



City Research Online

City St George's, University of London

Citation: Lopedoto, E., Weyde, T. & Salako, K. (2024). DATE: Derivative Alignment Training for Extrapolation with Neural Networks. Paper presented at the SGAI BCS AI2024, 17-19 Dec 2024, Cambridge.

This is the accepted version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: <https://openaccess.city.ac.uk/id/eprint/33866/>

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to [City Research Online policy](#).

DATE: Derivative Alignment Training for Extrapolation with Neural Networks

Enrico Lopedoto, Tillman Weyde, and Kizito Salako

City, University of London
Department of Computer Science
Northampton Square, London, UK
{enrico.lopedoto,t.e.veyde,k.o.salako}@city.ac.uk

Abstract. In this work we introduce *DATE* (Derivative Alignment Training for Extrapolation), a method to improve the extrapolation behaviour of neural networks (NN) with Rectified Linear Unit activation (ReLU) on univariate regression tasks. ReLU NNs naturally lend themselves to linear extrapolation beyond the training data range. However, there are two known limitations of extrapolation properties of trained ReLU NNs, that we address in this paper. When minimising the error of the prediction, the derivative of the NN model function can still show high variation, which can cause variable extrapolation. Non-linearities of the model function outside the training data range can lead to inconsistent extrapolation behaviour. In prior work, these issues have been addressed with a set of regularisation functions, called *ReLEx*. To address the issues named, we introduce two new regularisation terms: the *D1*-loss and the *IR*-loss. *D1*-loss directly penalises the deviation of the model derivative from a target derivative estimated from the data as the interpolation between neighbouring data points. The *IR*-loss penalises positions of the non-linearities of the ReLU units outside a given range. Optimising the combination of *D1* with *IR* loss and/or some of the *ReLEx* functions constitutes the *DATE* method. We evaluate *DATE* on regression tasks with noiseless data generated from analytic functions. We test different *DATE* configurations and find that training with *DATE* can reduce the variability of the model slope, prevent non-linearities outside the training data range, and improve extrapolation consistency as measured by different metrics.

Keywords: Extrapolation · Regression · Neural Networks · Derivative.

1 Introduction

In supervised machine learning the immediate objective is to fit the model function to the training data by minimising a loss function. This loss function is typically defined in terms of the values of the data and the model, and for regression the error metric is typically the mean squared error (MSE). The most popular machine learning models in recent years have been neural networks

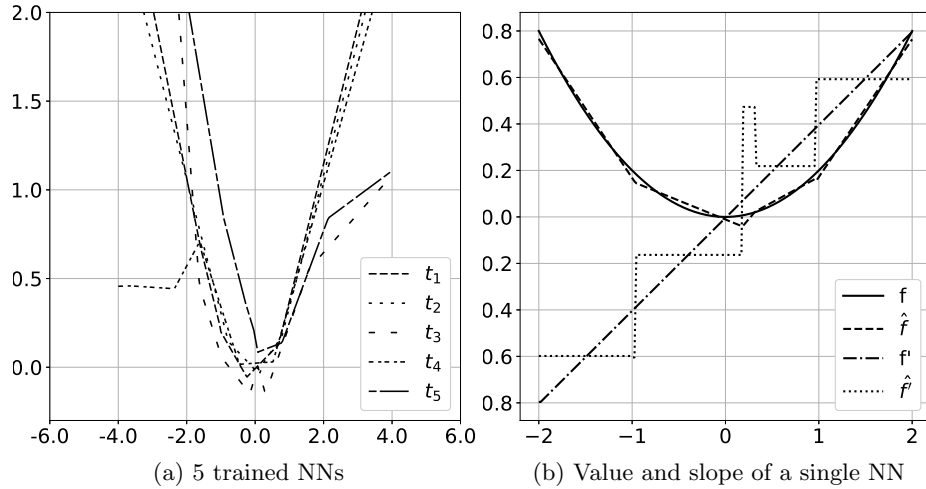


Fig. 1: Prediction of quadratic function with data in the range $[-2,2]$ with a ReLU NN trained with MSE loss only. We display the true function (solid), the model function (dashed), the slope of the true function in (dashdot) and the slope of the model function (dotted).

(NN). In particular, NNs with rectified linear unit activation (ReLU) have desirable properties in terms of computational efficiency and conceptual simplicity so that ReLU has become the standard activation function [6].

Here, we are concerned with the extrapolation of the model beyond the range of the training data. We explore specifically linear extrapolation behaviour, which is made possible by the ReLU activation function as opposed to previously more common sigmoid functions (logistic function and tanh). When training ReLU NNs to minimise the MSE, there are two problems that can be observed in Figure 1: (a) shows predictions inside and outside of the training data range $[-2,2]$ of five separately trained NNs with one hidden layer of 10 neurons, and we observe a wide variation of slopes outside the training data range, at least for some models. (b) shows the value and slope of the model function vs value and slope of the target function in the training data range, where we observe that the the slope of the model function deviates in most cases far from the true slope value especially in the central segments. A solution to the issue in (a) has been proposed in [10], however, it has been observed empirically that the regularisation proposed increases the computational complexity of the training because of the double pass in calculating the derivatives, and the issue in (b) has, to our knowledge, not been addressed at all in the context of extrapolation, yet.

The goal of this work is to take the opportunity that arises from the use of ReLU activation and devise a method that addresses these issues and achieves desirable linear extrapolation behaviour efficiently in standard ReLU neural net-

works in a way that requires minimal changes from standard neural network learning.

Our contributions are as follows: 1) we introduce improved extrapolation of NN models through derivative alignment; 2) we propose *DATE*, a new method to improve extrapolation in regression task with ReLU neural networks including two new regularisation terms, $\mathcal{L}_{D1}$ and $\mathcal{L}_{IR}$; and 3) we provide an implementation of *DATE* in PyTorch and evaluation on the extrapolation of five different target functions with simple NNs.

2 Related Work

In supervised machine learning, over the last decade, neural networks (NN) have been adopted for many machine learning problems [6,4]. However, neural networks display differences in behaviour when predicting a function between data points and predicting a function for points outside the range of the training data. We refer to prediction in the training data range as *interpolation* and outside the training data range as *extrapolation* see, e.g., [1]. Some methods to address the extrapolation problems for mathematical functions have been proposed by [15] and [12], who introduce non-standard NN models with activation functions such as identity and multiplication functions and achieve improved extrapolation.

A lack of extrapolation has also been observed in the context of linguistic structures, where rules are not generalised between different words or syllables [7,16,8,14], or when we need to extrapolate from even to odd numbers [13], or when extrapolating to Dyck language recognition to long sequences [3].

2.1 Extrapolation with ReLU NNs

The rectified linear unit (*ReLU*) function is the most popular activation function in modern neural networks [6], firstly introduced by [5] to the best of our knowledge. The ReLU activation offers the possibility of performing unbounded linear extrapolation that is not possible with bounded activation function like the logistic sigmoid or *tanh* functions. However, NNs trained with MSE using ReLU show high variability in the extrapolation, as illustrated above.

There is no single ideal extrapolation behaviour in the absence of a known function outside the training data range. As we have decided to limit ourselves to linear extrapolation, a natural target is the tangent line at the last data point. In addition, there are considerations with respect to the learning process of NNs, where we adopt the desiderata from [10]:

- the NN should learn how to extrapolate from the training data;
- the NN should adapt the learning process making as little changes to the network structure, activation functions and training process as possible;
- the implementation should make it easy for practitioners to replicate and be used.

We also adopt the design decision of [10] to address the problem by changing only the loss functions and not the structure of the NN itself.

2.2 Simple ReLU NNs and *dying ReLUs*

In the remainder of this paper we consider a simple feed-forward ReLU neural network with a single hidden layer (consisting of H neurons) that calculates the function $\hat{y} = f(x, \beta) = \sum_{h=1}^H w_o \cdot \text{ReLU}(w_h x + b_h) + b_o$, where w_h and b_h are the weight and bias value of hidden neuron h , respectively. The weight between hidden neuron h and the output neuron is w_o , while b_o is the bias of the output neuron. All of these weights and biases make up the vector β .

The ReLU function, $\text{ReLU} : \mathbb{R} \rightarrow \mathbb{R} = \max(x, 0)$, has two parts, for negative arguments the output is constant 0, we call this the *constant part* of the ReLU. For positive inputs x , the output is x and we call this the *linear part* of the ReLU. At input 0, the ReLU is not differentiable, this is often called the *non-linearity* of the ReLU or the *0-point* as defined below. A known problem in NN learning is a *dying ReLU* [2,11]. This refers to a situation where all data points lead to a negative input to the ReLU function, so that it operates in its constant part, its output is 0, and its derivative is 0. In this situation, the input weights of that neuron will not be changed during learning. The linear part of that neuron will therefore not depend on the data but only on the initialisation of the weights. Examples of this are the observed variation of slopes outside the data range in Figure 1(a), where we plot the output over the range $[-8, 8]$ of the 5 NN training runs with different random initialisation. We use 200 data points sampled from the target function $f(x) = \frac{1}{5}x^2$ in the range $[-2, 2]$ and train for 700 epochs with a mini-batch size of 20 on a feed forward neural network with a single hidden layer with 10 neurons. In (b), we plot the output value and slope of a single trained NN and those of the target function.

2.3 *ReLEx* and *0-points*

ReLEx, introduced in [10], is a method that aims to control extrapolation behaviour by introducing additional loss functions, which mainly control the position and orientation of the ReLU function in the hidden neurons. The results show that the usage of the proposed loss functions improves the extrapolation behaviour, but the method has limitations and requires tuning of several parameters. In addition, these losses increase the computational complexity to be quadratic in the number of hidden neurons.

The *ReLEx* method, is focused on *0-points*, i.e., the points in the input space where the ReLU's constant and linear parts meet and where the ReLU function is non-linear. Formally, the *0-points* are the input values x_{h0} that lead to input 0 for a hidden neuron h : $0 = w_h \cdot x_{h0} + b_h$, so that the *0-point* for hidden neuron h is $x_{h0} = -b_h/w_h$.

The *ReLEx* regularisation introduces four individual loss terms $\mathcal{L}_l$ with associated scaling parameters θ_l . They serve different purposes which we describe here briefly. The formal definitions are listed in Table 1. The **Centripetal** loss ($\mathcal{L}_{CP}$) pulls all the *0-points* inside the training data range; the **Mutually Repellent** loss ($\mathcal{L}_{MR}$) is responsible for counter balancing the $\mathcal{L}_{CP}$ force by distributing the *0-points* inside the training data range; the **Weight Orientation**

Loss Name	Symbol	Definition	Complexity
Centripetal	$\mathcal{L}_{CP}$	$\sum_{h=1}^H \sum_{k=1}^K (x_{h0} - x_k)^2$	$O(HK)$
Mutually Repellent	$\mathcal{L}_{MR}$	$\sum_{h=1}^H \sum_{j>i}^H \frac{1}{(x_{h0} - x_{j0})^{2+\varepsilon}}$	$O(H^2)$
Weight Orientation	$\mathcal{L}_{WO}$	$-\sum_{h=1}^H \text{ReLU}((x_{h0} - \bar{x}) \cdot w_h + \varepsilon)$	$O(H)$
Weight Sign	$\mathcal{L}_{WS}$	$(\sum_{h=1}^H w_h)^2$	$O(H)$

Table 1: The loss terms used to improve extrapolation. K is the number of data points. H is the number of hidden neurons. x_{h0} is the θ -point of hidden neuron h (dependent only on the network weights w_h and b_h). $\bar{x}$ is the average of the training data.

loss ($\mathcal{L}_{WO}$) moves the θ -point for each ReLU so that the constant part covers most of the data points and only data points close to the margins of the training data range determine the slope of the extrapolation; lastly, the **Weight Sign** loss ($\mathcal{L}_{WS}$) recovers from the degenerate case when all the weights have the same sign (occurring on average in only $\frac{1}{2^{H-1}}$ of the cases given the random initialisation of a network with H hidden neurons).

3 DATE — Derivative Alignment Training for Extrapolation

In this work, we explore a new approach to improve extrapolation, which involves directly controlling the slope of the model function, and also a new approach to avoid θ -points outside the data range. This helps to solve the two issues of NNs for extrapolation illustrated previously (Figure 1). To avoid the θ -points outside the training data range, i.e. *dying ReLUs*, as shown in 1 (a), we introduce a new efficient loss term $\mathcal{L}_{IR}$. To improve the alignment of the slope of the target and model, as shown in 1 (b), we introduce a new loss term $\mathcal{L}_{D1}$. Optimising the combination of $D1$ and IR loss constitute the *DATE* method.

3.1 $\mathcal{L}_{D1}$

The concept of derivative alignment has been introduced in [9] with a special regularisation term, the *DLoss*. Here, we introduce $D1$ -loss $\mathcal{L}_{D1}$, a univariate version of *DLoss*. Like *DLoss*, $\mathcal{L}_{D1}$, penalises the deviation of the model’s derivative from the target function derivative estimated from the training data points. More formally, it is the sum of the squared differences between the model derivative, $\hat{f}'$ – the first derivative of the model function $\hat{f}$ – and an estimate of the target derivative, y' , based on the training data.

While *DLoss* uses sampling, as in the multi-dimensional case there is no total order of the input data, in the univariate real case, so we can define the data vector $XY = [(x_1, y_1), \dots, (x_K, y_K)]$ which is ordered by x .

Then, we estimate the first derivative of the target function y' at $k-1$ points x' as the vector of pairs

$$XY' := [(x'_1, y'_1), \dots, (x'_{k-1}, y'_{k-1})] \quad (1)$$

where

$$(x'_k, y'_k) = \left((x_{k+1} + x_k)/2, (y_{k+1} - y_k)/(x_{k+1} - x_k) \right), \quad (2)$$

i.e., we take y'_k , the slope of the line between two data consecutive points, as the derivative value for argument x'_k , the mid-point between two consecutive x -values.

Without additional assumptions, we do not know anything about the true target function except for the provided data and, thus, cannot give any guarantees on how far the calculated derivative may be from the true derivative. However, if the true function f is differentiable, we do know that y'_{k+1} is the derivative value $f'(x)$ for some $x \in [x_k, x_{k+1}]$ by the mean value theorem.

We calculate the model derivative $\hat{f}'$ using the finite difference method with a parameter ε as

$$\hat{f}'(x) = \frac{\hat{f}(x + \varepsilon) - \hat{f}(x - \varepsilon)}{2\varepsilon}. \quad (3)$$

We use $\varepsilon = 0.05$ as it is large enough to avoid numerical issues and suitable for the curvature of the functions that we use in our experiments here. $\mathcal{L}_{D1}$ is then defined as

$$\mathcal{L}_{D1} = \sum_{(x', y') \in XY'} (y' - \hat{f}'(x'))^2 \quad (4)$$

and with it we are able to approximate the local slope of the model function to the local slope of the data with complexity of $O(HK)$.

3.2 $\mathcal{L}_{IR}$

In Figure 1 (a) we saw undesired behaviour outside the data range. We are interested in making sure that at least two data points are covered by the linear part of every ReLU, so that the slope of the model in the extrapolation range will be determined by the data.

To do so, in *ReLU*, $\mathcal{L}_{CP}$ tries to pull the 0-points inside the training range. For higher θ_{CP} values, however, we have observed that all 0-points cluster around the data mean. To counterbalance this effect $\mathcal{L}_{MR}$ was introduced to equally distribute the 0-points over the range.

To avoid 0-points outside the training data range, we introduce the Internal Range loss $\mathcal{L}_{IR}$ defined as:

$$\mathcal{L}_{IR} = \sum_{k=1}^K \left(ReLU(-x_{k0} + L - \tau) + ReLU(x_{k0} - R + \tau) \right)^2, \quad (5)$$

where K is the number of hidden neurons, L and R are the most left point and right point of the training data range respectively and τ defines the margin from

where data points within the training data range are covered. This loss pulls the 0-points outside the training data range toward the inside with margin τ .

Only hidden neurons where the 0-points fall outside of the training data range with a margin τ are affected by $\mathcal{L}_{IR}$ as opposed to $\mathcal{L}_{CP}$, where stronger weight drives 0-points toward the mean of the data set and $\mathcal{L}_{MR}$ distributes them.

Another benefit of using $\mathcal{L}_{IR}$ is that it has lower computational complexity $-O(K)$ than $\mathcal{L}_{CP} + \mathcal{L}_{MR} - O(H^2)$. The complexity of $\mathcal{L}_{CP}$ is $O(HK)$ (with a possibility to be reduced to $O(H)$ by using the mean of the data instead of the individual points) and $\mathcal{L}_{MR}$ complexity is $O(H^2)$.

4 Experiments

We evaluate the effect of three different *DATE* configurations by training a small ReLU NN on data sampled from analytical functions. The baselines are standard NNs with MSE and *ReLU* losses. $MSE = \frac{1}{n} \sum_{y \in Y} (y - \hat{y})^2$. We run different experiments with settings described below.

Method. We chose ReLU-based NN with a single input, single output, and one hidden linear layer with 10 neurons. To train it we use Adam optimiser.

Data generation. For this paper, we only consider noise-free data points sampled from a specific mathematical function $f(x)$. The training data range of the input x is $[-2, 2]$ with a fixed grid of 200 equidistant points. The prediction range is $[-10, 10]$ with another 200 points. We use different function families to include linear (*lin* $f(x) = x$, *abs* $f(x) = |x|$), with curvature (*pow* $f(x) = \frac{1}{5}x^2$, *sigm* $f(x) = 1/(1 + e^{-x})$), and periodic (*sin* $f(x) = \sin(x)$).

Hyper-parameters. We use the following hyper-parameters for the training: learning rate $\epsilon = 0.004$; momentum $\nu = 0.9$; and 700 epochs with mini-batch size 20. These parameters have shown in preliminary experiments to reliably achieve convergence without major instabilities during the training process. We varied $\theta_{D1} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ to explore the effect of *DATE*. For the other losses the parameters that we use in the experiments are those resulting from the [10] being $\theta_{CP} = 0.3$, $\theta_{MR} = 10$, $\theta_{WO} = 0.3$ and $\theta_{WS} = 0.1$, $\theta_{IR} = 5$.

4.1 Evaluation metrics

Interpolation metrics are calculated on the training data range $[-2, 2]$ where we use 200 equidistant samples:

- MSE_{int} : MSE of prediction against true value on a 200 feature sample from a uniform distribution in the training data range;
- MSE_{D1} : MSE of model derivative against f' on a 200 feature sample from a uniform distribution in the training data range.

Extrapolation metrics are calculated on the *extrapolation set*, which consists of 200 feature points sampled from a uniform distribution in the left $[-10, -2]$ and right $[2, 10]$ extrapolation range. We calculate these metrics individually for the right and left extrapolation set and we aggregate them as an average.

A different amount of data at the margins carry a different amount of information and define how local the trend will be that we extrapolate. Less local extrapolations will be more robust against noise in the data, but further away from the tangent at the last data point. Therefore we have a spectrum of targets to measure the precision of extrapolation. This is reflected in the different metrics calculated on the extrapolation data range:

- Tan_{ex} : MSE of prediction against the tangent lines to the true function at the margins of the training data range.
- WLR_{ex} : MSE of model prediction against a weighted linear regression (hence WLR) done on which points. The WLR is a regression where we assign a different weight to each input. In this way, we can over-weight data closer to the margins of the training data range. The weighting scheme used is as per [10] section 5.1, defined by a exponential function which allocates weights rising exponentially toward the margin L and R of the training data range. The weights are then normalised so that they sum up to 1.
- $D1_{ex}$: MSE of the model prediction against the line of the last two point with slopes y'_1 for and y'_{k-1} .

The metrics reported are the average values obtained from 10 runs for each experiment. In addition, we measure extrapolation consistency with the standard deviation of the slope of the linear regression on the extrapolation set to the NN predictions. We do this, as before, on both the negative and positive extrapolation ranges.

- σ_S : is the extrapolation variability over 10 runs. It is calculated as the standard deviation of the slope resulting from the linear fitting over the predicted values in the portion S of the extrapolation range, i.e. $[-10, -2]$ for left and $[2, 10]$ for right.

4.2 DATE Configurations

We compare different configurations of *DATE* in order to evaluate the effects of $\mathcal{L}_{D1}$ and $\mathcal{L}_{IR}$. We test MSE and *ReLEx* and three other different variants of *DATE*. Below we describe each configuration.

*DATE*₁: *Full ReLEx* + $\mathcal{L}_{D1}$. Here, we add $\mathcal{L}_{D1}$ to all *ReLEx* loss terms. We hypothesise that $\mathcal{L}_{D1}$ will improve the derivative alignment and, thus, lead to better extrapolation. The total loss function $\mathcal{L}$ is defined as:

$$\mathcal{L} = ReLEx + \theta_{D1} \mathcal{L}_{D1} + MSE \quad (6)$$

*DATE*₂: $\mathcal{L}_{WO} + \mathcal{L}_{WS} + \mathcal{L}_{IR} + \mathcal{L}_{D1}$ In this loss configuration, we replace the $\mathcal{L}_{CP}$ and $\mathcal{L}_{MR}$ terms that are part of *ReLEx* with $\mathcal{L}_{IR}$. We hypothesise that this configuration should improve the adaptability within the training data range compared to the *ReLEx* and *DATE*₁. The total loss function $\mathcal{L}$ is defined as:

$$\mathcal{L} = \theta_{WO} \mathcal{L}_{WO} + \theta_{WS} \mathcal{L}_{WS} + \theta_{IR} \mathcal{L}_{IR} + \theta_{D1} \mathcal{L}_{D1} + MSE \quad (7)$$

$DATE_3: \mathcal{L}_{WS} + \mathcal{L}_{IR} + \mathcal{L}_{D1}$ We hypothesise that $\mathcal{L}_{WO}$ might not be needed, as we do not have a strong theoretical justification for it. The total loss function $\mathcal{L}$ is defined as:

$$\mathcal{L} = \theta_{WS} \mathcal{L}_{WS} + \theta_{IR} \mathcal{L}_{IR} + \theta_{D1} \mathcal{L}_{D1} + MSE \quad (8)$$

While $DATE_1$ includes $\mathcal{L}_{MR}$, its complexity is quadratic in terms of H , $O(H^2)$, $DATE_2$ and $DATE_3$ includes $\mathcal{L}_{IR}$, with complexity $O(H)$, making these last two alternatives computationally cheaper.

5 Results

We report the metrics as explained in previous section when applying different loss configurations in Table 2. We show how each individual analytical function behaves with the best parameters in Table 3.

In Table 2, we report results depending on different θ_{D1} for $\mathcal{L}_{D1}$. $DATE_2$ is the configuration which offers the best results for MSE_{int} and majority of the extrapolation metrics. Using $DATE_1$ provides the best result w.r.t. WLR_{ex} and $DATE_3$ provide the best MSE_{D1} result.

We have observed that the metrics reported for the non-sine functions group are lower than the ones for sine. For both groups we still observe that $DATE$ configurations achieve lowest MSE_{int} and MSE_{D1} compared to MSE and $ReLU$. In terms of extrapolation, while $DATE_2$ and $DATE_1$ achieve best results w.r.t. Tan_{ex} and $D1_{ex}$ for non-sine functions group and sine respectively. $ReLU$ still achieves the lowest error for extrapolation w.r.t. WLR_{ex} for both groups.

In Table 3 we report results for each individual function obtained with the best θ_{D1} configuration for each function – selected to achieve the lowest MSE_{D1} and make all the 0-points are inside the data range. We observe how linear function lin and abs achieve best results across the interpolation metrics and Tan_{ex} and $D1_{ex}$ with $DATE_1$. For pow , $sigm$ and sin the best results for interpolation, Tan_{ex} and $D1_{ex}$ are obtained with $DATE_2$ and $DATE_3$. WLR_{ex} remains the metric to be minimised by $ReLU$.

6 Discussion

With $DATE$ approach, MSE_{int} is in most cases improved. In $DATE_2$ and $DATE_3$, $\mathcal{L}_{IR}$ is responsible for pulling 0 -points toward the endpoints of the training data range. We observe that using $\mathcal{L}_{IR}$ is at least as good as using $\mathcal{L}_{CP} + \mathcal{L}_{MR}$. In $DATE_2$, the $\mathcal{L}_{WO}$ is apparently still useful because results show that having an influence on the linear part of ReLU contributes to having a less variable extrapolation.

Depending on the portion of the data range close to the margins that we consider local, we can arbitrarily decide how much influence this can have on the extrapolation. Tan_{ex} and WLR_{ex} are lower than for MSE and $ReLU$ training for all functions considered. The more data are considered, i.e., the less local to

Loss	θ_{D1}	MSE_{int}	MSE_{D1}	Tan_{ex}	WLR_{ex}	$D1_{ex}$
MSE	0.0	0.026	0.330	8.641	7.797	8.641
$ReLU_{ex}$	0.0	0.013	0.501	0.636	0.170	0.630
$DATE_1$	0.1	0.009	0.192	0.583	0.166	0.578
	0.3	0.008	0.141	0.548	0.164	0.544
	0.5	0.008	0.126	0.507	0.179	0.503
	0.7	0.011	0.127	0.497	0.172	0.494
	0.9	0.011	0.128	0.483	0.187	0.479
$DATE_2$	0.1	0.010	0.072	0.106	0.719	0.107
	0.3	0.016	0.071	0.185	0.832	0.186
	0.5	0.010	0.088	0.036	0.816	0.035
	0.7	0.008	0.081	0.056	0.747	0.055
	0.9	0.005	0.059	0.074	0.755	0.074
$DATE_3$	0.1	0.014	0.118	0.782	1.263	0.782
	0.3	0.020	0.065	0.195	0.737	0.190
	0.5	0.020	0.081	0.183	0.755	0.182
	0.7	0.011	0.053	0.114	0.806	0.114
	0.9	0.012	0.068	0.101	0.748	0.100

Table 2: Results with all θ_{D1} settings as average across all analytic functions by loss configuration. We report MSE_{int} , MSE_{D1} and Tan_{ex} , WLR_{ex} , $D1_{ex}$ metrics for interpolation and extrapolation respectively. Results as average across all the functions *lin*, *pow*, *abs*, *sigm*, *sin*.

the bounds of the range, the less variable the prediction. On the other hand, the closer data points to the bounds are considered, the more variable the prediction. Thus, we expect Tan_{ex} to be lower for $DATE_2$ and $DATE_3$, because the slope modelled would match more closely the one at the marginal range.

Figure 2 compares interpolation and extrapolation on 5 runs for $ReLU_{ex}$ and the configurations $DATE_1$, $DATE_2$ and $DATE_3$. The lower row shows the convergence of the predicted function to the true function (black line dashed and black solid respectively) in the range $[-2, 2]$ obtained with each of the models trained over 700 epochs, batch of 20, with Adam optimisation. In Figure 2 we note that the slope of the model prediction in the training data range is still not matching the slope of the function. We note that $DATE_1$ reduces the interpolation error, while having a controlled extrapolation but the y values for $DATE_1$ are systematically too low around $x = 0$.

Lastly, Figure 3 shows the learning curves for $DATE_1$, $DATE_2$ and $DATE_3$. Compared to $DATE_1$, MSE_{D1} seems to converge faster and make interpolation error less variable across multiple runs for $DATE_2$ and $DATE_3$. With $DATE_1$ we have a more stable learning toward the ends of the training, but at higher loss levels. With $DATE_2$ and $DATE_3$ we observe more fluctuations toward the end of training for MSE_{int} , but at reduced error levels. We also observe that the Tan_{ex} error is more controlled at earlier stages for $DATE_1$ but reaches lower error with $DATE_3$.

$f(x)$	Loss	θ_{D1}	σ_R	σ_L	MSE_{int}	MSE_{D1}	Tan_{ex}	WLR_{ex}	$D1_{ex}$
<i>abs</i>	<i>MSE</i>	-	0.047	0.134	0.004	0.347	2.268	2.341	2.282
	<i>ReLEx</i>	-	0.004	0.001	0.020	1.389	0.021	0.599	0.016
	<i>DATE₁</i>	0.5	0.000	0.001	0.001	0.080	0.001	0.432	0.000
	<i>DATE₂</i>	0.5	0.002	0.003	0.007	0.115	0.005	0.417	0.004
	<i>DATE₃</i>	0.5	0.004	0.010	0.023	0.100	0.016	0.417	0.016
<i>lin</i>	<i>MSE</i>	-	0.399	0.745	0.019	0.138	13.219	13.227	13.227
	<i>ReLEx</i>	-	0.003	0.003	0.002	0.152	0.005	0.002	0.002
	<i>DATE₁</i>	0.7	0.002	0.002	0.001	0.002	0.002	0.001	0.001
	<i>DATE₂</i>	0.7	0.002	0.003	0.004	0.073	0.007	0.007	0.007
	<i>DATE₃</i>	0.7	0.005	0.027	0.009	0.019	0.053	0.055	0.055
<i>pow</i>	<i>MSE</i>	-	0.490	0.459	0.081	0.808	8.974	5.317	8.958
	<i>ReLEx</i>	-	0.004	0.024	0.036	0.673	2.970	0.244	2.955
	<i>DATE₁</i>	0.5	0.027	0.008	0.038	0.511	2.376	0.460	2.362
	<i>DATE₂</i>	0.5	0.021	0.022	0.007	0.108	0.141	3.463	0.138
	<i>DATE₃</i>	0.7	0.055	0.024	0.011	0.097	0.206	3.272	0.202
<i>sigm</i>	<i>MSE</i>	-	1.873	1.527	0.006	0.219	5.522	4.874	5.511
	<i>ReLEx</i>	-	0.052	0.142	0.005	0.137	0.180	0.002	0.175
	<i>DATE₁</i>	0.3	0.014	0.045	0.002	0.034	0.154	0.003	0.149
	<i>DATE₂</i>	0.7	0.088	0.246	0.002	0.006	0.005	0.152	0.005
	<i>DATE₃</i>	0.7	0.074	1.527	0.017	0.021	0.020	0.139	0.019
<i>sin</i>	<i>MSE</i>	-	1.895	0.511	0.096	4.040	22.916	34.459	23.270
	<i>ReLEx</i>	-	0.016	0.010	0.499	8.972	0.605	14.676	0.879
	<i>DATE₁</i>	0.5	0.004	0.007	0.184	3.633	0.313	25.968	0.170
	<i>DATE₂</i>	0.7	0.004	0.034	0.023	0.796	2.924	37.808	2.448
	<i>DATE₃</i>	0.7	0.095	0.024	0.031	0.874	1.050	28.014	0.857

Table 3: Results with the best model parameters for each function. We report results with MSE_{int} and MSE_{D1} metrics for interpolation: MSE on training set and $D1$ -Loss, respectively. We report results with Tan_{ex} , WLR_{ex} and $D1_{ex}$ metrics for extrapolation with respect to tangent, weighted linear regression and slope respectively. We display σ_R and σ_L , the extrapolation variability on the left and right side, separately because certain functions are not symmetric. θ_{D1} reported is the one which minimises the MSE_{D1} .

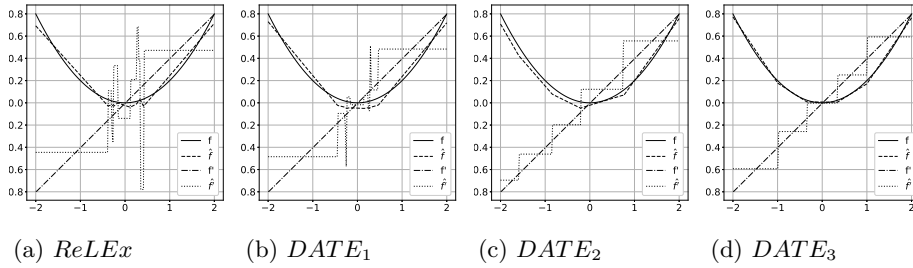
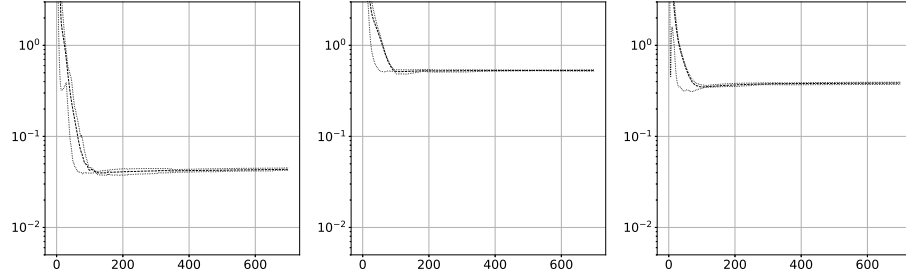
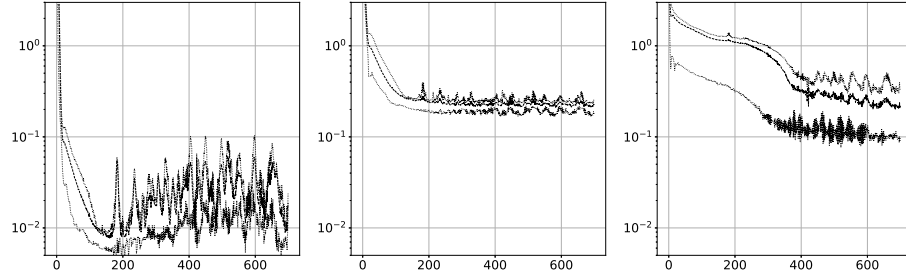


Fig. 2: Various models trained to approximate a quadratic function: *ReLEx* in (a) and *DATE* variations in (b)-(d). The slope of the true function derived from the data points - $f'(x)$ - and the slope of the prediction - $\hat{f}'(x)$. For presentation purposes, only the range $[-1, 1]$ is accounted for.

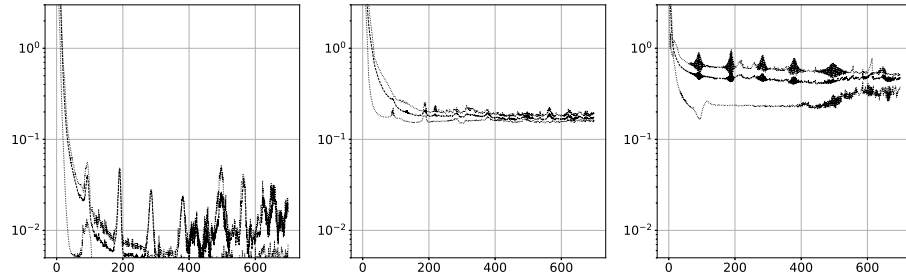
$DATE_1$



$DATE_2$



$DATE_3$



(a) MSE_{int}

(b) MSE_{D1}

(c) Tan_{ex}

Fig. 3: Learning curves from training on scaled quadratic function for $DATE_1$, $DATE_2$ and $DATE_3$ respectively. The learning represented is the average over 10 runs with 700 epochs, batch size 20. The dashed black line represents the average and the dotted lines represents the percentile interval $[20, 80]$.

Limitations. One of the fundamental assumption in *DATE* method is the continuation of the local linear trend at the boundary regions. If the interpolation was not linear, this assumption would fall. This loss would need to be reconsidered under the new non-linear assumption. This would involve training with non-linear activation functions and higher order derivatives. *DATE* method is only defined for univariate situation and only for a single hidden layer. This is obviously a severe limitation in practice and will be need to be addressed. However, as the goal extrapolation is already not uniquely defined in the univariate case, in the multi-dimensional feature space, both, appropriate loss functions and metrics will be even more complex. The *DATE* method only supports linear extrapolation and it assumes that the slope at the margins of the data range is most important for the extrapolation. However, linear extrapolation is preferable because of its simplicity. The reliance on slope at the margins will need more attention in the case of real data, as it may amplify noise.

7 Conclusions

We propose *DATE*, a method which introduces two new loss terms to regularise the behaviour of a ReLU network, such that it produces desirable linear extrapolations beyond the margins of the training data range. We implemented *DATE* in PyTorch and we have applied it to a network with a single hidden layer of 10 neurons, single input, and single output. We have proposed different metrics to measure the extrapolation, as well as interpolation, performance.

In *ReLEx* we observed that better extrapolation results were coming at the cost of higher interpolation instability. We observe now with *DATE* how, in general, consistent linear extrapolations paired with low interpolation error is achieved. *DATE* has shown some preliminary benefits in the NN case and we expect that *DATE* can be extended to other regression methods.

Although with *DATE* we have improved the complexity of the regularisation, we have still some limitations to overcome with future work. Future work will include the formulation of higher order derivative, extrapolation applied to multiple dimensions, different network structures and/or other regression methods as well as working with classification problems.

References

1. Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A., Arshad, H.: State-of-the-art in artificial neural network applications: A survey. *Heliyon* 4(11), e00938 (2018)
2. Douglas, S.C., Yu, J.: Why relu units sometimes die: Analysis of single-unit error backpropagation in neural networks. In: 2018 52nd Asilomar Conference on Signals, Systems, and Computers. pp. 864–868. IEEE (2018)
3. El-Naggar, N., Madhyastha, P.S., Weyde, T.: Theoretical conditions and empirical failure of bracket counting on long sequences with linear recurrent networks. In: Conference of the European Chapter of the Association for Computational Linguistics (2023)

4. Fan, J., Ma, C., Zhong, Y.: A selective overview of deep learning. *Statistical science: a review journal of the Institute of Mathematical Statistics* **36** 2, 264–290 (2019)
5. Fukushima, K.: Cognitron: A self-organizing multilayer neural network. *Biological Cybernetics* **20**, 121–136 (1975)
6. Goodfellow, I., Bengio, Y., Courville, A.: *Deep learning*. MIT press (2016)
7. Lake, B., Baroni, M.: Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In: *International conference on machine learning*. pp. 2873–2882. PMLR (2018)
8. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.: Learning to generalize: Meta-learning for domain generalization. In: *AAAI Conference on Artificial Intelligence* (2018)
9. Lopedoto, E., Shekhunov, M., Aksenov, V., Salako, K., Weyde, T.: Derivative-based regularization for regression (2024)
10. Lopedoto, E., Weyde, T.: Relax: Regularisation for linear extrapolation in neural networks with rectified linear units. In: *Artificial Intelligence XXXVII - 40th SGAI International Conference on Artificial Intelligence, Proceedings. Lecture Notes in Computer Science*, vol. 12498, pp. 159–165. Springer (2020)
11. Lu, L., YeonjongSu, Y., Em Karniadakis, G.: Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics* **28**(5), 1671–1706 (2020)
12. Madsen, A., Johansen, A.R.: Neural arithmetic units. In: *International Conference on Learning Representations* (2020)
13. Marcus, G.F., Vijayan, S., Rao, S.B., Vishton, P.M.: Rule learning in 7-month-old infants. *Science* 283 p. 77–80 (1999)
14. Mitchell, J., Stenetorp, P., Minervini, P., Riedel, S.: Extrapolation in nlp. In: Bisk, Y., Levy, O., Yatskar, M. (eds.) *Proceedings of the Workshop on Generalization in the Age of Deep Learning*. pp. 28–33. Association for Computational Linguistics, New Orleans, Louisiana (2018)
15. Sahoo, S., Lampert, C., Martius, G.: Learning equations for extrapolation and control. volume 80 of. *Proceedings of Machine Learning Research* pp. 4442–4450
16. Suzgun, M., Belinkov, Y., Shieber, S.M.: On evaluating the generalization of LSTM models in formal languages. In: Jarosz, G., Nelson, M., O’Connor, B., Pater, J. (eds.) *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*. pp. 277–286 (2019)