \mathcal{N}(3, 1)$  $x_{\text{test}} \sim \mathcal{N}(3 + t, 1)$ , $t$ ranges from 5 to 100	LSTM: $1 \times 100 \times 100 \times 1$ (42,210)	Regular: $A \in \mathbb{R}^{200 \times 200}, B \in \mathbb{R}^{200 \times 10}, C \in \mathbb{R}^{10 \times 200}, D \in \mathbb{R}^{10 \times 10}$ (44,100)	Encoder: Single layer and 2 attention heads, (42,210)
Rolling Argmax	$x_{\text{train}} \sim U(0, 1)$  $x_{\text{test}} \sim U(0, t)$ , $t$ ranges from $10^1$ to $10^5$	LSTM: $1 \times 19 \times 19 \times 10$ (1,872)	Regular: $A \in \mathbb{R}^{33 \times 33}, B \in \mathbb{R}^{33 \times 10}, C \in \mathbb{R}^{10 \times 33}, D \in \mathbb{R}^{10 \times 10}$ (1,849)  RNN: $A \in \mathbb{R}^{18 \times 18}, B \in \mathbb{R}^{18 \times 23}, C \in \mathbb{R}^{22 \times 18}, D \in \mathbb{R}^{22 \times 23}$ (1,870)	Masked decoder: 1 layer, 2 attention heads (1,920)  Unmasked decoder: 1 layer, 2 attention heads (1,920)
Spiky Time Series	Predict rare volatile patterns (see A.4 for more)	LSTM: $1 \times 20 \times 20 \times 1$ (1,861)	RNN: $A \in \mathbb{R}^{20 \times 20}, B \in \mathbb{R}^{20 \times 21}, C \in \mathbb{R}^{20 \times 20}, D \in \mathbb{R}^{20 \times 21}$ with a linear layer $20 \times 1$ (1,661)	Masked decoder: 1 layer, 10 attention heads (43,529)
Earthquake Location Prediction	Train: 720,576 ( $X, Y, Z$ ) locations sampled between $(90, -90)^\circ\text{E}$ , 30 features  Test: 20,016 samples in each extrap. region $(90 - 10k, 100 - 10k) \cup (-100 + 10k, -90 + 10k)$ , $k$ ranges from 1 to 9	EikoNet: $270 \times 32 \times 128 \times 128 \times 128 \times 32 \times 4$ (42,500)	Regular: $A \in \mathbb{R}^{190 \times 190}, B \in \mathbb{R}^{190 \times 270}, C \in \mathbb{R}^{4 \times 190}, D \in \mathbb{R}^{4 \times 270}$ (42,680)	N/A

Table 2: Testing loss of the implicit model and five MLP models with specific activations on the identity function task. The results show that the implicit model outperforms the MLP across activation functions. Description of the activation functions in the appendix.

Activation	Train MSE		Test MSE	
	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 3: Test MSE table of two MLPs and the 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	$3.69 \times 10^{19}$	$1.87 \times 10^4$	$3.40 \times 10^{-2}$