Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Exploring Activation Ensembles for Feed-Forward Neural Networks

Bachelor's Thesis

Sven Brändle

sbraendle@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Florian Grötschla, Joël Mathys
Prof. Dr. Roger Wattenhofer

December 18, 2023

Acknowledgements

I would like to sincerely thank my two supervisors, Florian Grötschla and Joël Mathys. Their consistent support and guidance throughout this bachelor thesis has been invaluable. Their experience and patience allowed me to genuinely enjoy my first research project. A special thanks goes to Professor Doctor Roger Wattenhofer and the Distributed Computing Group for granting me the opportunity to undertake this project.

Abstract

This thesis explores and experiments with various mixtures of activation functions inside a multilayer perceptron. It establishes a baseline for simple synthetic datasets and real-world datasets (MNIST, CIFAR10, ...). The performance of several different arrangements of activation functions such as random initialization, split layers or sequential usage, is investigated. This reveals the potential of combining activation functions. Ultimately, it compares them to the novel Multi-Lane architectures that have two, three or four lanes of neurons with different activation functions. The Multi-Lane models demonstrate improved robustness towards the synthetic dataset selection. Moreover, on the real-world datasets, they are consistently comparable or superior to the traditional ReLU model. Finally, an entropy Lane-Loss for the Multi-Lane model is implemented and tested with two different criteria. This enables the model to implicitly pick the most promising activation function lane of the Multi-Lane network.

Contents

Chapter 1

Introduction

Feed-forward neural networks are one of the fundamental building blocks for the success of deep learning. The ability of a network to learn tasks beyond the complexity of linear regression is given thanks to the activation function. It is the component that expands the learnable space to non-linearity. There are an infinite number of possible activation functions. Finding the most powerful one, or a combination thereof, is an ongoing area of research. Activation functions such as the sigmoid, tanh, or the widely employed Rectifier Linear Unit (ReLU), each have their advantages and disadvantages. Therefore, some functions are better suited for certain datasets than others. This challenge often results in a trial-and-error desperation. But research has shown that combinations or modifications of activation functions may be superior to pure, single activation models (Agostinelli et al. [2014]).

The Universal Approximation Theorem states, that even with a single hidden layer feed-forward neural networks can effectively approximate any continuous function, given a large enough number of neurons (Cybenko [1989]). However, instead of solely approximating the input-output relationship, the model ideally also generalizes to a wider range of input data. Problem specific activation functions might help minimizing the out-of-distribution error.

In this thesis, we explore various activation ensembles and compare them to the traditional, single activation function approach. We choose to evaluate them on multilayer perceptrons. Typically, research focuses on the performance of models on real-world datasets such as MNIST, CIFAR10. To fully comprehend how different activation functions behave, we additionally test models on more elementary, synthetic datasets, such as addition and multiplication. Furthermore, we examine the performance of the models on domains outside the training set. This provides insight into the model's ability to learn the underlying mathematical function, rather than simply learning the relationship within the given training bounds.

As a contribution, we introduce a novel architecture called the Multi-Lane network. Instead of using a single fully connected network, we create multiple lanes that operate as individual feed-forward networks. The outputs of these lanes are then combined in the final hidden layer. Each lane utilizes a separate activa-

tion function. This allows the model to prioritize the most important activation function for a given task more easily. The Multi-Lane model takes advantage of undisturbed sections of neurons with the same activation function.
The architecture of our Multi-Lane models allows the conduction of further experiments. To further reduce the loss of the out-of-distribution range, we introduce an entropy loss to our Multi-Lane model. This theoretically allows the network to implicitly determine which lane is more suitable for the given dataset. The unused lanes could be eliminated, in the hope of leaving the network with the most suitable activation function.

Chapter 2

Background

2.1 Multilayer Perceptron (MLP)

The classic multilayer perceptron (MLP) is a simple yet powerful neural network that is widely used in machine learning and artificial intelligence. Also known as a feedforward artificial neural network, the MLP is used for various tasks including image recognition (Touvron et al. [2022]), weather forecasting (Narvekar and Fargose [2015]), and medical diagnosis (Yan et al. [2006]). To illustrate the functionality of the network, we look at an example. Suppose one wants the network to learn to add two numbers x_1 and x_2. Each input feature corresponds to a neuron/perceptron in the input layer, in our case x_1 and x_2. The input data is then passed through weighted connections to hidden layer neurons. Each neuron performs a weighted summation with all inputs from the previous layer. The first hidden layer passes the values to the next hidden layer, and so forth. The output layer processes the final hidden layer activations, resulting in the model prediction (see Figure 2.1). In our example, we want to predict the sum of x_1 and x_2.
A single perceptron is shown at the bottom of Figure 2.1. It takes the weight of the incoming connections and multiplies them by the outputs/activations of the previous layer. The sum of these products, along with an additional bias b, results in the value z of the neuron. This value z is then fed into an activation function to create the activation of the neuron (see also Chapter 2.3). The activation functions play a key role in enabling the network to learn non-linear behavior.
Each input x of the input layer is connected to each neuron in the first hidden layer, which is connected to each neuron from the second layer and so on. Each connection has a weight w, which is initialized randomly when the model is created. The value z of the j-th neuron in the l-th hidden layer is calculated as follows:

$$z_j^{(l)} = \sum_{i=1}^{n} w_{ij}^{(l)} \cdot h(z_i^{(l-1)}) + b_j, \tag{2.1}$$

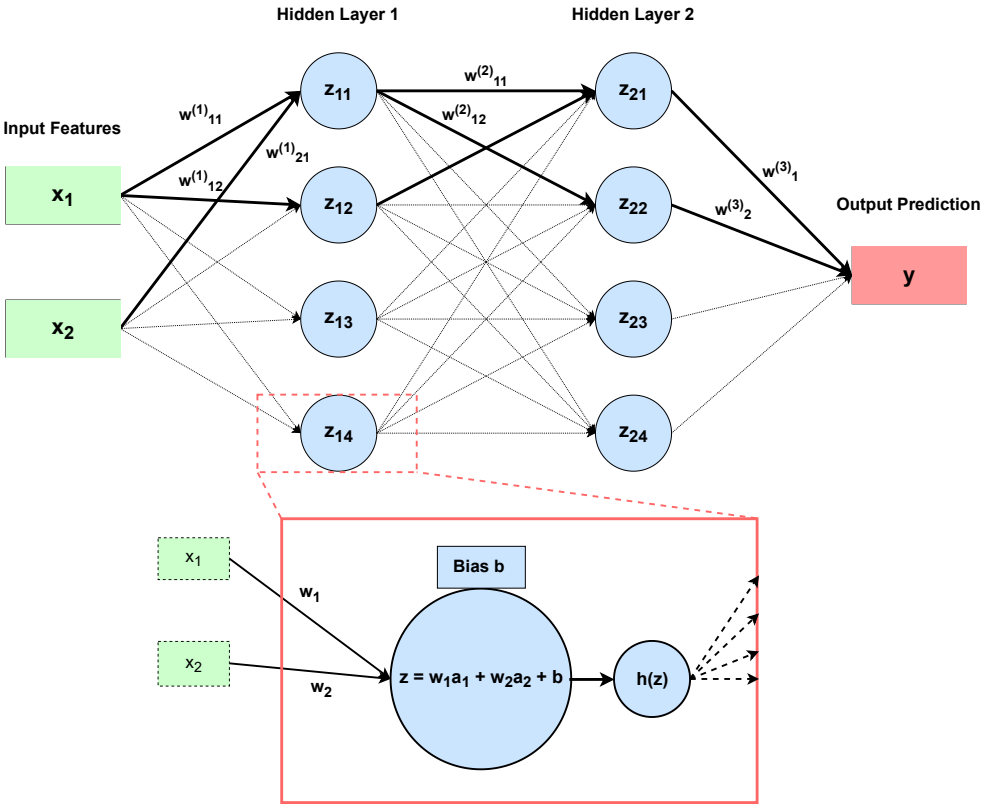

Figure 2.1: This multilayer perceptron (MLP) consists of two layers, each with four neurons (blue). The output prediction (red) is learned by the network from the input features (green). A close-up of a neuron is shown at the bottom with activation function $h(z)$. In this case it is $a_1 = x_1$ and $a_2 = x_2$. The network adapts the weights, w, in a way that the output is as close to the true value as possible (see also Section 2.2).

where n is the number of previous neurons, b is an additional bias scalar and h is the activation function. More common is the representation in matrix form:

$$Z^{(l)} = W^{(l)} \cdot A^{(l-1)} + b^{(l)}, \tag{2.2}$$

where W is the weight matrix, A represents the activations of the previous layer and b in the bias vector of the l-th layer.

2.2 Training and Evaluation of the Network

To ensure effective learning, a large enough dataset of samples is necessary. The dataset is typically divided into three different parts: a training set, a validation

set, and a testing set. The training set is the largest and is used to train the model, while validation and testing sets are smaller. The dataset is usually split in a 70%-15%-15% ratio for training, validation, and testing, respectively. In this project, we use the term "validating" to refer to the evaluation of the model *during* training, and "testing" to refer to evaluation of the model *after* training.
To prioritize essential aspects, we will provide only a concise overview of backpropagation. Backpropagation is the fundamental training algorithm used in neural networks. We have our model as described in Section 2.1. We initialize the weights in an MLP randomly before the start of training. When inputting features from the training dataset, the output will also be random. However, with samples from the training set, we know what the output should be. For each input sample, we compare its output with the true label and use a loss function to quantify the network's error. We calculate the gradient of the loss with respect to the weights and biases. Since we want to minimize the loss function, we take a step in the opposite direction of the gradient. The step size is also called the learning rate. This step happens after a predefined number of samples, also called the batch size.
For the purpose of understanding the following chapters we will look at two different loss functions.

2.2.1 MSELoss

The mean squared error, also known as the L2 loss, measures the squares of the errors. Our errors are the differences between the model predictions and our true labels. Therefore, the loss for a batch with n samples, prediction $\hat{Y}$ and given label Y is:

$$MSELoss = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2. \tag{2.3}$$

2.2.2 Cross-Entropy Loss

The cross entropy loss function is a commonly used loss function for classification tasks. In such tasks, the model has one output for each possible class. Using a softmax function, the outputs are transformed into a probability distribution, which sums up to one. Each output now represents the probability that the input belongs to its respective class. The cross entropy loss function for k classes is calculated as follows:

$$CELoss = -\sum_{i=1}^{k} Y_i \cdot log(P_i), \tag{2.4}$$

where Y_i is the true label of class i and P_i is the predicted probability of the input belonging to class i.

2.3 The Role of the Activation Function

The activation function is used to introduce non-linearity into the network. Suppose we use a linear activation function, such as the identity function. Equation 2.2 would change to:

$$Z^{(l)} = W^{(l)} \cdot Z^{(l-1)} + b^{(l)}. \tag{2.5}$$

Looking at the input-output relation reveals:

$$Y = W^{(l)} \cdot W^{(l-1)} \cdot ... \cdot W^{(1)} X + W^{(l)} \cdot ... \cdot W^{(2)} \cdot b^{(l-1)} + ... + b^{(l)}, \tag{2.6}$$

where X is the input vector. This results in a purely linear relationship between inputs and outputs, essentially becoming a linear regression model. But even relatively simple problems like learning multiplication need non-linearity (see Chapter 5).

There is only one important property that the activation functions should have. In the backpropagation process, the gradient must "flow" through the network back to the weights and biases. As stated in Gulcehre et al. [2016]'s work: "An activation function is a function h: $\mathbb{R} \to \mathbb{R}$ that is differentiable almost everywhere."

The choice of an activation function is crucial for the performance of a network. It greatly influences the training process, affecting convergence, speed of convergence (Gulcehre et al. [2016]), generalization and the ability to efficiently extract patterns from input features (see Chapter 5). There are many possible activation functions, but we will focus on some of the most commonly used ones, which we will also use in the following experiments.

2.3.1 Identity

The simplest activation function is the identity function:

$$h_{identity}(z) = z. \tag{2.7}$$

The identity function is mainly used for linear regression because it is unable to approximate non-linear functions. It does not transform the input, it simply returns it as is (see Figure 2.2). The identity function is continuous and differentiable with a gradient equal to 1 everywhere, which simplifies the backpropagation and makes computation extremely effective (essentially there is no computation needed).

2.3.2 Sigmoid

The sigmoid function is defined as follows:

$$h_{sigmoid}(z) = \frac{1}{1 + e^{-z}}. \tag{2.8}$$

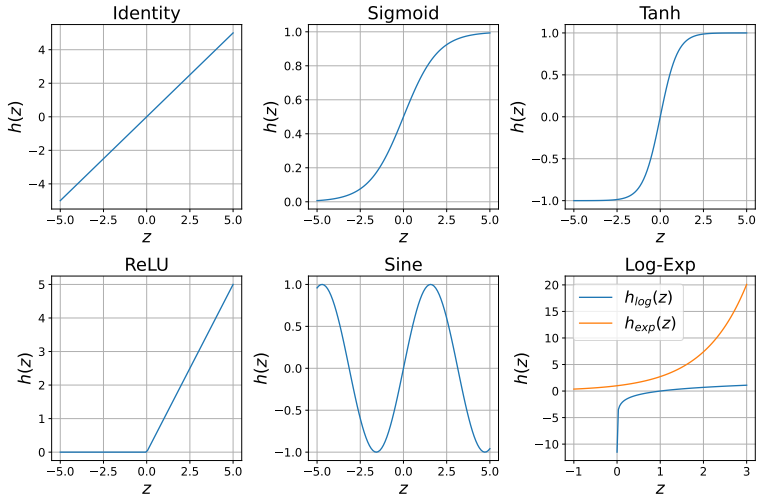

Figure 2.2: Six distinct activation functions were employed in the experiments. x-axis illustrates the neuron values, denoted as z. The y-axis represents the transformed activations.

The sigmoid function is a type of logistic function. Its output ranges from 0 to 1 and it is differentiable and continuous throughout its domain, therefore resulting in smooth gradients. The main advantage of the sigmoid function is its usability in binary classification, as it outputs the probability that the input belongs to class 0/1. However, it also has some disadvantages. When the input to the sigmoid function becomes either very large or very small, the gradients of the sigmoid converge to zero. This phenomenon is also called saturation and makes convergence very slow (Gulcehre et al. [2016]).

2.3.3 Tanh

The hyperbolic tangent function, often abbreviated as tanh, is defined as follows:

$$h_{tanh}(z) = \frac{e^{2z} - 1}{e^{2z} + 1}. \tag{2.9}$$

The tanh function is similar to the sigmoid function. It is also continuous and differentiable everywhere, although the gradient of the tanh function is greater around zero. It also suffers from saturation for very large or very small inputs (see Dubey et al. [2022]). Unlike to the sigmoid function, the values for the tanh function range from -1 to 1.

2.3.4 ReLU

The Rectifier Linear Unit (ReLU) is the most commonly used activation function in neural networks. Unlike the previous activation functions, ReLU is continuous, but not differentiable everywhere:

$$h_{relu}(z) = max(z, 0). \tag{2.10}$$

In practice, the gradient of ReLU at zero is set to 0. ReLU solves some of the problems of tanh and sigmoid. First, it reduces the saturation problem, since the gradient for all positive inputs is 1. This enables consistent and strong signals for backpropagation. The function is very simple, which makes it computationally efficient (see Dubey et al. [2022]). Additionally, ReLU is generally sparse as it is zero for all negative values (also called dying ReLU). This can be beneficial with regards to regularization (see Ide and Kurita [2017]) and network pruning (see Lee et al. [2018]). However, under certain circumstances this may lead to convergence problems (see Szandała [2021]).

2.3.5 Sinusoidal

Sinusoidal functions are not as commonly used as the previously mentioned activation functions, but they are still viable. Sinusoidal activation functions introduce periodic patterns into the network. This periodicity can be beneficial for some parts and problems (see Chapter 5), but it can also hinder the training process (Parascandolo et al. [2016]). We will use the sine activation function in the following experiments:

$$h_{sin}(z) = sin(z). \tag{2.11}$$

Around zero the sine function behaves similarly to the tanh function. It is also shown that the function trains comparable to the tanh function and that the networks often rely only sparsely on the periodicity of the activation function (see Parascandolo et al. [2016]).

2.3.6 Log-Exp

The Log-Exp function is not a classic activation function. We use the Log-Exp function as a combination of a logarithmic activation function, followed by a layer of exponential activation function. This results in the following input-output relationship in a two-layer network:

$$Y = W^{(2)} \cdot \exp\left(W^{(1)} \cdot \left(\ln\left(W^{(0)} \cdot X + b^{(0)}\right) + b^{(1)}\right)\right) + b^{(2)} \tag{2.12}$$

The exponential function cancels out the logarithm, leaving behind the product of the input vector. Lets consider the network as in Figure 2.3. The input-output

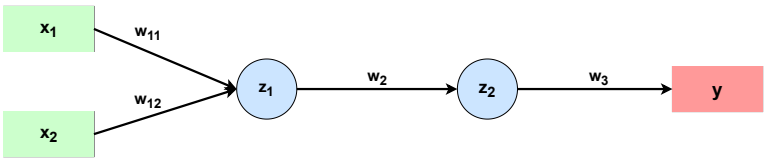

Figure 2.3: Log-Exp Example Network. The Log-Exp model uses a logarithmic activation function in the first layer and a exponential activation function in the second layer.

relation is as follows:

$$\begin{aligned} y &= w_3 \cdot \exp(z_2) + b_3 \\ &= w_3 \cdot \exp(w_2 \cdot a_1 + b_2) + b_3 \\ &= w_3 \cdot \exp(w_2 \cdot \ln(w_{11} \cdot x_1 + w_{12} \cdot x_2 + b_1) + b_2) + b_3 \\ &= w_3 \cdot (w_{11}x_1 + w_{12}x_2 + b_1)^{w_2} \cdot \exp(b_2) + b_3, \end{aligned} \tag{2.13}$$

where w are weights, b are biases, z are the values of the neurons and a are activations. With this property, Log-Exp may work very well on the `Multiplication` dataset.

2.3.7 Softmax

The softmax function is typically used in the output layer of multi-class classification. It transforms the raw model outputs into a probability distribution. For an input vector $z = (z_1, z_2, ..., z_k)$ with k classes, the softmax function is defined as follows:

$$softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}, \tag{2.14}$$

with z_i being the i-th element of the input vector. The softmax function ensures that the elements of the output vector are non-negative and sum up to 1, effectively representing probabilities.

Chapter 3

Related Work

The work on new combinations of activation functions within a single neural network has received much attention in recent years and decades. The widely popular used Rectifier Linear Unit (ReLU) was introduced as an activation function (Glorot et al. [2011], Nair and Hinton [2010]) around 2010/2011, which led to a renewed search for new and better activation functions. With its simple forward and backward computation and its non-saturating nature(Dubey et al. [2022]), ReLU has the ability to effectively help the network learn (Krizhevsky et al. [2012]). Another independent idea was ensemble learning in the work of Sagi and Rokach [2018]. It uses the concept of "wisdom of the crowd", which combines and considers the results of many (different) models to make the final prediction. Instead of using multiple models, we use multiple activation functions. We focus on mixing activation functions to take advantage of the strengths of each activation function. However, there is a lot of work focusing on other ideas and approaches.

Qian et al. [2018] create adaptive activation functions. They improve the performance of ReLU by solving the vanishing gradient problem. Introducing a trainable parameter into the ReLU function prevents the left side of the function from being zero. This activation function is also called parametric ReLU (PReLU). They also successfully propose two additional parameters, s and a, for the tanh function. They help to regulate the slope and amplitude of the function to prevent saturation.

Similar work has been done by Trottier et al. [2017]. They modify the Exponential Linear Unit (ELU) function with additional parameters. Two additional parameters, a and b, allow the network to use different activation functions in different parts of the network.

Nandi et al. [2020] propose an alternative ensemble of activation functions. They combine the model ensemble approach with our activation ensemble approach. The activation ensemble is proposed to improve classification accuracy. Several networks are trained, each with a different activation function. The predictions of the models result in the final prediction using majority voting. This reduces the importance of a single activation function and instead relies on a diverse set of functions.

Similar to the previously mentioned PReLU, the SReLU was introduced in the work of Jin et al. [2016]. The SReLU is defined as a combination of three linear functions:

$$h(x_i) = \begin{cases} t_i^r + a_i^r(x_i - t_i^r), & \text{if } x_i \geq t_i^r \\ x_i, & \text{if } t_i^r < x_i < t_i^l \\ t_i^l + a_i^l(x_i - t_i^l), & \text{if } x_i \leq t_i^l, \end{cases} \tag{3.1}$$

where $\{t_i^r, a_i^r, t_i^l, a_i^l\}$ are four different learnable parameters, ensuring individual activation functions for each neuron. SReLU can be easily be integrated into existing networks and demonstrates the ability to learn both convex and non-convex functions.

A key statement, very relevant to our work, is made in the work of Agostinelli et al. [2014]. They create an Adaptive Piecewise Linear (APL) unit, which is the sum of hinge-shaped functions:

$$h_i(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s), \tag{3.2}$$

where x is the input to the activation function, S is a hyperparameter that determines the number of hinges in the function. a and b determine the slope of the piecewise unit and the location of the hinge, respectively. They show that their activation function leads to a significant improvement. The network learned several different functions, suggesting that one universal activation function may not work optimally.

A particular motivation for this work was provided by Harmon and Klabjan [2017]. In their paper *Activation Ensembles for Deep Neural Networks* they introduce a new concept called activation ensembles. It focuses on the idea of combining different, already known activation functions within a single neural network. By introducing a new variable, α, for each activation layer, it allows the network to have multiple activation functions at each neuron. The network implicitly "chooses" the stronger activation functions by learning the specific α. Depending on the dataset used, the network will favour different activation functions in different parts of the model. Again, choosing different activation functions in a network improves its performance.

Chapter 4

Model Architectures

In this chapter we introduce the most important models that we will be focusing on in our experiments. The models can be sorted into two different types. The first type, the fully connected networks, only changes the activation function arrangement of the classic MLP. The second type is the newly introduced Multi-Lane network, that modifies the traditional MLP architecture.

4.1 Fully Connected Networks

These models are basic multilayer perceptrons as described in Section 2.1. However, we arrange different activation functions in specific ways. We create baseline, sequential, ensemble and random models. The number of parameters within a network is calculated as described in Appendix A.

4.1.1 Baseline Model (B)

In order to fairly compare our different model types, we will build baseline models. The baseline models (B) are MLPs with only one activation function. Every neuron uses the same activation function. The baseline serves as a reference point. It provides a simple benchmark for evaluating other models and will help us determine if our other models show improvement.

4.1.2 Sequential Model (S)

The sequential model uses specific activation functions in different layers of the model. Our model utilizes one activation function in the first half of the layers and a different one in the second half of the layers. Since we will be looking at two-layer models, we will only use two different functions for the two layers. The model is illustrated in Figure 4.1.

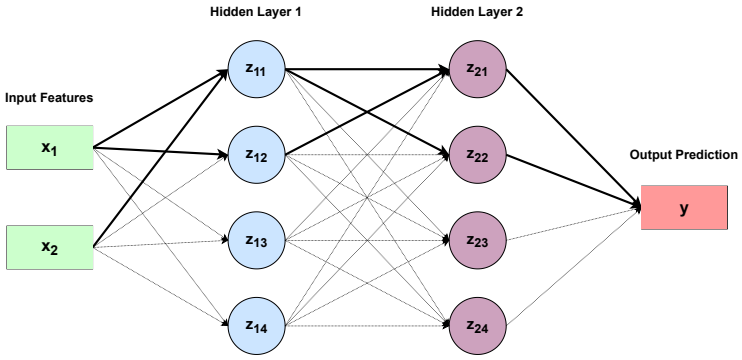

Figure 4.1: This is an illustration of a Sequential (S) model. The model utilizes one activation function in the first layer (blue) and different activation function in the second layer (purple).

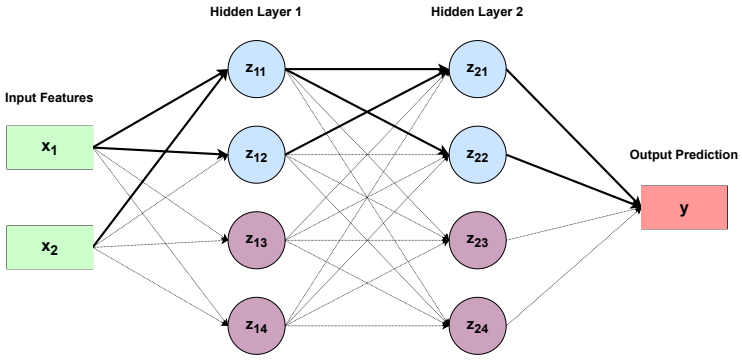

Figure 4.2: This is an illustration of an Ensemble (E) model. The model uses one activation function in the top half of the layer (blue) and different activation function in the bottom half of the layer (purple).

4.1.3 Ensemble Model (E)

Our ensemble model also uses two distinct activation functions. Rather than using two different functions for different layers, the ensemble model employs two different functions within each layer. We therefore split each layer in two (see Figure 4.2). The first half of the layer uses one activation function, while the second half of the layer uses a different activation function.

4.1.4 Random Model (R)

In the Random model, we use various activation function. This model does not use any fixed activation function. It rather chooses an activation function ran-

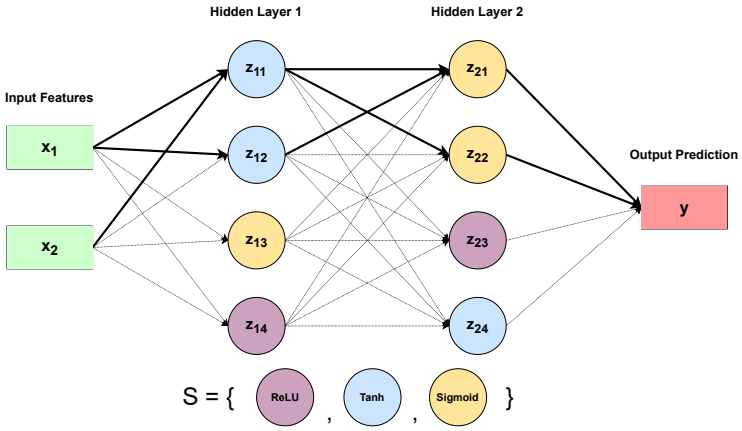

Figure 4.3: This is an illustration of an example Random model. The model initializes each neuron with a random activation function from a given set S. In this example, the set includes ReLU, Tanh and Sigmoid.

domly for each neuron in the network. This happens at initialization of the network. The function is chosen randomly from a given set S of activation functions. The model can have as many activation functions as the set has. An example network is illustrated in Figure 4.3. For larger models and a set S of size two, the random model basically functions as an ensemble model. The expected number of neurons in a layer is 50%. Since the weights of our models are initialized randomly, it does not matter which locations the neurons are in.

4.2 Multi-Lane Networks

4.2.1 The Multi-Lane Network (D, T, Q)

The Multi-Lane network is a novel architecture based on the MLP. It consists of multiple smaller fully connected networks. Each of these sub-networks uses a different activation function. A Multi-Lane network with two lanes (Double-Lane) is depicted in Figure 4.4. We use Multi-Lane models with two (D), three (T) and four (Q) different lanes. We hope that the Multi-Lane structure of the network will make it easier for the model to prioritize activation functions. The Multi-Lane might especially contribute to a network's robustness. This could enhance adaptability to various datasets, particularly simpler ones. On real-world datasets like MNIST or CIFAR10, the Multi-Lane model may be able to extract different features on different lanes, depending on the activation functions. This could prevent one activation function from dominating over the others and lets us assign contributions to the distinctive lanes (see Section 4.3).

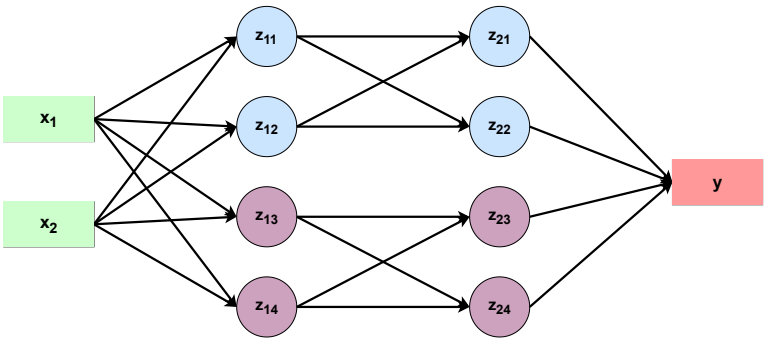

Figure 4.4: This is an illustration of a Multi-Lane model with two lanes. The top lane uses an activation function, the bottom lane uses a different activation function.

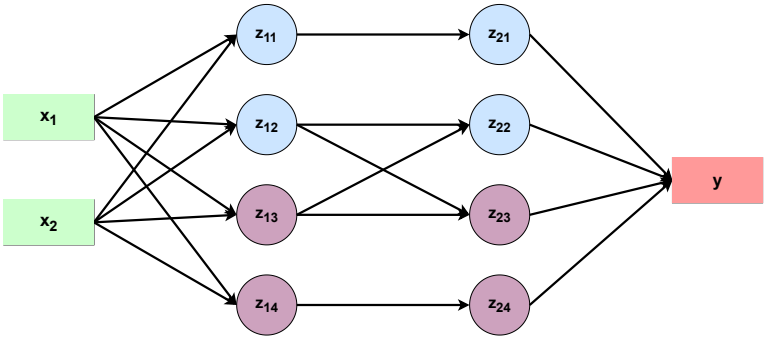

Figure 4.5: The modified Multi-Lane model combines the approaches of the Ensemble and the Multi-Lane networks. It has two lanes, on top and on the bottom, and a section that combines the two activation functions.

4.2.2 The Modified Multi-Lane Network

The modified Multi-Lane model is a combination of the Double-Lane model and the Ensemble (E) model. It features two independent lanes on the top and bottom (see Figure 4.5). Each lane includes 25% of the neurons in each layer. The middle section combines the two activation functions and consists of the remaining 50% of the neurons in each layer.

4.2.3 The Double Quad-Lane Network (DQL)

The Double Quad-Lane model utilizes two distinct Multi-Lane modules consecutively. The model is depicted in Figure 4.6. The shown model uses two Quad-Lane modules. We will use this model only on specific datasets in the Lane-Loss Section 4.3. The datasets consist of two nested mathematical operations. With Lane-Loss implemented as described in Section 4.3, the model can theoretically

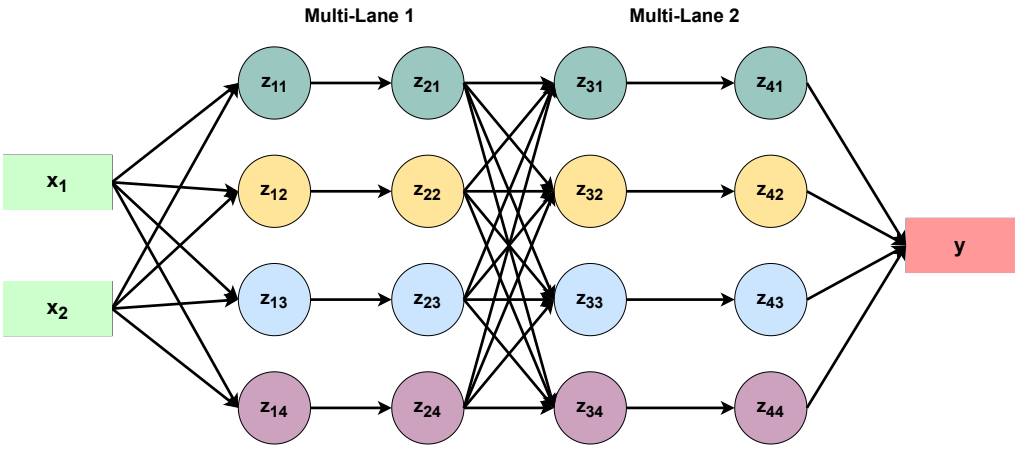

Figure 4.6: This is an illustration of the Double Quad-Lane model. The model is built with two consecutively used Quad-Lane modules. In this example, the lane is only made up of a single neuron. Usually the lane contains more than one neuron.

learn the underlying function better by sequentially using different lanes.

4.3 Lane-Loss

We want to further improve our Double Quad-Lane model by trying to let the network itself choose which lanes to take. For example, say we are using a Quad-Lane model on the `Sine` dataset. The model should choose the sine lane because it works best for this data set. We do this by introducing an entropy loss in addition to the MSE loss. Entropy represents the disorder of a system. It is defined as follows:

$$H(X) = -\sum_{x \epsilon X} p(x) \cdot \log_2 p(x), \tag{4.1}$$

where X is the set of values in our system and $p(x)$ are the values to adjust. The $p(x)$ values in our system should represent the percentage usage of the lanes. If, for example, only one lane x_1 is used, then $p(x_1)$ is 1. If the network does not use lane x_1, then the $p(x_1)$ value is 0. Therefore, minimizing the entropy loss converges to the network also only using one lane. For our x values, we need an additional criterion that represents the importance of the lanes. We tried two different criteria.

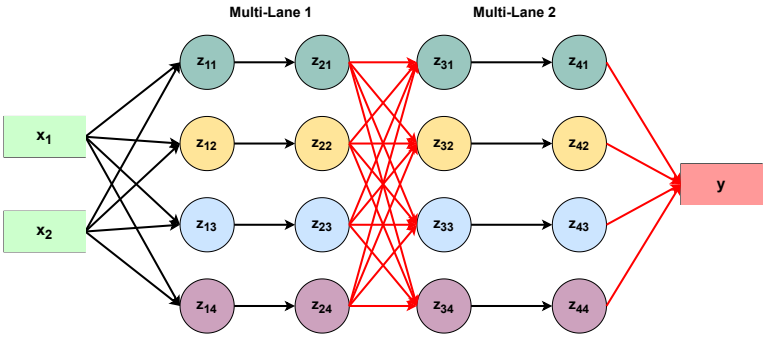

Figure 4.7: The Double Quad-Lane model for our Lane-Loss experiment. The red colored weights in the middle are used to calculate the Lane-Loss for the first Multi-Lane module, while the output weights are used for the Lane-Loss of the second Multi-Lane module.

4.3.1 Criterion: Weight Squares

Our first criterion is based on the square values of the weights as an indicator of lane relevance. We work with the following assumption: The higher the average squared weights of a lane, the more important the lane is. First, we compute the squares of the output weights of the Multi-Lane modules (colored red in Figure 4.7). We do this for both the first Multi-Lane module and the second Multi-Lane module. The averaged weights per lane are then scaled with a softmax function to create our $p(x)$. Finally, our lane entropy loss is defined as:

$$H_{lane}(X) = -\sum_{i}^{n} p(x_i) \cdot \log_2 p(x_i), \tag{4.2}$$

with

$$p(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}, \tag{4.3}$$

for k lanes and $x_i = \frac{1}{m}\sum_{j=1}^{m} {w_j}^2$ being the mean square of all the m weights of lane i. Adding up the weighted lane loss to the MSE loss results in the final loss of:

$$L_{tot} = L_{MSE} + \alpha H(X). \tag{4.4}$$

In this case $H(X)$ is the sum of $H_1(X)$ and $H_2(X)$, which represent the Lane-Loss for the first and the second Multi-Lane module, respectively. We examine two different strategies for α. We choose different constant alphas at the beginning of training and we used a scheduler that increases alpha every epoch by a factor of 1.1, starting with an alpha of 0.00005. We will only on the scheduled version of α.

4.3.2 Criterion: Contribution

Instead of looking only at the weights of the layer, we now decide to include the activation values. With this approach, the criterion not only depends on the weights of the layer, but also on the previous layers. We therefore calculate the contribution by multiplying the weight matrix by the previous layer activations. We then square the contributions and average them over all the contributions of the specific activation function. Using the softmax function again, we transform our values to get the "percentage" $p(x)$ contribution to the output for each activation function. Again, we do this for both the first and the second Multi-Lane module.
Again, we get a final loss of:

$$L_{tot} = L_{MSE} + \alpha H(X). \tag{4.5}$$

In this case $H(X)$ is the sum of $H_1(X)$ and $H_2(X)$, which represent the Lane-Loss for the first and the second Multi-Lane module, respectively. We examine two different strategies for α. We choose different constant alphas at the beginning of training and we used a scheduler that increases alpha every epoch by a factor of 1.1, starting with an alpha of 0.00005. We will focus on the scheduled version of α.

Chapter 5

Experiments and Results

Activation functions add complexity to the models by introducing non-linear properties. There are many different activation functions available, each with its own strengths and weaknesses (see Section 2.3). It is easiest to just use one activation functions in the neural network. We explore the idea of combining different activation functions, in the hopes that the network takes advantage of the individual strengths of each function. The goal of the following chapter is to improve our comprehension of how activation functions interact with each other within a single network. We begin by explaining our setup, hyperparameters, and datasets. Next, we will create the baseline models for each dataset. We compare these with our new model architectures and activation combinations. We continuously compare the model types on synthetic (more basic) and real-world (MNIST, CIFAR10, ...) datasets. For the synthetic datasets, we are additionally interested in the model's performance outside of its training ranges.

5.1 Setup

5.1.1 Datasets

To learn as much as possible about different network structures and activation function combinations, we test our implementations on both real-world and synthetic datasets. For the real-world datasets, we chose MNIST, FashionMNIST, ISOLET and CIFAR10. MNIST is a large collection of handwritten digits. FashionMNIST contains images of ten different types of clothing. Both are low resolution images with 28x28 pixels and 10 different output classes. CIFAR10 is also an image dataset. It contains 10 different objects and animals such as airplanes, cats and ships. These are color images with a resolution of 32x32 pixels. ISOLET is a dataset with 617 different features of 26 spoken letters (Fanty and Cole [1990]).

Since it is harder to fully understand what is happening on the larger datasets, we decide to create some simpler datasets ourselves. These "synthetic" data have exactly two inputs x_1 and x_2 and one output. The formula for the `Addition`

Dataset	Formula
Addition	$x_1 + x_2$
Multiplication	$x_1 \cdot x_2$
Sine	$sin(x_1 + x_2)$
Max	$max(x_1, x_2)$

Table 5.1: A dataset table for the following experiments. It shows the formula with which the dataset is generated from. The ranges for x_1 and x_2 are found in Table 5.2.

dataset, `Multiplication` dataset, `Sine` dataset, and `Max` dataset are listed in Table 5.1. The ranges of x_1 and x_2 are according to the Table 5.2. We split the datasets into three different parts, training, validation and test set. For the synthetic dataset, we create additional samples with an extended range. This helps us test the model on out-of-distribution (OOD) samples and shows us how well the model adapts to ranges outside the training range.

Dataset	Train/Val/Test/OOD Samples	TVT Range	OOD Range
Addition	7000/1500/1500/10000	[0, 100]	[0, 10000]
Multiplication	7000/1500/1500/10000	[0, 20]	[0, 40]
Sine	7000/1500/1500/10000	[0, 4π]	[0, 8π]
Max	7000/1500/1500/10000	[0, 20]	[0, 40]
MNIST	48000/12000/10000	-	-
FashionMNIST	48000/12000/10000	-	-
ISOLET	5457/1170/1170	-	-
CIFAR10	40000/10000/10000	-	-

Table 5.2: A dataset table for the following experiments. The datasets are as described in Section 5.1.1. The second column shows how many samples the training, validation, test and OOD set have. OOD stands for out-of-distribution and refers to samples from outside the training range. TVT range is the range used for training, validating and testing. OOD Range is the range of the OOD data samples.

5.1.2 Hyperparameters

Synthetic Datasets. In order to fairly compare different activation functions and models, we fix some of the parameters and hyperparameters. We train each model for a maximum of 200 epochs. If the validation loss does not improve for 20 epochs, we stop the training process early. For testing, we use the model with the lowest validation loss. We train each model type exactly 10 times and

take the average of the loss and the accuracy as relevant. We use a batch size of 16 and a learning rate specific to each dataset (see Appendix A.2). When we talk about loss, we use an MSELoss. For the sake of simplicity, and particularly, interpretability, we do not make use of batch normalization and dropout, and only use linear layers and activation functions.

Real-World Datasets. The hyperparameters for the real-world datasets are similar to those used for the synthetic datasets. Each model is trained for a maximum of 150 epochs and is stopped early if the validation loss does not improve for 20 epochs. All model results are the average of 10 runs. We use a batch size of 16 for CIFAR10 and ISOLET, and 64 for MNIST and FashionMNIST. We fix the individual learning rate for each dataset (see Appendix A.2). For the real-world datasets, we use batch normalization. To generate the output probabilities we use a softmax function following the output layer. The cross entropy loss is best suited for the classification tasks.

5.2 Model Performances

Here we show how the models and architectures work. All models and architectures are defined in Chapter 4. Unless otherwise stated, we use two-layer models with different numbers of neurons per layer. In the experiments below and in the figures, we will use the abbreviations given in brackets in Chapter 4. Because the ratio of neurons to weights is different for some models, we will often plot performance relative to the number of parameters in the network. For the real-world datasets, we evaluate performance in terms of test loss (not test accuracy). Some experiments will also show that certain activation functions perform better with a larger number of parameters, while others learn better (relatively speaking) with fewer parameters. For information on how we calculate the number of parameters, see the Appendix A.1. We will not present the performance of all our models, but rather show the most interesting ones. We refer to the appendix for additional figures and results.

5.2.1 Synthetic Datasets

Baseline (B). We want to compare our models with standard single activation function models. Therefore, we have train models with sigmoid, tanh, ReLU, sine, log-exp, and identity activation functions for all datasets. These will serve as our baseline against which we will compare our other models.
The test loss for the synthetic datasets are is shown in Figure 5.1.
As anticipated, the identity network performs well only on the linear `Addition` dataset and fails to learn any other non-linear dataset. On the `Sine` dataset, the sine activation function is far superior to the other functions. Sigmoid and tanh

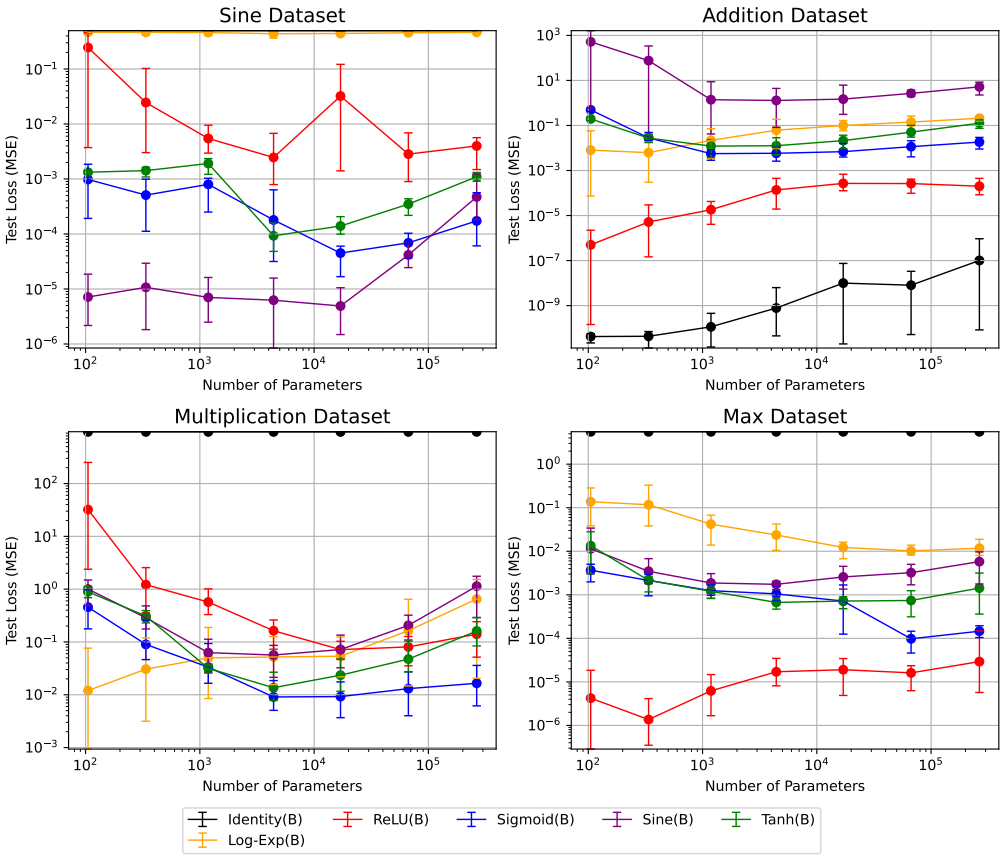

Figure 5.1: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as defined in Section 5.1.1. The graphs show the trend of the MSE loss as the model complexity increases. Notably, sine, identity, and ReLU activation functions are best for `Sine`, `Addition`, and `Max` dataset respectively. On the `Multiplication` dataset more simple models perform better with Log-Exp activation, more complex networks favor Sigmoid Activation function.

are both competing for second place. One could argue that since the sigmoid/tanh function has have very sine-like properties around the origin (Parascandolo et al. [2016]), it also performs better than, for example, the ReLU activation function. On the `Addition` dataset, both identity and the piecewise **linear** ReLU clearly outperform the others. Again, sigmoid and tanh achieve similar results. The sine function performs worst on the `Addition` dataset.
There is no clear winner on the `Multiplication` dataset. For fewer parameter networks, it is the Log-Exp function that has a test loss almost two orders of magnitude lower than the other functions. This changes significantly as the number of parameter increases. Sigmoid, tanh, and ReLU all outperform Log-Exp, with sigmoid being the best one out of them.
ReLU performs best for the `Max` dataset. Sigmoid and tanh perform similarly, with sigmoid being slightly better.
We now analyze the out-of-distribution datasets as described in Subsection 5.1.1 (see Fig. 5.2).
Many models do not generalize well or at all to out-of-distribution (OOD) data. It is important to be cautious when analyzing OOD loss. Just because the identity function achieves a lower OOD loss than ReLU on the `Sine` dataset, it does not mean that the identity function generalizes better. Rather, it means that ReLU fails to predict the output range of the sine function.
On the `Sine` dataset, only the sine function itself with fewer parameters adapts somewhat to an increased range. All other functions do not generalize at all. On the `Addition` dataset, identity generalizes incredibly well with a maximum loss of about 10^{-2}. The networks with fewer parameters even have a loss as small as 10^{-6}. ReLU performs better than the other models with an average loss of 10^2 to 10^3, even including instances with OOD loss of 10^{-2} and lower.
For the `Multiplication` dataset, it is really only the Log-Exp models that are able to generalize somewhat with an MSE loss of less than 100 (remember, with an OOD input range of 0-40 instead of 0-20).
The underlying function of the `Max` dataset is best learned by the ReLU function. What is interesting is the performance of Log-Exp relative to the performance of the other functions, with a loss of about 1 to 0.1.

Now we want to further analyze the learning capabilities of the MLP. On our synthetic datasets, we only work with functions that have an input dimension of two. This means that we can visualize the loss on a 2D heatmap. More plots can be found in the appendix B.1. We plot the loss space of four activation functions. The loss is given as a color that represents the loss according to the color bar. Thus, the perfect function would show a plot that is only dark purple. The red rectangle shows the area the network has trained on.
The loss space of the `Sine` dataset is shown in Figure 5.3. We can clearly see that the sine activation function minimizes the loss best within the training range. Outside of this range, the loss is similar to the sigmoid function. The loss increases

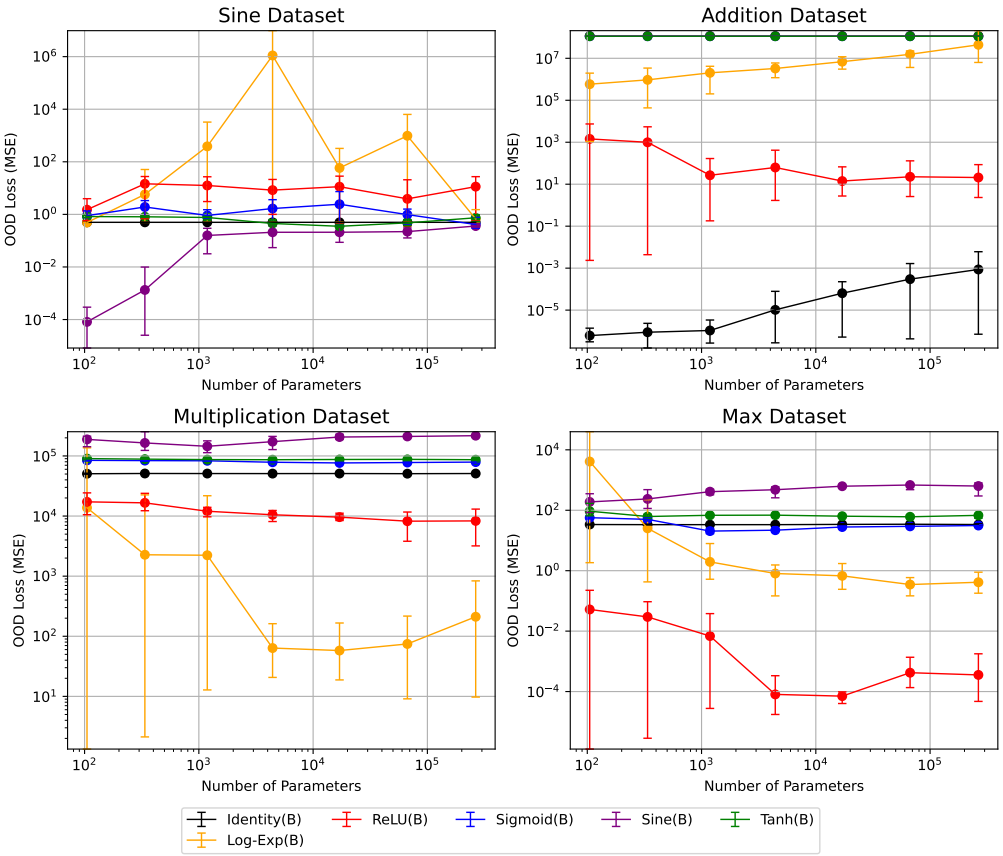

Figure 5.2: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as defined in Section 5.1.1. The graphs show the trend of the MSE loss on the out-of-distribution set as the model complexity increases. Notably, sine, identity, Log-Exp and ReLU activation functions generalize best for `Sine`, `Addition`, `Multiplication` and `Max` dataset respectively. Sigmoid and tanh do not generalize well on any dataset.

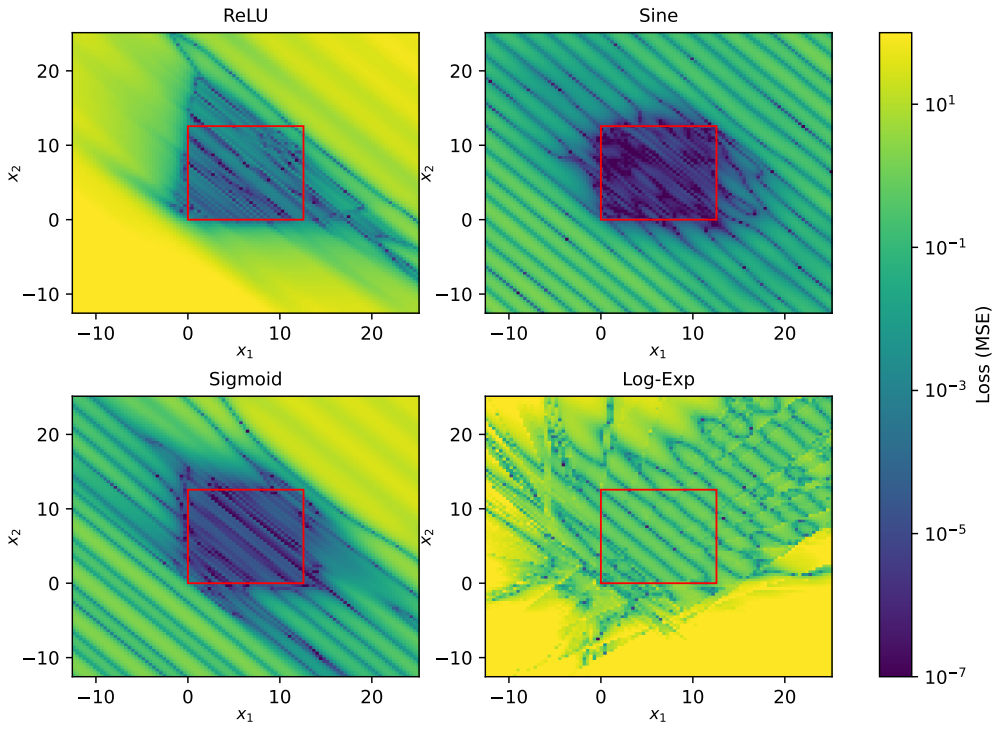

Figure 5.3: MSE loss landscape for four different activation functions on the `Sine` dataset. The network consists of two layers with 128 neurons each. The two axes of each plot represent the two inputs to the network. The loss of the network is color-coded according to the color bar. The red rectangle indicates the area the model was trained on. Notably, the sine activation shows superior performance. The sigmoid and ReLU activation functions also do well. However none of the functions seem to maintain low loss values outside of the training region.

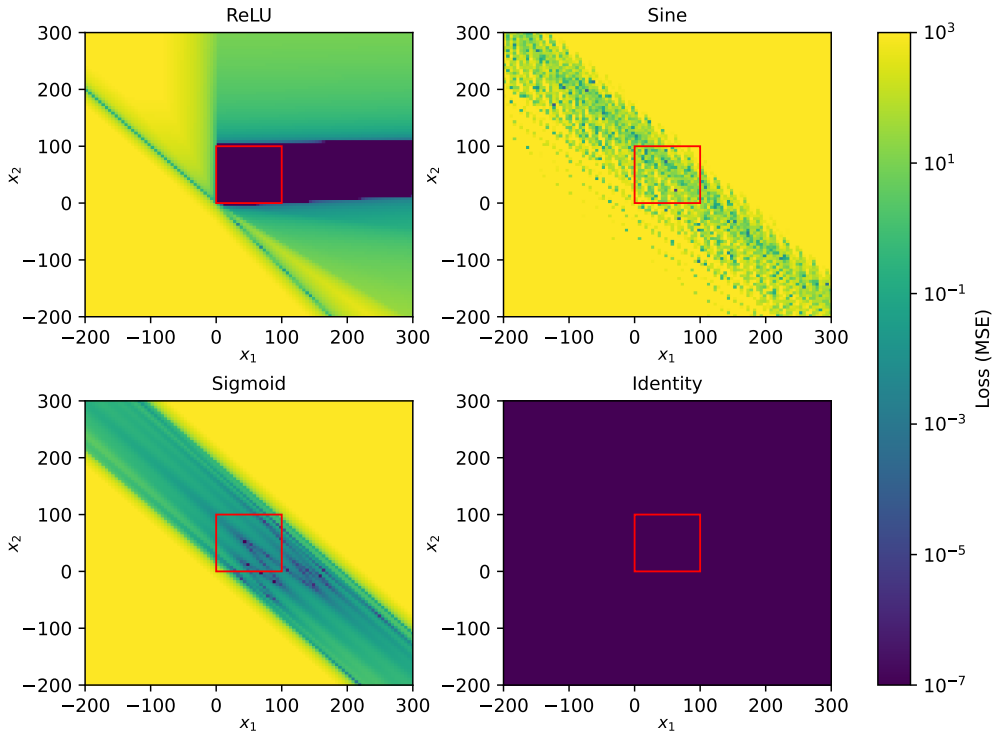

Figure 5.4: MSE loss landscape for four different activation functions on the `Addition` dataset. The network consists of two layers with 8 neurons each. The two axis of each plot represent the two inputs to the network. The loss of the network is color-coded according to the color bar. The red rectangle indicates the area the model was trained on. The Identity activation shows superior performance. Both sigmoid and sine show a diagonal ribbon where the loss is minimized. The ReLU model achieves low loss inside the trained region and displays an ongoing low loss area to the right of it.

by 3 to 5 orders of magnitude. ReLU approximates sine only for the training range and shows poor performance outside the training range, especially for negative inputs. Sigmoid is also able to learn the training domain well, as we can see in figure 5.1. Identity performs poorly, as expected, due to its lack of ability to model non-linear relationships. It becomes interesting when we reduce the size of the network and zoom out a bit in figure B.2.
The minimized error of the sinusoidal activation network is no longer limited to the learning regions, but has extended outward. There is a long strip where the loss remains consistently low. We also see this in Figure 5.2, where the OOD loss of the sine function stays lower for smaller networks. The performance of the sigmoid and ReLU gets worse with fewer parameters. We now look at the `Addition` dataset in figure 5.4. The identity network has no problem adapting to

a wider range of inputs. Its loss stays low over a much wider range than visible. We see this also in figure 5.2, as the OOD loss is about 10^{-6} for the input range 0 to 10000. ReLU adapts really well only to the right of the learning range, but fails in other regions.
As the network grows larger, the ReLU network fans out and covers the entire upper right part of the plane (see figure B.1 in the Appendix B.1).

Sequential (S). Looking at some of the sequential models in Figure 5.5, there are not many big surprises. For the `Sine` dataset, we look at the two layered sequential models ReLU-Sine (S), Sine-Identity (S), and Identity-Sine (S). Sine-Identity (S) and Identity-Sine (S) perform very similarly. Compared to the pure Sine (B) network, the sequential use of Sine and Identity improves the results not only on the test set, but also on the OOD set in Figure B.7 in the Appendix B.2. The loss compared to the test dataset only gets worse by one to two orders of magnitude, with a model that preserves a loss of less than 10^{-11} on OOD data. The difference is particularly noticeable on smaller networks and can be as large as 7 orders of magnitude. This makes sense when considering that in a perfectly constructed network only one neuron with the sine activation function would be needed. Adding more layers and neurons might just add unnecessary noise and complexity to the network. As for the `Addition` dataset, it is interesting to note that Identity-ReLU (S) performs significantly better than ReLU-Identity (S).
Since we were only working with two-layer models, we did not create any new sequential models with the Log-Exp function. Therefore, none of the new models were able to generalize well to the `Multiplication` dataset.

Ensembling (E). In the top left of figure B.8 we see the plot of the `Sine` dataset. The ensemble models are unable to correctly select the right neurons. The Identity-Sine (E) ensemble model does not come close to the sequential model from the last subsection. The ensemble model would have the ability to produce the same low score if it chose to use the Identity neurons from the first layer and the sine neurons from the second layer. However, none of the ensemble models do this. In the `Addition` dataset, it seems to be able to select the right models sometimes, since some of the Identity-Sine (E) ensemble models achieve a similar low score as Identity (B) itself.
As for the `Max` dataset, there are some models that achieve similar results to ReLU. It seems that ReLU-Tanh (E), ReLU-Sine (E), and especially ReLU-Sigmoid (E) are able to learn the `Max` dataset almost as well as the ReLU (B) network.

In contrast to the sequential models, the ensemble models generally adapt better to out-of-distribution data (Fig. B.9) in Appendix B.2. For example, on the `Addition` dataset, we see that the Identity-Sine (E) model has a lower

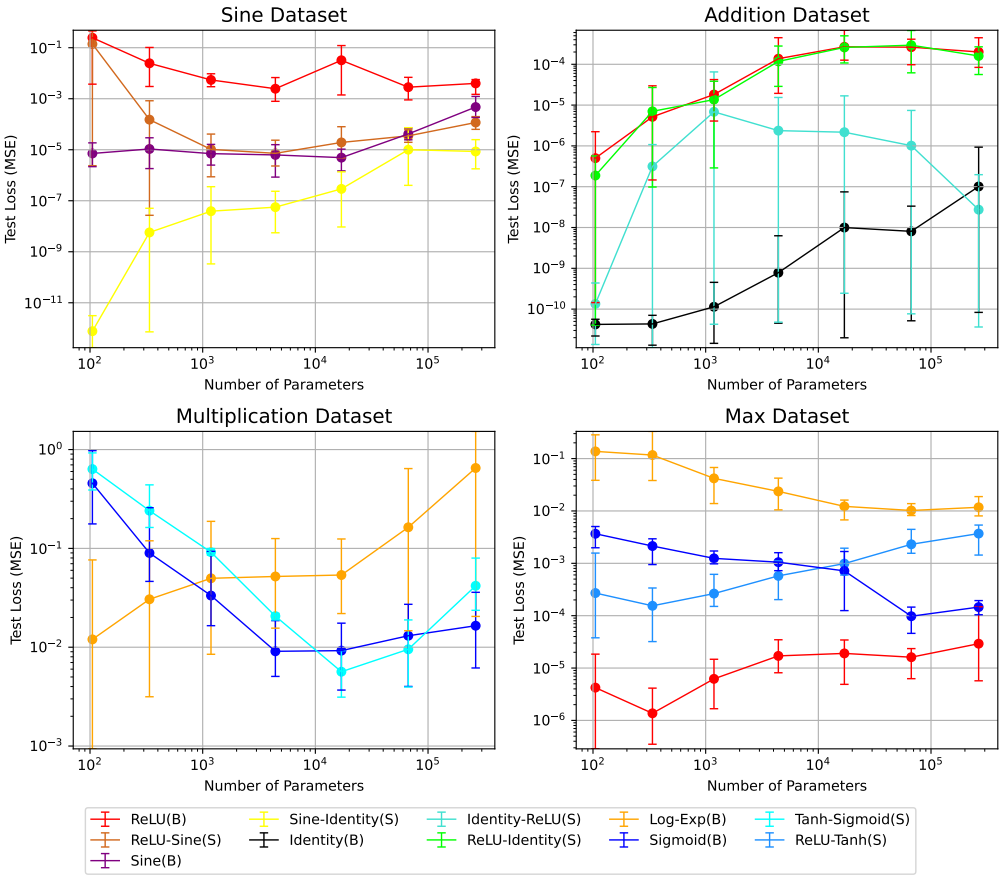

Figure 5.5: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (S) indicates that it is a sequential model with the first layer having a different activation function to the second layer. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases. Notably, on the `Sine` dataset, the Identity-Sine (S) and Sine-Identity (S) model outperform the Sine (B) model. The Identity-ReLU (S) model has some models with equally low score than Identity (B) network on the `Addition` dataset.

OOD error than the ReLU (B) model. Thus, it is less dependent on the sine activation neurons because it does not need to use them. This is best illustrated using the `Max` dataset. There are several combinations that generalize well, such as ReLU-Tanh (E), ReLU-Sigmoid (E), or even ReLU-Sine (E). It appears that the network minimizes the influence of the secondary activation function (in this case, tanh, sigmoid, and sine, respectively).

Random Initialization (R). Again, for numerical stability reasons we omit Log-Exp activation functions. Some of the random models do not perform much worse than the single activation function models (Fig. 5.6). On the `Sine` dataset, we see that the average ReLU-Identity-Sine (R) and ReLU-Identity-Sine-Tanh-Sigmoid (R) models come close to achieving the same score as Sine (B). In some cases, they are more accurate than the sine model by more than an order of magnitude.
On the `Addition` dataset, the Random models perform similarly to the single activation ReLU (B) network, with some outliers in lower loss regions. The `Max` function is learned as well by the ReLU (B) network as by some of the models containing the ReLU function. What should be the main takeaway for this section is that not only do these models learn the function nearly as well, but they also generalize well to out-of-distribution data (Fig. B.10). The baseline performs best, and is expected to perform best.
However, the network is not able to select the identity neurons for the `Addition` dataset, and does so infrequently. On the `Max` dataset, the smaller random networks perform even better than ReLU (B).
Since we did not include Log-Exp activation functions, we do not see good adaptation to OOD data on the `Multiplication` dataset.

Doublelane (D). The Double-Lane model is the Multi-Lane model with two lanes. The plots for the Double-Lane models are shown in Figure 5.7. We see that LogExp-Sine (D), for example, performs well on both the `Sine` and `Multiplication` datasets. Additionally, it adapts similarly well to out-of-distribution data as the Log-Exp (B) model. On the `Sine` dataset, we see that the Double-Lane models with the sine activation function are within an order of magnitude of the pure sine model. Only a few instances of the models fit the out-of-distribution range (see Figure B.11). Similar things happen on the `Addition` dataset. Although the Identity-Sine (D) model does not come close to the Identity (B) model on the test set, it still adapts quite well to the OOD samples (see Figure B.11). This is not the case for all ReLU-Identity (D) models. Some of the models have a score as low as Identity (B), but the average performance of ReLU-Identity (D) is closer to ReLU (B) than to Identity.
On the `Multiplication` dataset it is the LogExp activation combined with sine and ReLU that performs just as well as the LogExp function. Larger networks achieve an even lower test loss. Again, both models, LogExp-ReLU (D) and

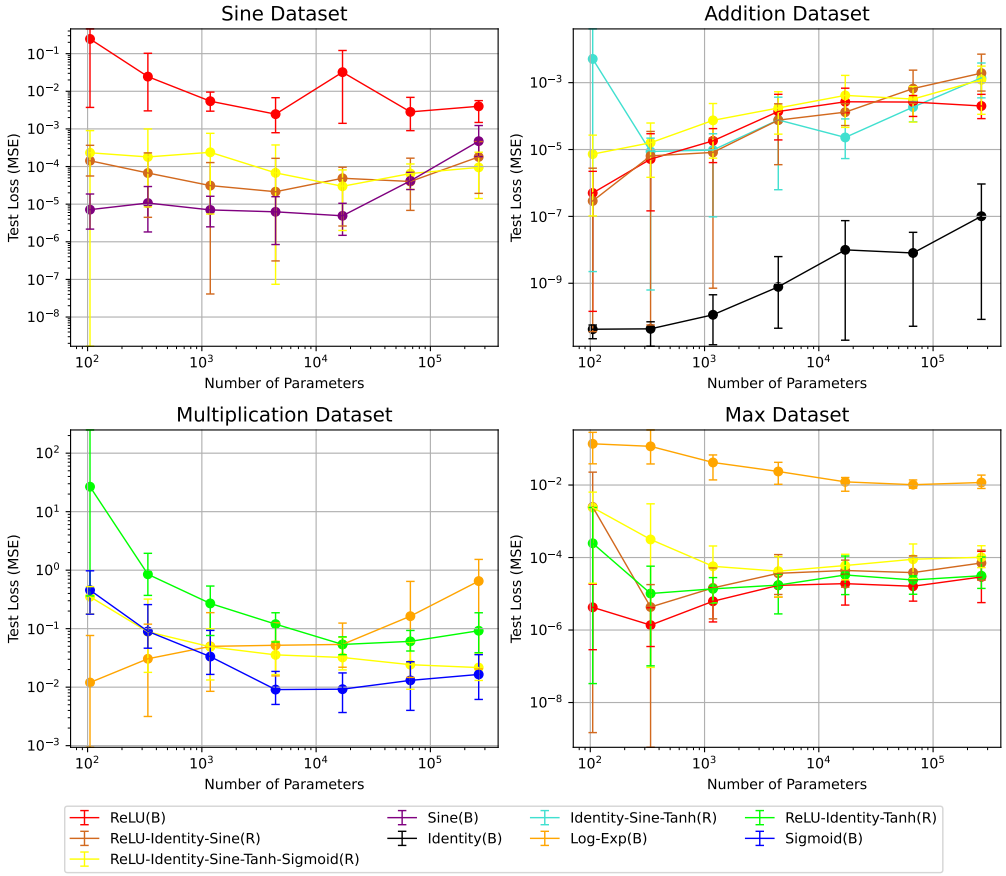

Figure 5.6: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (R) indicates that it is a random model, meaning that for every neuron in the network a random activation function is chosen from the given set. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases. Similar to the ensembling models, only certain instances of random models (R) are performing as strongly as the pure models (B). However on the `Max` dataset, various random models (R) score equally as low as the ReLU (B) network.

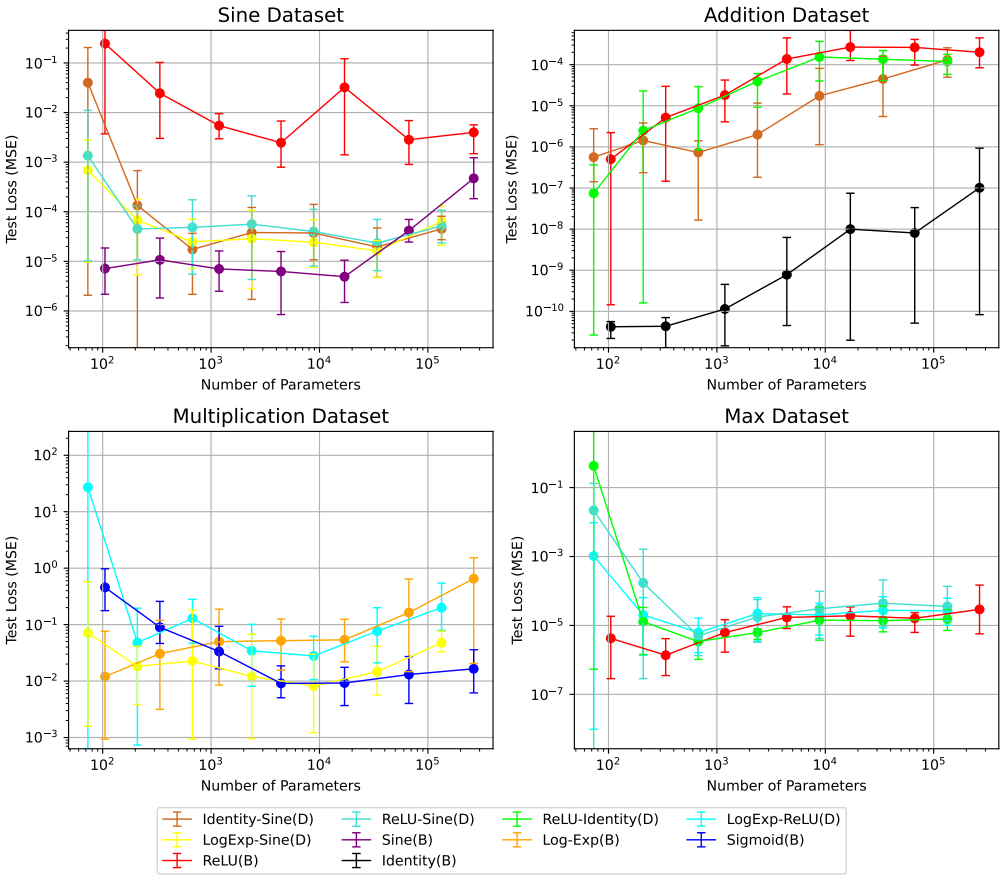

Figure 5.7: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (D) indicates that it is a Doublelane model, consisting of two different lanes of neurons (see Figure 4.4). (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases. Notably, on the `Multiplication` dataset, LogExp-Sine (D) outperforms Log-Exp (B) consistently. Among larger models, LogExp-ReLU (D), ReLU-Sine (D) and ReLU-Identity demonstrate the ability to compete with the results ReLU (B) achieved.

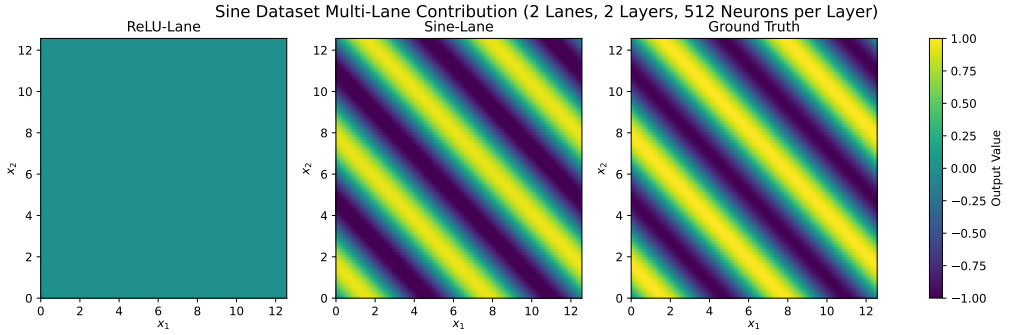

Figure 5.8: Comparison of the two lanes within the Double-Lane model that is trained on the `Sine` dataset. The x-axis represents the first input while the y-axis represents the second input. The values are color-coded according to the color bar and show the contribution of the lanes to the final output. The first plot shows the contribution of the ReLU lane of the network. The subsequent plot displays the contribution of the sine lane. The third plot shows the output labels for the input 1 and 2. Notably, the sine lane carries the structure of the output, whereas the ReLU lane contributes, at most, a constant summand.

LogExp-Sine (B) adapt similarly well to unseen out-of-distribution ranges as the LogExp (B) function itself. The `Max` function is learned equally satisfactorily by the Double-Lane models. The test score is just as good, and for some dimensions of the network the doublelane performs even better than ReLU (B).
We notice that many Double-Lane models such as LogExp-ReLU (D), LogExp-Sine (D) or ReLU-Sine (D) do not necessarily worsen the performance of the model. The model is able to adjust the weights and biases accordingly so that not only the test performance, but sometimes also the OOD performance is maintained.
To find out more about the nature of the Double-/Multi-Lane models, we want to visualize the output of each of the lanes. We calculate the contribution of the two lanes to the output by manually multiplying the activations with the weights of the output layer. The following figures show us which lane contributes how much to the output and compare it with the ground truth of the dataset. Other datasets and combinations can be found in the Appendix B.3. In Figure 5.8 we see the contribution of the ReLU and the sine lane on the `Sine` dataset. The sine lane carries almost the entire structure, while the ReLU lane contributes at most a constant. However, this changes when we combine the ReLU function with the Identity function in Figure B.24.
The network "realizes" that the output can only be learned via the ReLU path, and therefore chooses to build the structure in the ReLU part of the network. Furthermore, when the ReLU and sine activation functions are combined on the `Max` dataset in Figure 5.9, the network structure is mainly achieved through the ReLU lane. This shows how well the MLP determines which lane is more valu-

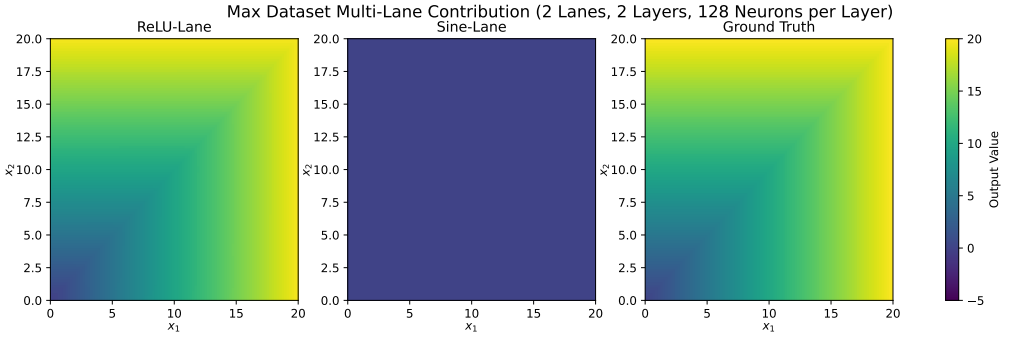

Figure 5.9: Comparison of the two lanes within the Doublelane model that is trained on the `Max` dataset. The x-axis represents the first input while the y-axis represents the second input. The values are color-coded according to the color bar and show the contribution of the lanes to the final output. The first plot shows the contribution of the ReLU lane of the network. The subsequent plot displays the contribution of the sine lane. The third plot shows the output labels for the input 1 and 2. Notably, the ReLU lane is solely responsible for the structure of the output. The sine lane contributes, at most, a constant summand.

able for the ability to learn the output labels. More heatmaps can be found in the Appendix A.

To confirm our observations, we additionally examine the gradients of the loss function with respect to activations. We would expect the gradients to be greater for the lane that is more important. We calculate the derivative of the loss functions with respect to the activations:

$$\frac{\partial \mathcal{L}_{\mathcal{MSE}}}{\partial a_i^{(L)}}, \tag{5.1}$$

where $\mathcal{L}_{\mathcal{MSE}}$ is our loss function and a_i is the i-th activation of the last layer L. We find the mean gradient magnitude by averaging the absolute values of all the gradients of the lane. Therefore, the average gradient magnitude of all neurons in a lane of a Double-Lane model is:

$$\overline{g} = \frac{1}{\frac{n}{2}} \sum_{i=1}^{\frac{n}{2}} |\frac{\partial \mathcal{L}_{\mathcal{MSE}}}{\partial a_i^{(L)}}|, \tag{5.2}$$

with n being the number of neurons in the last layer L.

In Figure 5.10 we compare the average gradient of the ReLU lane and the Identity lane on the `Sine` dataset. As expected, the average magnitude of the gradient is greater on the ReLU lane for all network sizes. This indicates that the ReLU lane is more important to the network. Using the same model for the `Max` dataset (see Figure 5.11), the ReLU has larger activation gradients. This correlates with our visualized loss landscapes in the previous section. Unfortunately, this is not true

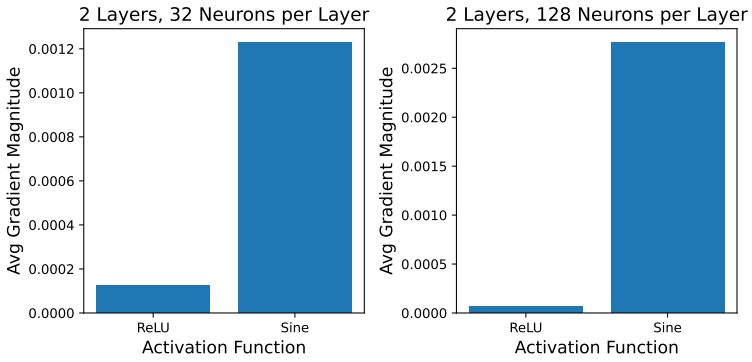

Figure 5.10: Comparison of the gradient magnitude for two different lanes within a Double-Lane model. The compared plots were trained on the `Sine` dataset, with each one of the models having a different size. The x-axis represents the activation functions of the two lanes. The y-axis indicates the average gradient magnitude of the activations in a lane. Notably, the gradient of the ReLU activations with respect to the loss is greater than the gradients of the identity activations on all the models.

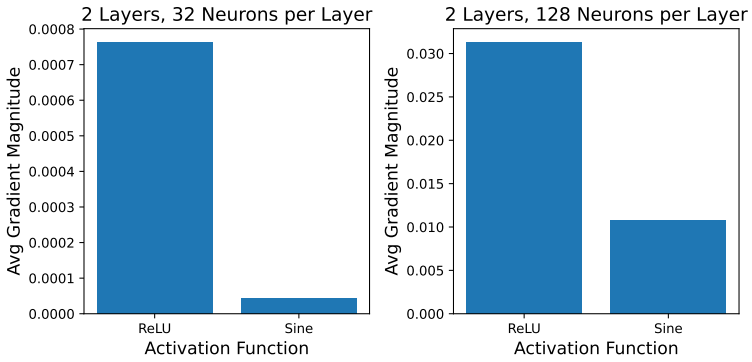

Figure 5.11: Comparison of the gradient magnitude for two different lanes within a Double-Lane model. The compared plots were trained on the `Max` dataset, with each one of the models having a different size. The x-axis represents the activation functions of the two lanes. The y-axis indicates the average gradient magnitude of the activations in a lane. Notably, the gradient of the ReLU activations with respect to the loss is consistently greater than the gradients of the sine activations on all the models.

for all datasets and activation combinations. For example, on the `Max` dataset, we see in Figure B.32 in the Appendix B.4 that although the ReLU lane is the more important lane for structure, the gradient of the Identity lane is larger.
The gradient of the loss function with respect to the activations suggests some correlation with the relevance of the neurons. However, this is not true for all models and Multi-Lane combinations.

Triple-Lane (T). We first look at the LogExp-Identity-Sine (T) Triple-Lane model in Figure 5.12. It performs similarly to Sine (B) on the test data. On the `Multiplication` dataset, the model achieves excellent results, especially on the test data, but only a mediocre score for the generalization domain. The LogExp-ReLU-Sine (T) model performs similarly to the LogExp-Identity-Sine (T) model on the `Sine` and `Multiplication` dataset. It shows acceptable results on the `Max` dataset. It performs similarly to the ReLU model on the test set. For out-of-distribution ranges the Triple-Lane model manages to score better on smaller networks, while scoring worse on networks with more parameters (see Figure B.12 in the Appendix B.2).
The third Triple-Lane model, ReLU-Identity-Sine (T), displays almost the same results on `Sine`, `Max` and `Addition` datasets as Sine (B), ReLU (B) and ReLU (B) respectively.

Quad-Lane (Q). For the Quad-Lane models, we will focus on the LogExp-Identity-Sine-ReLU (Q) model, as it would theoretically have the composition of activation functions to learn all data sets as perfectly as the baseline models. The sine function is well learned (see Figure B.13). The test loss on the `Multiplication` and `Max` datasets is no worse than Log-Exp (B) and ReLU (B), respectively. Larger networks perform well within training bounds. However, they do not fit out-of-distribution as well as Log-Exp (B). On the `Max` dataset, the model achieves solid scores when tested with **on**-distribution-data and smaller networks also generalize well to OOD ranges. Other Quad-Lane models achieve similar scores. For more information see Appendix B.2.

Modified Multi-Lane (MM). The performance of the Modified Multi-Lane is shown in Figure 5.13. Identity-Sine (MM) outperforms all other models in this comparison. Similar to the sequential Identity-Sine (S) model, the use of more than one sine layer may only add unnecessary noise and complexity to the model. Identity-Sine (MM) also performs well on the `Addition` dataset. It is the only model we have seen that comes close to the performance of Identity (B). The ReLU modified Multi-Lane models, such as ReLU-Sigmoid (MM) or ReLU-Tanh (MM), seem to have no problem learning the dataset as well as ReLU (B). As noted above, the Identity-Sine (MM) models perform well on out-of-distribution data (see B.15 in Appendix B.2). ReLU-Sine (MM) has some instances that have

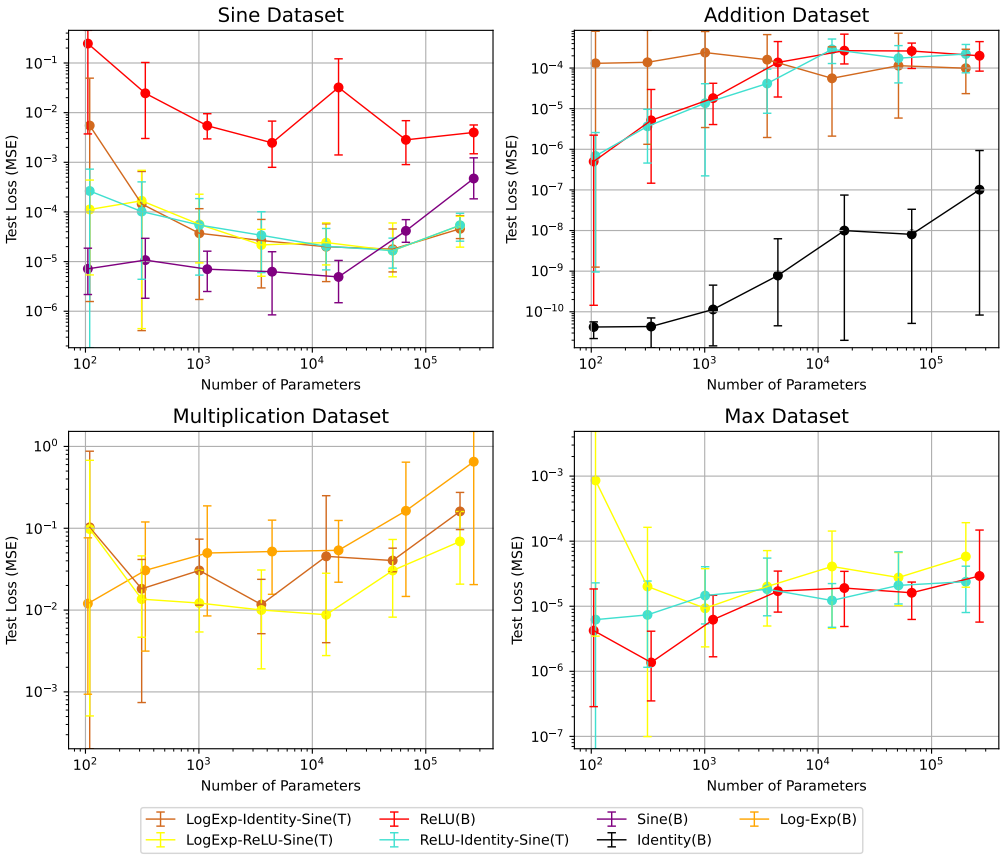

Figure 5.12: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (T) indicates that it is a Triplelane model, consisting of three different lanes of neurons. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases. LogExp-Identity-Sine (T) and LogExp-ReLU-Sine (T) perform superior to the Log-Exp (B) models on the `Multiplication` Dataset. Additionally LogExp-ReLU-Sine (T) performs also decent on the `Max` dataset.

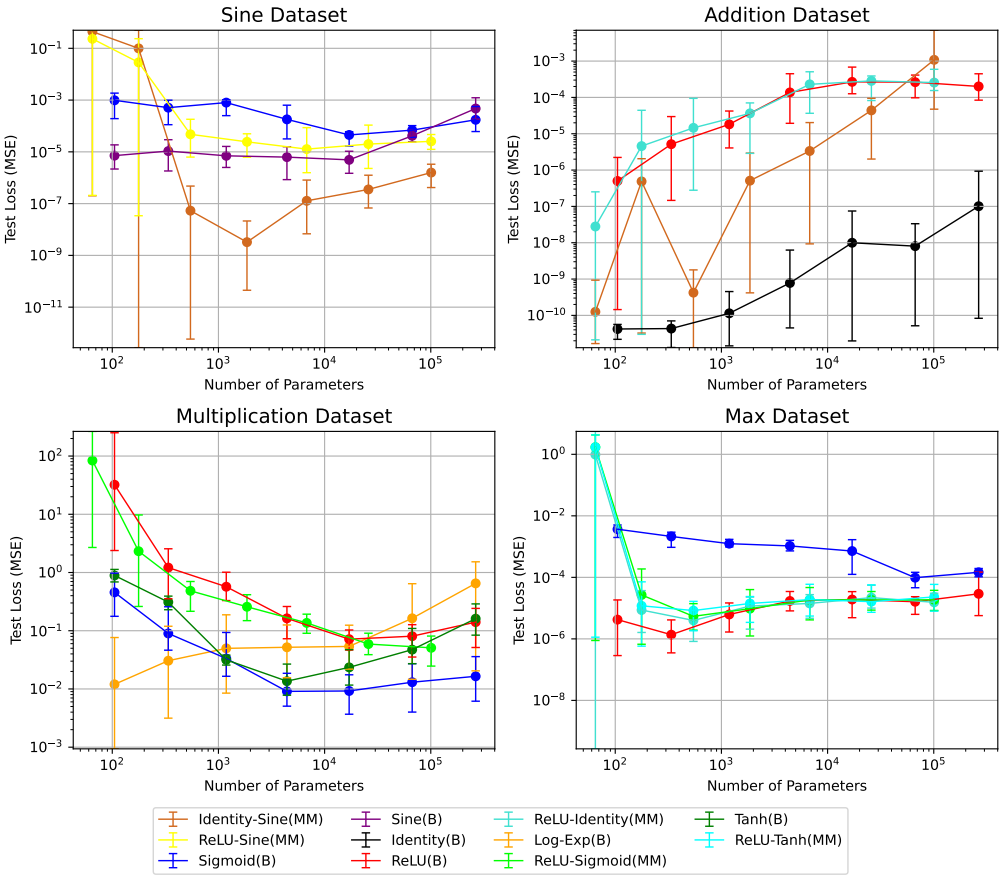

Figure 5.13: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (MM) indicates that it is a modified Multi-Lane model. It has a structure as described in Figure 4.5. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases. Notably, the Identity-Sine (MM) model performs superior to the other models on the `Sine` dataset. This model also performs better than any other model, except for the Identity (B) model, on the `Addition` dataset.

the same generalization loss as the Sine (B) model. On the `Addition` dataset, both Identity-Sine (MM) and Identity-Tanh (MM) perform significantly better than ReLU (B). Identity-Sine (MM) is almost as good as Identity (B) for some network sizes. ReLU-Sine (MM) and ReLU-Identity (MM) achieve similar OOD loss on the `Max` dataset, with better performance on larger models.

5.2.2 Real-World Datasets

To keep the focus on our synthetic datasets, we move the figures to the appendix. For a quick overview there is a paragraph at the end of this section. It concisely summarizes the best performing models. The figure of these models follows this summary. For plots of all the other analysis we refer to the Appendix A.

Baseline (B). For all real-world datasets we see similar, somewhat expected behavior in Figure B.16. ReLU performs best on all datasets. It is only really challenged by the sine activation function on the ISOLET and FashionMNIST datasets. On the MNIST and ISOLET datasets, tanh and sigmoid achieve similar results, unlike on the other datasets. FashionMNIST and CIFAR10 are learned better by tanh than by sigmoid. As a reference, the difference between the loss of ReLU and the loss of tanh on the FashionMNIST dataset results in about a 1% difference in accuracy between them.

Sequential (S) and Ensembling (E). All of the models mentioned and plotted now contain a ReLU layer. Other models that do not contain ReLU either did not show any improvement over single activation models or did not show any improvement worth mentioning.
The MNIST dataset still learns best with the ReLU activation function (see Figure B.17). However, we see that sequential ReLU-Tanh (S), ReLU-Sigmoid (S), and Tanh-ReLU (S) all perform similarly or only slightly worse.
FashionMNIST matches this behavior, with ReLU-Tanh (S) and ReLU-Sigmoid (S) close to ReLU (B), but still inferior.
On the ISOLET dataset we actually see a really interesting property. Not only is ReLU-Sigmoid (S) able to catch up to ReLU (B), but it actually outperforms ReLU (B) by quite a bit. This is fascinating when comparing the ReLU-Sigmoid (S) model to the normal Sigmoid (B) and ReLU (B) models. Sigmoid, for example, performs much worse when used alone. In terms of accuracy, the difference is 0.44%.
Unfortunately, on the CIFAR10 dataset, ReLU-Sigmoid (S) does not perform nearly as well. In this case, it is ReLU-Tanh (S) that performs as well as the single ReLU (B) network, although the single Tanh (B) model performs poorly.
Ensembling two activation functions into one model on the MNIST dataset gives similar results to the sequential models (see Figure B.18). ReLU-Tanh (E) and

ReLU-Sigmoid (E) achieve almost the same loss as ReLU (B). On the FashionMNIST, ISOLET, and CIFAR10 datasets, we have a new, unexpected competitor. ReLU-Sine (E) performs incredibly well on all of them. Its performance is roughly equal to that of the single activation ReLU (B) network. This suggests that either the network is "intentionally" missing the sine neurons, or the sine part of the network is actually helping to extract features from the input space.

Random (R). Many of the random models perform similarly to the ReLU models (see Figure B.19). For example, on the ISOLET dataset, ReLU-Tanh-Sigmoid (R) outperforms ReLU (B) for some network sizes. On the FashionMNIST dataset, the ReLU-Tanh-Sine (R) model achieves a lower average loss than ReLU (B).

Double-Lane (D). On the real-world datasets we see, in Figure B.20, a similar behavior on all of them. The ReLU-Sigmoid (D) and ReLU-Tanh (D) Double-Lane models do not perform quite as well, except for the ISOLET dataset, for which the combination seems to improve larger networks. Interestingly, the ReLU-Sine (D) combination is actually preferred on FashionMNIST and the ISOLET dataset. Other combinations, such as Sigmoid-Tanh (D), did not perform noteworthy. The Appendix B.2 contains more results.

Triple-Lane (T). The Triple-Lane models perform well on the real-world datasets (see Figure B.21). On the MNIST dataset, both ReLU-Tanh-Sigmoid (T) and ReLU-Tanh-Sine (T) perform similarly to ReLU (B). ReLU-Tanh-Sine (T) outperforms ReLU (B) on both FashionMNIST and ISOLET, especially for larger models. On the ISOLET dataset, ReLU-Tanh-Sigmoid (T) also achieves lower loss than ReLU (B). ReLU is preferred on all datasets for smaller number of parameters.

Quad-Lane (Q). The performance of the Quad-Lane model is shown in Figure B.22. The Quad-Lane model outperforms the ReLU (B) model in three out of four cases. On the MNIST dataset, the Multi-Lane achieves only a slightly better loss. The same is true for the FashionMNIST dataset. On the ISOLET dataset, the Quad-Lane performs significantly better than ReLU (B). On CIFAR10, for the model sizes where both models were tested, the ReLU model still outperforms the ReLU-Tanh-Sigmoid-Sine (Q) model. ReLU (B) is consistently best for smaller networks.

Modified Multi-Lane (MM). On real-world datasets, the smaller modified Multi-Lane models generally perform poorly compared to ReLU (B) (see Figure B.23). However, as the networks become larger, the performance approaches that

of ReLU (B). The loss is smaller for ReLU-Sine (MM) on the ISOLET dataset.

Overview. In addition to the synthetic dataset, it is interesting to see how the activation ensemble models perform on real-world datasets. Instead of presenting all models individually, we limit the plots to the best performances. We plot the three best performing models for each dataset in Figure 5.14. We also provide the performance of ReLU (B) and Tanh (B) for all datasets as a reference.
Interestingly, neither the Random (R), nor the Ensemble (E) models are able to keep up with the best networks. There is one Sequential (S) model, the ReLU-Sigmoid (S), that is able to heavily outperform all other models on the ISOLET dataset. All other models are of the Multi-Lane structure. For example the ReLU-Tanh-Sine (T) Triple-Lane model is ranked along the top performing datasets in three (MNIST, FashionMNIST and ISOLET) out of the four datasets. For a more detailed analysis, we refer to the Appendix B.2.2.

5.3 Lane-Loss

5.3.1 Model Architecture

Instead of using a normal Multi-Lane model, we will use the Double Quad-Lane model (see Figure 4.6). Because we are using datasets with nested mathematical functions of level two (see Subsection 5.3.2), this should theoretically allow the network to better learn the underlying function.

5.3.2 Datasets

We test our entropy loss on other synthetic datasets. The datasets we create are made out of two nested mathematical operations. The datasets are defined as in Table 5.3, with the given splits and ranges. Additionally, **Poly-v1** is defined as

$$y(x_1, x_2) = 3x_1^2 - 2x_1x_2 + 4x_2^2 \tag{5.3}$$

and **Poly-v2** is defined as

$$y(x_1, x_2) = x_1^2 + \frac{x_1x_2}{3}. \tag{5.4}$$

5.3.3 Criterion: Weight Squares

We describe the experiments for the Lane-Loss model as in Chapter 4.3. We inspect the lanes the model chooses, disregarding the achieved best score. To keep

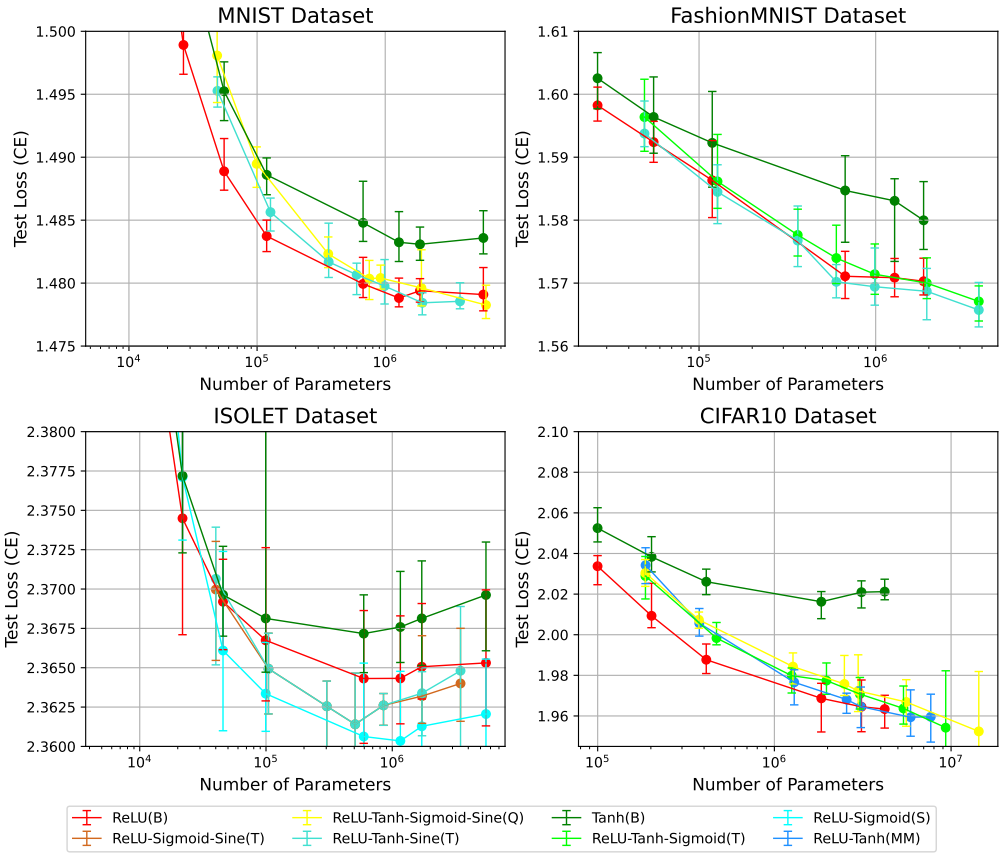

Figure 5.14: Comparison of the model performance on the test set relative to the number of parameters for four different datasets (MNIST, FashionMNIST, ISOLET, CIFAR10), as defined in Section 5.1.1. The graphs show the trend of the cross entropy loss on the test set as the model complexity increases. (S) indicates that it is a sequential model with the first layer having a different activation function to the second layer. (B) marks a pure, single activation function model. (T), (Q) and (MM) stand for Triple-, Quad-, and Modified-Multi-Lane models, respectively (as defined in 4). The graphs show the trend of the cross entropy loss on the test set as the model complexity increases. Notably, most well performing models rely at least partially on ReLU activation function. The ReLU-Sigmoid (S) network does achieve lower losses than ReLU (B) on the ISOLET dataset.

Dataset	Train/Val/Test/OOD Samples	TVT Range	OOD Range
$\sin x_1 + \cos x_2$	7000/1500/1500/10000	[0, 4π]	[0, 8π]
$\sin x_1 \cdot \sin x_2$	7000/1500/1500/10000	[0, 4π]	[0, 8π]
Poly-v1	7000/1500/1500/10000	[0, 20]	[0, 40]
Poly-v2	7000/1500/1500/10000	[0, 10]	[0, 20]
$\max(\sin x_1, \sin x_2)$	7000/1500/1500/10000	[0, 4π]	[0, 8π]

Table 5.3: Datasets used for the Lane-Loss experiments. The datasets are defined as in column *Dataset*. Additionally textttPoly-v1 and textttPoly-v2 are defined as in Equation 5.3 and 5.4. The second column shows the number of samples for training, validating, testing and out-of-distribution (OOD). TVT Range shows the range of the training, validation and testing samples, whereas OOD range stands for the range of the out-of-distribution data.

the chapter concise, we only present models with 32 neurons per layer. Figure 5.15 shows the selected lanes for both the first Multi-Lane module and the second Multi-Lane module. The model is trained on the dataset $sin(x_1) + cos(x_2)$ and uses 32 neurons per layer, resulting in eight neurons per lane per layer. It shows that both ReLU and sine tracks dominate over identity and sigmoid. One could argue that the model should choose a sine function for the first module and an identity function in the second module for better performance. It seems to choose the ReLU function instead of Identity. For the $sin(x_1) \cdot sin(x_2)$ dataset, we swap the Identity lanes with LogExp lanes to give the model a chance at a good multiplication activation function. The model's choices are shown in Figure 5.16. The model does not choose the LogExp lane and instead combines the sine and ReLU lanes. In particular, for larger models, the networks end up heavily selecting the sigmoid lane, followed by a ReLU lane. But again, LogExp remains unused. Next, we look at the $max(sin(x_1), sin(x_2))$ dataset. This should optimally choose the sine lane, followed by a ReLU lane. As we see in Figure B.33 in Appendix B.5, that the model roughly does so. It favours the sine lane in the first part of the network and then picks either ReLU or Sigmoid. Divergent behaviour is visible for the textttPoly-v1 dataset. Again, we would like to see the model choosing LogExp, to optimally rebuild the underlying function. However the model clearly functions differently. Figure 5.17 shows, that the model not only fails to choose LogExp, but it also almost exclusively picks the sine lane. The network also fails to perform as expected on the last dataset, textttPoly-v2. This time, the model is indecisive on the first Multi-Lane module. 5 out of 10 models chose the sine lane, the other 5 chose the ReLU lane. However, all models chose the ReLU lane as the second lane.
For some models and datasets, this approach may work just fine. The model often heavily favours the ReLU and the sine lane.This approach does not seem to work optimally.

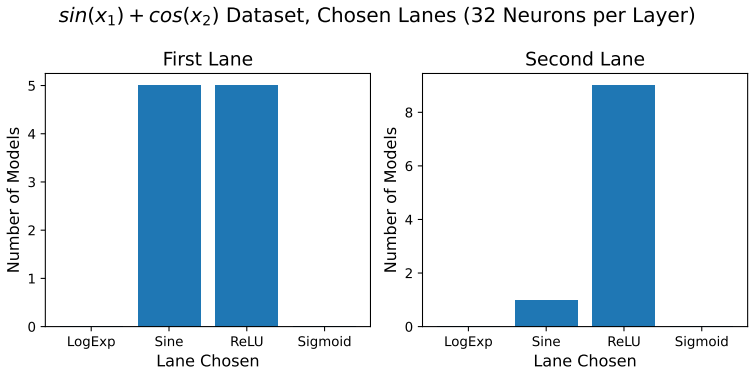

Figure 5.15: Shows the lanes chosen from the model with Lane-Loss. There were ten runs evaluated. The x-axis shows the lane that was chosen. The y-axis indicates how often the lanes were selected. This model type was trained on the $sin(x_1) + cos(x_2)$ dataset. The model has 32 neurons per layer. Notably, the first Multi-Lane module selects both the sine and the ReLU lane. The second Multi-Lane module primarily chooses the ReLU lane.

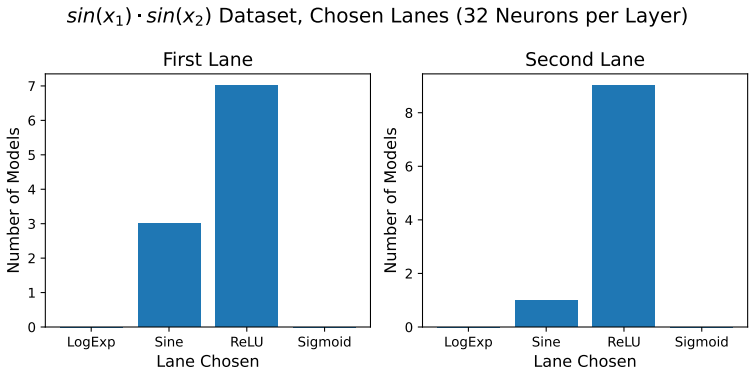

Figure 5.16: Shows the lanes chosen from the model with Lane-Loss. There were ten runs evaluated. The x-axis shows the lanes that were chosen. The y-axis indicates how often the lanes were selected. This model type was trained on the $sin(x_1) \cdot sin(x_2)$ dataset. The model has 32 neurons per layer. Notably, the first Multi-Lane module selects both the sine and the ReLU lane. The second Multi-Lane module primarily chooses the ReLU lane.

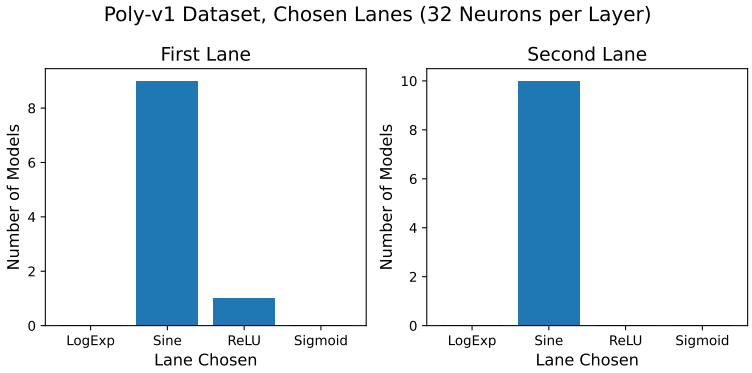

Figure 5.17: Shows the lanes chosen from the model with Lane-Loss. There were ten runs evaluated. The x-axis shows the lanes that were chosen. The y-axis indicates how often the lanes were selected. This model type was trained on the texttttPoly-v1 dataset (see Table 5.3). The model has 32 neurons per layer. Notably, the first Multi-Lane module selects primarily the sine lane. The second Multi-Lane module primarily chooses the sine as well.

The test/OOD score is still calculated from the models with the least validation losses. However, there are some things to consider. Because we are using the models with the best validation loss, the model most often still uses all of the lanes at least a bit. Furthermore, since we let the model converge to lowest lane-loss as well, we let the training process continue for all of the 200 epochs (remember that we used early stopping for most other models). This generally also allows the model to get lucky more often.
We compare this model with single, pure activation function models with 4 layers.
On $sin(x_1) + cos(x_2)$ and on $sin(x_1) \cdot sin(x_2)$, the sine models perform best for smaller parameter count. However, the double Quad-Lane models consistently perform well on all datasets. On three of the four datasets in Figure 5.18, the DQL models achieve the lowest score. On the OOD range, the DQL models show some instances with low loss. However, the average of the models does not indicate a (better) generalization (see Appendix B.5).

5.3.4 Criterion: Contribution

We now experiment with the contribution criterion. It is explained in Section 4.3. We first look at the selected lanes in Figure 5.19. It shows the chosen lanes of the first and the second Multi-Lane module on the textttPoly-v1 dataset. In contrast to the lane-loss with weights criterion, this model now does choose the Log-Exp lane. The first module exclusively selects the LogExp lane, while the second module chooses it 7 out of 10 times. On the $max(sin(x_1), sin(x_2))$ dataset (see Figure B.33), the networks choose the sigmoid lane nine out of ten times for the output lane. Whether or not this is related to the sigmoids similarity to sine around its origin is not further explored. However, the model select the sine lane in the first module 7 out of 10 times, which is what we would expect. For the third dataset we look at the $sin(x) \cdot sin(y)$ dataset in Figure 5.20. For this dataset, the double Quad-Lane model exclusively chooses Log-Exp and ReLU lanes. One would expect the sine lane to be selected more often, because it is the only activation function with periodicity.
The performance of the model with the best validation loss is shown in Figure 5.21. The DQL models with lane-loss perform consistently better or at least equal to the baseline models. The variance of the models is somewhat higher, but not excessively. The performance of the double Quad-Lane models is particularly good with higher parameter count.
As can be seen in the next section, the good performance does not necessarily have something to do with the Lane-Loss approach. The model certainly performs superior because of the different activation functions used in the network. More figures are found in the Appendix B.5.

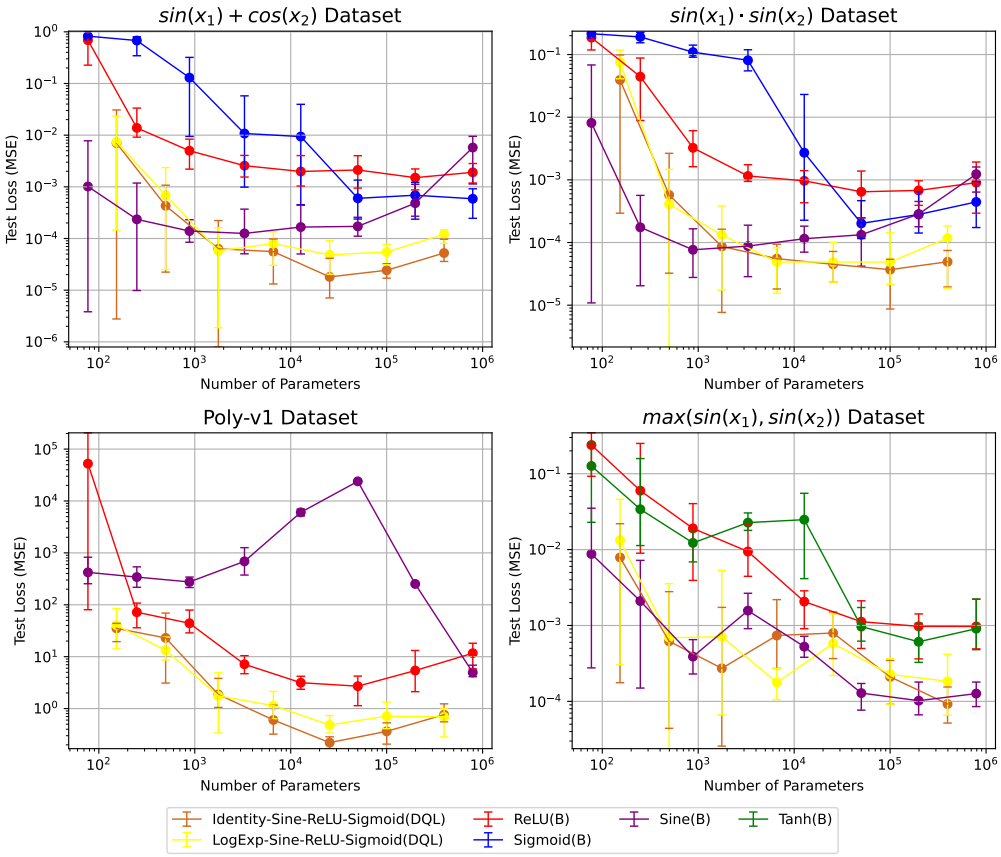

Figure 5.18: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets as described in Section 5.3.2. (DQL) indicates that it is a Double Quad-Lane model with Lane-Loss and the **weight** criterion. It has a structure as described in Figure 4.6. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases. Notably, on the upper datasets, the Sine (B) models achieve the lowest scores for smaller number of parameters. However, the more parameters the DQL networks have, the better it performs. It performs best on three of the four datasets.

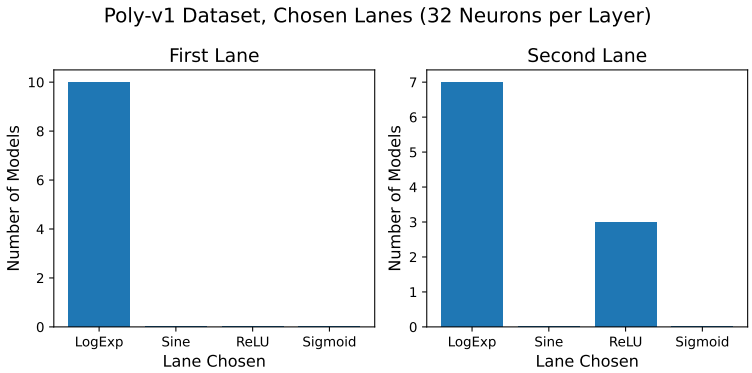

Figure 5.19: Shows the lanes chosen from the model with Lane-Loss and the contribution criterion. There were ten runs evaluated. The x-axis shows the lanes that were chosen. The y-axis indicates how often the lanes were selected. This model type was trained on the textttPoly-v1 dataset (see Table 5.3). The model has 32 neurons per layer.

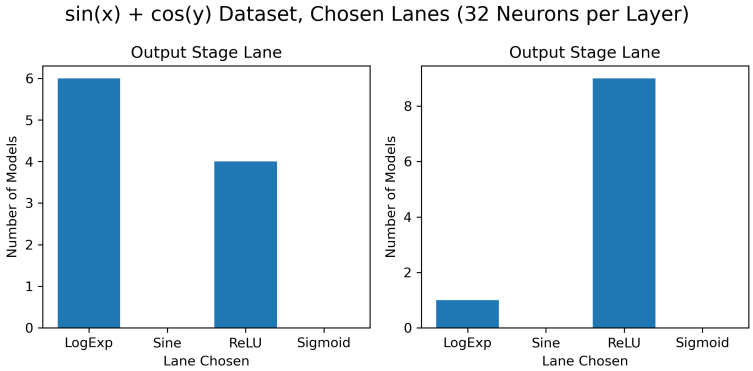

Figure 5.20: Shows the lanes chosen from the model with Lane-Loss and the **contribution criterion**. There were ten runs evaluated. The x-axis shows the lanes that were chosen. The y-axis indicates how often the lanes were selected. This model type was trained on the textttPoly-v1 dataset (see Table 5.3). The model has 32 neurons per layer.

5.3.5 Eliminating Lanes

The Lane-Loss with the contribution criterion minimizes the impact of three of the four lanes. Assuming the impact of one of these lanes is actually close to zero, one could think that it only contributes as a remnant. This would mean, that eliminating these "unused" lanes would improve performance. We therefore delete these lanes and evaluate the test loss. Unfortunately this did not work as intended. The average contribution magnitude of the neurons of different lanes are shown in Figure 5.22. The model of the figure has learned $sin(x) \cdot sin(y)$ and is evaluated with an input sample (9.0, 3.0). This initially indicates good learning from the implemented entropy loss. However, when considering the contribution of the whole lane and not the neurons, we get Figure 5.23. Even though the values from the sine lane have higher magnitude, they get canceled out by each other. Identity lane does not show large positive or negative values. However, the values are all negative, resulting in a large contribution towards the output. For future experiments it might be better to minimize the whole contribution of the lanes, instead of the contribution of individual neurons of a lane.

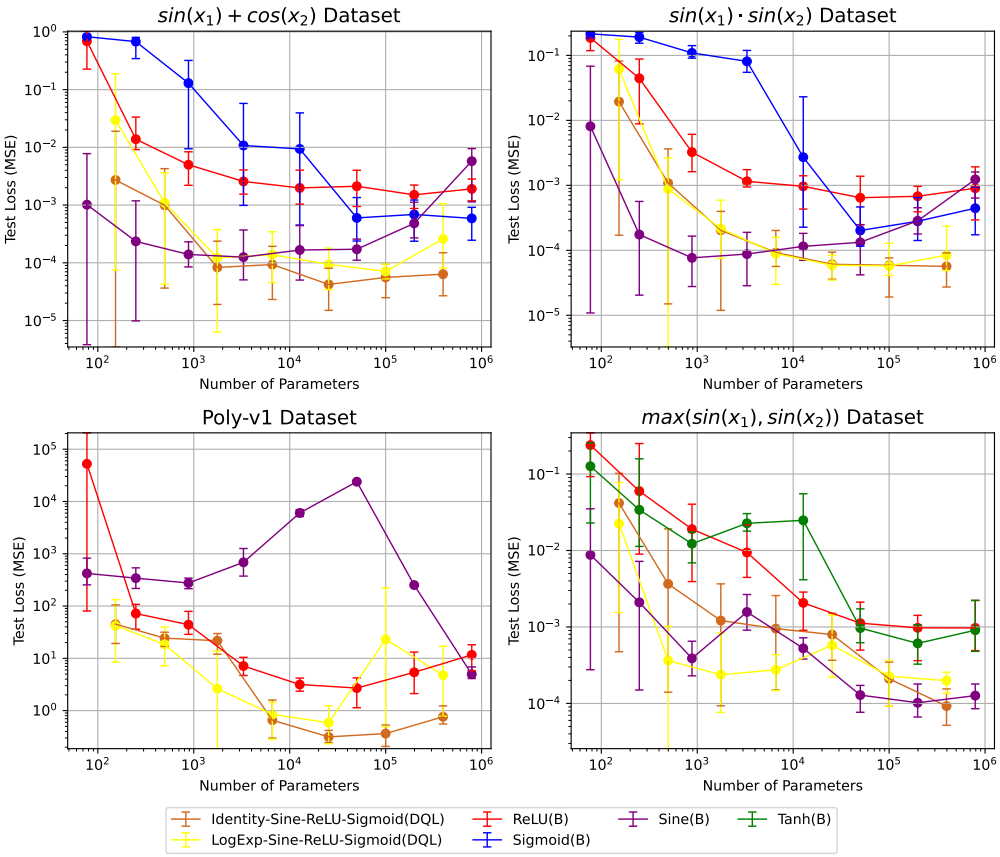

Figure 5.21: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets as described in Section 5.3.2. (DQL) indicates that it is a Double Quad-Lane model with Lane-Loss and the **contribution** criterion. It has a structure as described in Figure 4.6. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases. Notably, on the upper datasets, the Sine (B) models achieve the lowest scores for smaller number of parameters. However, the more parameters the DQL networks have, the better it performs. It performs best on three of the four datasets.

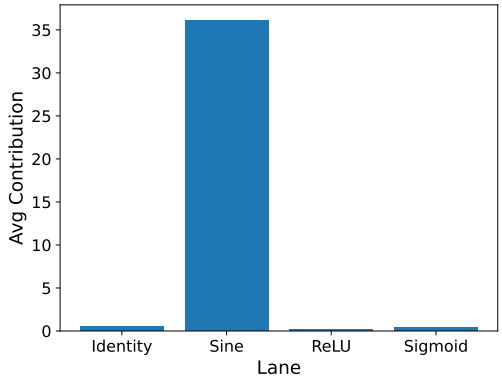

Figure 5.22: This shows the average contribution magnitude of the neurons in the specific lane. The x-axis shows the lane. The y-axis represents the average contribution (magnitude) of a neuron in the specific lane. This indicates good minimizing of the lane-loss.

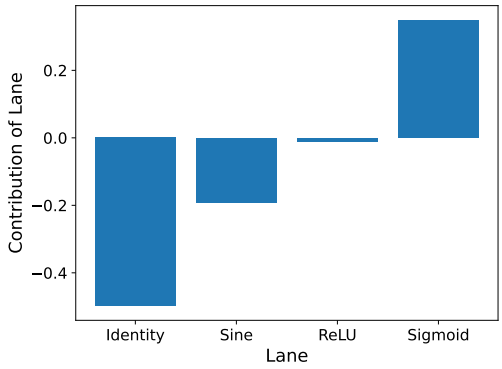

Figure 5.23: This shows the contribution to the output from different lanes. The x-axis shows the lane. The y-axis represents the contribution of the lanes. Notably, the contribution of the lanes does not at all match the contribution of neurons.

5.4 Takeaways

In this section, we condense the findings from the experiments chapter for each type of model.

Baseline (B). The baseline models perform somewhat as expected. The best activation function for `Sine`, `Addition`, `Multiplication`, and `Max` dataset is sine, identity, Log-Exp, and ReLU, respectively. This applies to both the training loss and the out-of-distribution loss.

Sequential (S). Sequential models perform rather poorly on the synthetic datasets. There are certain occasions with our synthetic data where a simplification with the identity function improves the performance. However, this is not the intended purpose. The sequential model can have a positive effect on the real-world datasets, as seen in Figure 5.14.

Ensemble (E)/Random (R). Ensemble as well as Random models have a similar effect on performance. They reach similar performance as a combination of activation functions, than their single activation function model equivalent. So instead of testing different activation function for each dataset, the combination of these functions will achieve acceptable results on all of the synthetic datasets. Unfortunately, the models do not adapt well to out-of-distribution ranges of some datasets.

Multi-Lane ((D), (T), (Q)). Our Multi-Lane models exhibit comparable performance to the Random (R) models. Specifically, the Multi-Lane model excels on our simpler, computation-based datasets, effectively balancing performance across different datasets. Instead of experiencing inconsistent performance - poor on some datasets and good on others - the Multi-Lane model mitigates these variabilities. Some models achieve satisfactory results across all of the test sets from diverse datasets. There are some instances of the model that generalize well on some datasets. However, in general, the generalization error does not improve substantially.
Some Multi-Lane model show consistent accuracy improvements over ReLU (B) and other activation ensemble models on the real-world datasets.
Furthermore, the Multi-Lane model has one advantage compared to normal ensemble models. Because of its structure, it is easy to eliminate lanes from the given network (Lane-Loss), to either compress the network to a smaller size or to further increase performance.

Lane-Loss. The implemented Lane-Loss regularizes the contribution of different lanes. An entropy loss helps the model select a specific lane with a distinct activation function. Although the model chooses the right lane in some cases, the elimination of the other lanes is not yet possible. The contribution of these lanes to the output is still too big.

Chapter 6

Conclusion and Future Work

Recent research suggests that relying solely on one single activation function in a network may not be optimal. This thesis combines various activation functions in feed-forward neural networks and explores their effects. We conducted tests on different models using synthetic datasets involving operations such as addition and multiplication, as well as real-world datasets such as MNIST and CIFAR10. The findings indicate that using a combination of activation functions can have a positive impact on performance, especially on the synthetic datasets. They can overcome the limitations of a single activation function network, which may perform well on some datasets but poorly on others. The ensemble models reduce the variance in overall performance and even achieve lower loss on certain datasets. The model maintains a high level of performance and demonstrates robustness across all of our synthetic datasets. However, while there are instances of models with improved generalization, altogether the improvement is not significant in our testing.

On real-world datasets, specific combinations outperform the classical ReLU network, whereas systems lacking ReLU achieve lower accuracy. We then introduce the novel Multi-Lane network architecture. It demonstrates similar performance compared to conventional activation mixture models on all synthetic datasets. Once again, the models struggle to adapt to out-of-distribution ranges.

Analyzing the contribution of the different lanes of the Multi-Lane network reveals, that the network successfully learns the underlying function primarily through the lane with the more suitable activation function.

The Multi-Lane consistently demonstrates good results on real-world datasets. The Quad-Lane model (Multi-Lane with four lanes) exhibits a lower test loss than the ReLU models on three out of four real-world datasets (with the same number of parameters).

Moreover, we introduce the concept of Lane-Loss to our Multi-Lane network. Our Lane-Loss approach aims to enable the model to minimize the influence of lanes, or rather to select the optimal one. We tested the modified model on synthetic, math-formula datasets. The model with Lane-Loss consistently achieves low loss values across all of the tested datasets. This was implemented by introducing an entropy loss with two different criteria. However, the Lane-Loss approach did

not lead to further performance improvement.
While the thesis provides insights into the impact of activation combinations, it is important to consider certain limitations. The choice of the Adam optimizer, learning rate, and loss function have an influence on performance. The different model architectures, especially with varying ratio of parameters to neurons, might have a considerable impact. The synthetic dataset ranges were not further investigated, therefore a model's behavior is not necessarily transferable to other or larger training ranges. Despite these limitations, this study contributes valuable information to the understanding of activation functions in neural networks. Our study establishes the foundation for future experiments. These experiments could extend our findings to other promising network architectures, such as convolutional neural networks, transformers, and recurrent neural networks. An improved criterion for the Lane-Loss might enhance the network's robustness and scope of application. Additionally, further investigation of the Lane-Loss could increase performance and lead to effective elimination of lanes. This could enable the model to be compressed, providing an opportunity to save memory space.

Bibliography

Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 2022.

Mark Fanty and Ronald Cole. Spoken letter recognition. *Advances in neural information processing systems*, 3, 1990.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/glorot11a.html.

Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. Noisy activation functions. In *International conference on machine learning*, pages 3059–3068. PMLR, 2016.

Mark Harmon and Diego Klabjan. Activation ensembles for deep neural networks. *arXiv preprint arXiv:1702.07790*, 2017.

Hidenori Ide and Takio Kurita. Improvement of learning for cnn with relu activation by sparse regularization. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2684–2691, 2017. doi: 10.1109/IJCNN.2017.7966185.

Xiaojie Jin, Chunyan Xu, Jiashi Feng, Yunchao Wei, Junjun Xiong, and Shuicheng Yan. Deep learning with s-shaped rectified linear activation units. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Dongwoo Lee, Sungbum Kang, and Kiyoung Choi. Compend: Computation pruning through early negative detection for relu in a deep neural network accelerator. In *Proceedings of the 2018 International Conference on Supercomputing*, pages 139–148, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

Arijit Nandi, Nanda Dulal Jana, and Swagatam Das. Improving the performance of neural networks with an ensemble of activation functions. In *2020 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2020.

Meera Narvekar and Priyanca Fargose. Daily weather forecasting using artificial neural network. 2015.

Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Taming the waves: sine as activation function in deep neural networks. 2016.

Sheng Qian, Hua Liu, Cheng Liu, Si Wu, and Hau San Wong. Adaptive activation functions in convolutional neural networks. *Neurocomputing*, 272:204–212, 2018.

Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

Tomasz Szandała. Review and comparison of commonly used activation functions for deep neural networks. *Bio-inspired neurocomputing*, pages 203–224, 2021.

Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):5314–5321, 2022.

Ludovic Trottier, Philippe Giguere, and Brahim Chaib-Draa. Parametric exponential linear unit for deep convolutional neural networks. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 207–214. IEEE, 2017.

Hongmei Yan, Yingtao Jiang, Jun Zheng, Chenglin Peng, and Qinghui Li. A multilayer perceptron-based medical decision support system for heart disease diagnosis. *Expert Systems with Applications*, 30(2):272–281, 2006.

Appendix A

Models and Setup

A.1 Calculate the Number of Parameters

A.1.1 Fully Connected Network

The number of parameters inside a fully connected network is calculated as follows. The parameters are made up of weights and biases. Additionally, for batch normalization (real-world datasets) there are two parameters per neuron. May l be the number of layers and n is the number of neurons per layer. i is the number of input features and m the number of outputs.
Weights between input to first hidden layer: $i \cdot n$
Weights between hidden layers: $(l-1) \cdot n^2$
Weights between last hidden layer and output: $m \cdot n$
Biases: $l \cdot n + m$
Additionally with batch normalization (two parameters per neuron in the hidden layers): $2 \cdot l \cdot n$
This **results** in:
$(l-1) \cdot n^2 + (i+m+l) \cdot n + m$,
for synthetic datasets and
$(l-1) \cdot n^2 + (i+m+3 \cdot l) \cdot n + m$,
for real-world datasets.

A.1.2 Multi-Lane Network

The number of parameters inside a Multi-Lane network is calculated as follows. The parameters are made up of weights and biases. Additionally, for batch normalization (real-world datasets) there are two parameters per neuron. May l be the number of layers and n is the number of neurons per layer. i is the number of input features and m the number of outputs. r is the number of lanes used in the network.
Weights between input to first hidden layer: $i \cdot n$
Weights between hidden layers: $r \cdot (l-1) \cdot (\frac{n}{r})^2$

Weights between last hidden layer and output: $m \cdot n$
Biases: $l \cdot n + m$
Additionally with batch normalization (two parameters per neuron in the hidden layers): $2 \cdot l \cdot n$
This **results** in:
$\frac{l-1}{r} \cdot n^2 + (i + m + l) \cdot n + m$,
parameters for the synthetic datasets and
$\frac{l-1}{r} \cdot n^2 + (i + m + 3 \cdot l) \cdot n + m$,
parameters for the real-world datasets.

A.2 Learning Rates

For the synthetic datasets, we most often used two different learning rates. This is because there were big differences between activation functions and sizes of the network. Larger networks for example generally prefer smaller learning rate. The real-world datasets use a learning rate as specified in Table. We made some test runs to see how many epochs the models run for and decided then appropriately.

Dataset	Learning Rates
Addition	0.0005
Multiplication	0.005, 0.0005
Sine	0.01, 0.001
Max	0.005, 0.0005
MNIST	0.0001
FashionMNIST	0.0001
ISOLET	0.0001
CIFAR10	0.00001
$sin(x_1) + cos(x_2)$	0.001
$sin(x_1) \cdot sin(x_2)$	0.001
$max(sin(x_1), sin(x_2))$	0.01, 0.0001
Poly-v1	0.01, 0.0001
Poly-v2	0.0005, 0.00001

Table A.1: Learning rates used for training the different datasets. The learning rate is fixed during the training. If the learning rates are defined, the training were completed with both. The better score was then used for comparison.

Appendix B

Further Results

B.1 2D Loss Heatmaps

In the following part we added different 2D loss heatmaps. The heatmaps are made from different datasets and different model sizes. The information is appended in the figure captions. For more information we refer to Chapter 5.

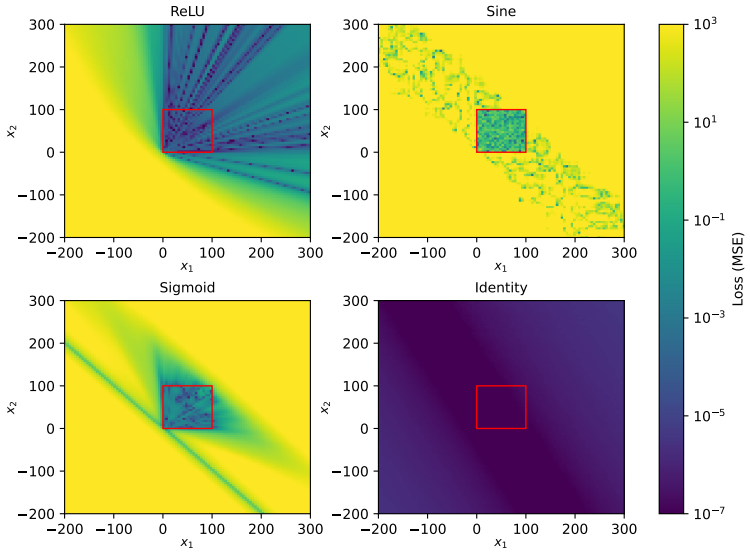

Figure B.1: MSE loss landscape for four different activation functions on the `Addition` dataset. The network consists of two layers with 128 neurons each. The two axis of each plot represent the two inputs to the network. The loss of the network is color-coded according to the color bar. The red rectangle indicates the area the model was trained on. The Identity activation shows superior performance. Both sigmoid and sine show a diagonal ribbon where the loss is minimized. In comparison to the low-complexity model in Figure 5.4, the sigmoid and sine versions perform better on the test range. The ReLU model also fans up, covering now the upper right quarter of the plane instead of only the right side at the expense of increased loss in the training region.

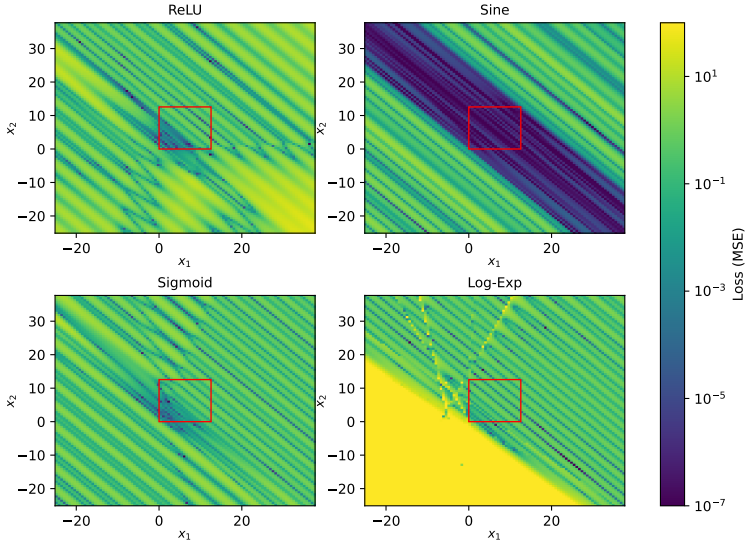

Figure B.2: MSE loss landscape for four different activation functions on the `Sine` dataset. The two axis of each plot represent the two inputs to the network. The loss of the network is color-coded according to the color bar. The red rectangle indicates the area the model was trained on. In comparison to the prior plot, the used network contains only two layers with 8 neurons each. The sine activation shows superior performance. Interestingly the `Sine` model shows a large stripe with low loss values. All other models show poor performance outside of the training area.

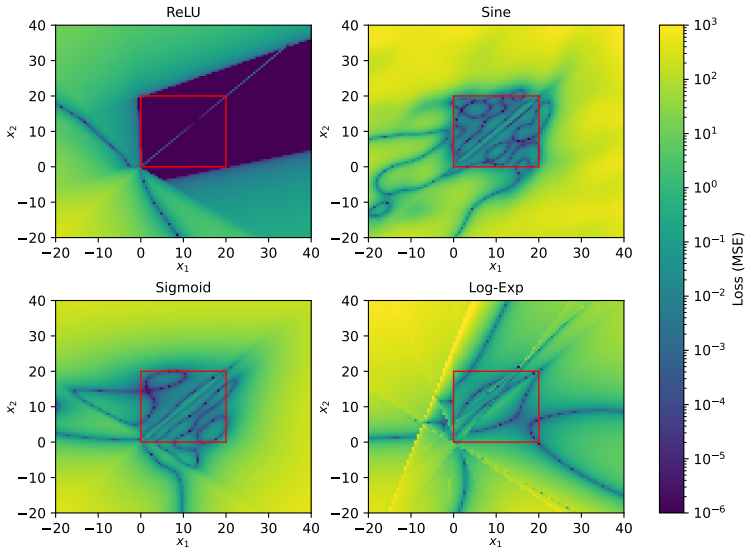

Figure B.3: MSE loss landscape for four different activation functions on the `Max` dataset. The network consists of two layers with 8 neurons each. The two axis of each plot represent the two inputs to the network. The loss of the network is color-coded according to the color bar. The red rectangle indicates the area the model was trained on.

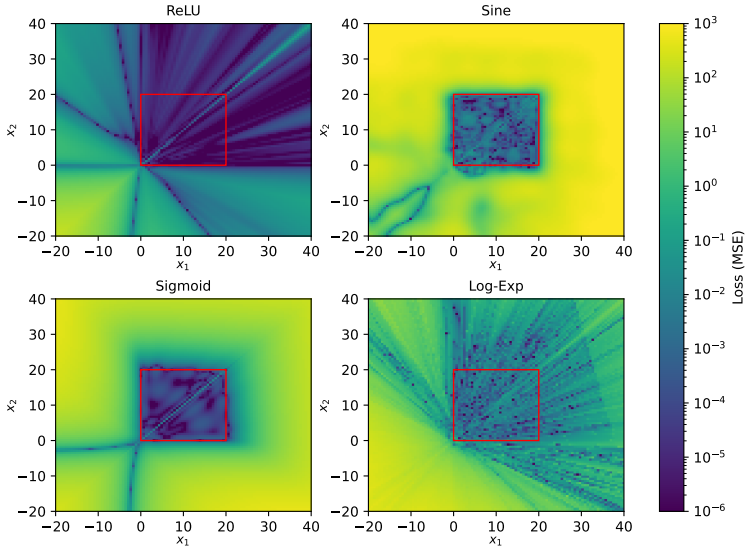

Figure B.4: MSE loss landscape for four different activation functions on the `Max` dataset. The network consists of two layers with 128 neurons each. The two axis of each plot represent the two inputs to the network. The loss of the network is color-coded according to the color bar. The red rectangle indicates the area the model was trained on.

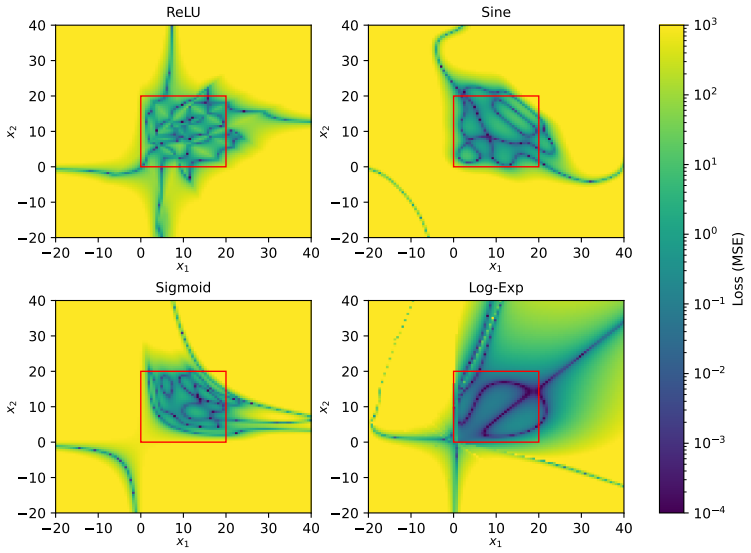

Figure B.5: MSE loss landscape for four different activation functions on the `Multiplication` dataset. The network consists of two layers with 8 neurons each. The two axis of each plot represent the two inputs to the network. The loss of the network is color-coded according to the color bar. The red rectangle indicates the area the model was trained on.

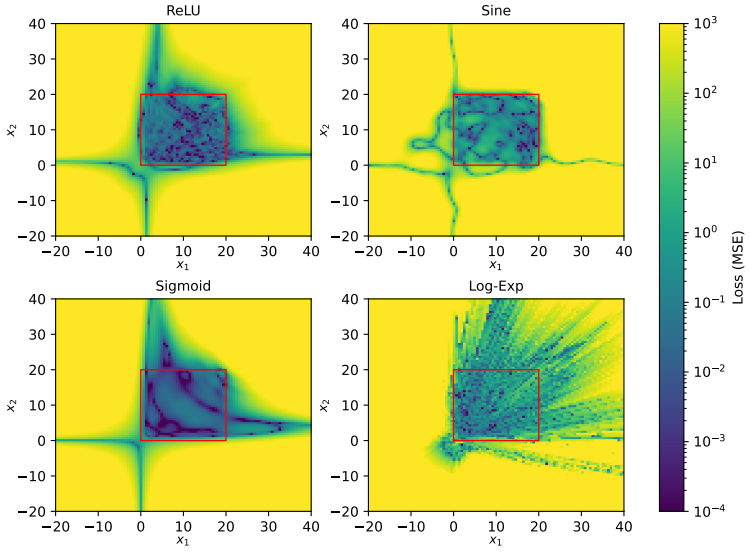

Figure B.6: MSE loss landscape for four different activation functions on the `Multiplication` dataset. The network consists of two layers with 128 neurons each. The two axis of each plot represent the two inputs to the network. The loss of the network is color-coded according to the color bar. The red rectangle indicates the area the model was trained on.

B.2 Model Performances

B.2.1 Synthetic Datasets

This sections shows the results of models trained on the synthetic datasets. For more information about the datasets see Section 5.2. For more information about the model architectures and their abbreviation see Section 4. The figures show the performance of the model with regards to the number of parameters. The x-axis represents the test loss of the model, while the y-axis represents the number of parameters. For more information about the performance of the models see Chapter 5.

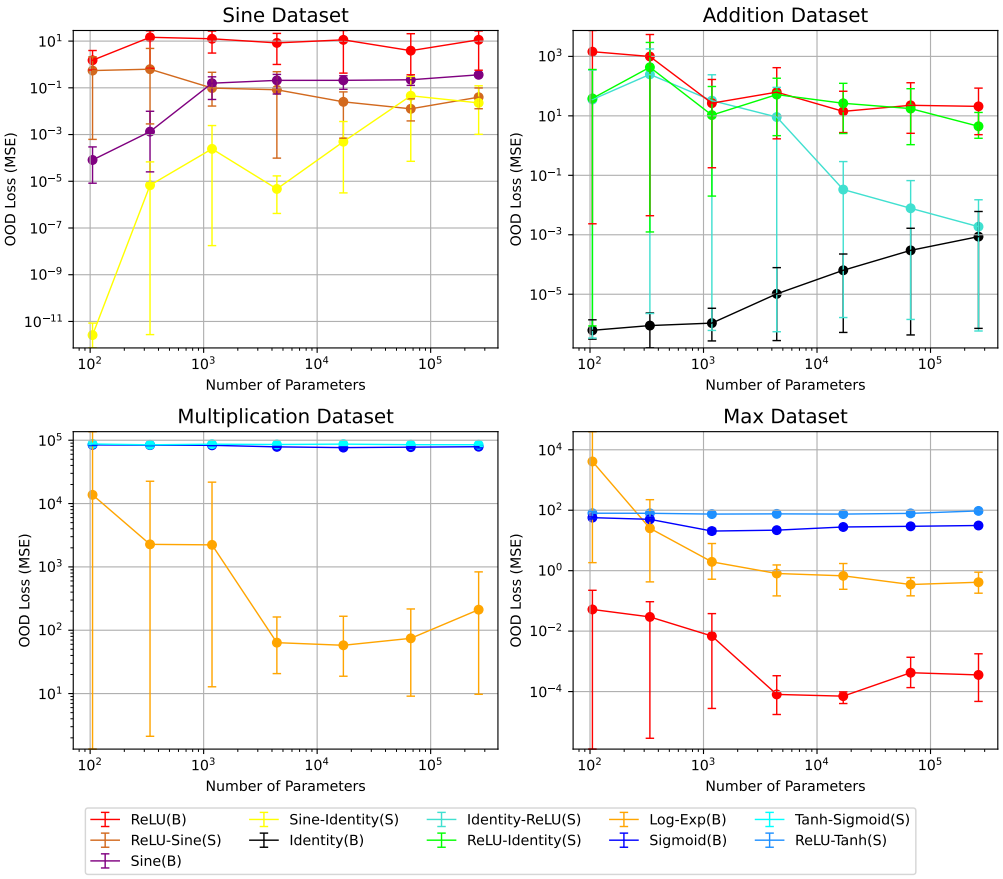

Figure B.7: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (S) indicates that it is a sequential model with the first layer having a different activation function to the second layer. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the out-of-distribution set as the model complexity increases. Notably, on the `Sine` dataset, the Identity-Sine (S) and Sine-Identity (S) model adapt really well to the new data. The Identity-ReLU (S) model in some occasions scores equally as low as than Identity (B) network on the `Addition` dataset. Furthermore, the Log-Exp (B) is still the sole model generalizing on the `Multiplication` dataset.

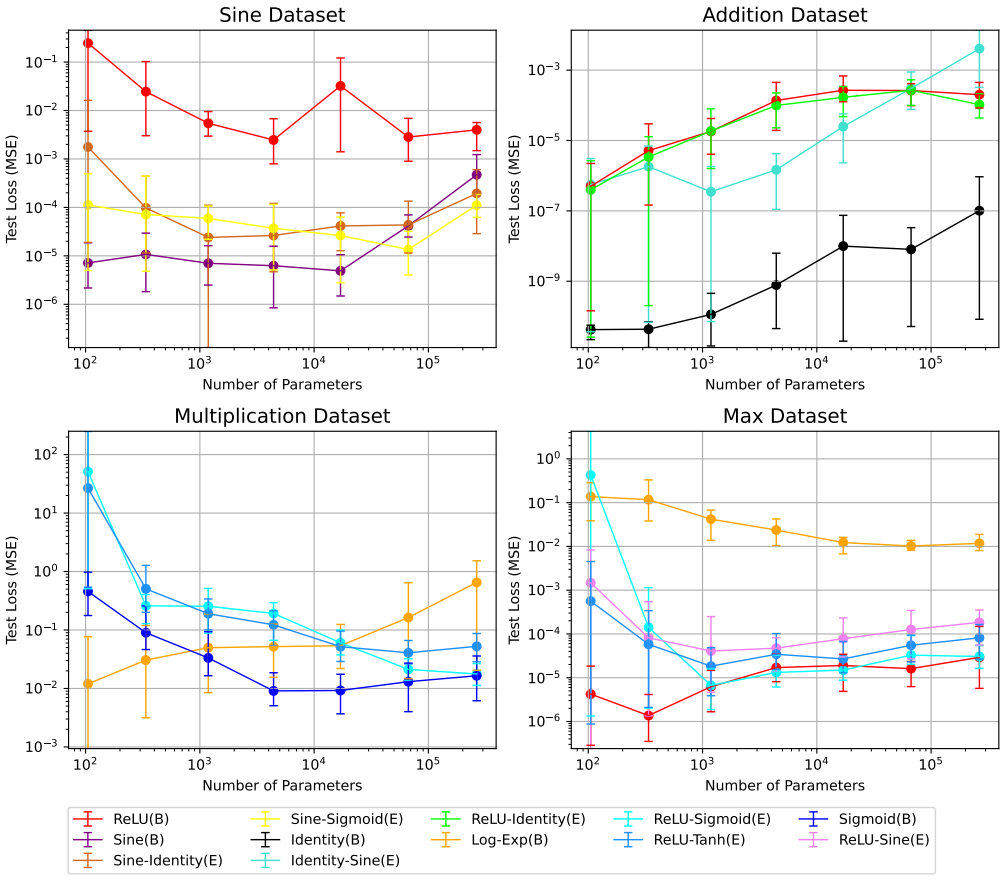

Figure B.8: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (E) indicates that it is a ensembling model with the two layers being divided into two halves, each using a different activation function. (S) is a sequential model with the first layer having a different activation function to the second layer. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases. Interestingly, neither the Sine-Identity (E) nor the Sine-Sigmoid (E) models come close to the Identity-Sine (S) model performance wise. Nevertheless various ensembling models perform similarly well to the ReLU (B) model on the `Max` dataset.

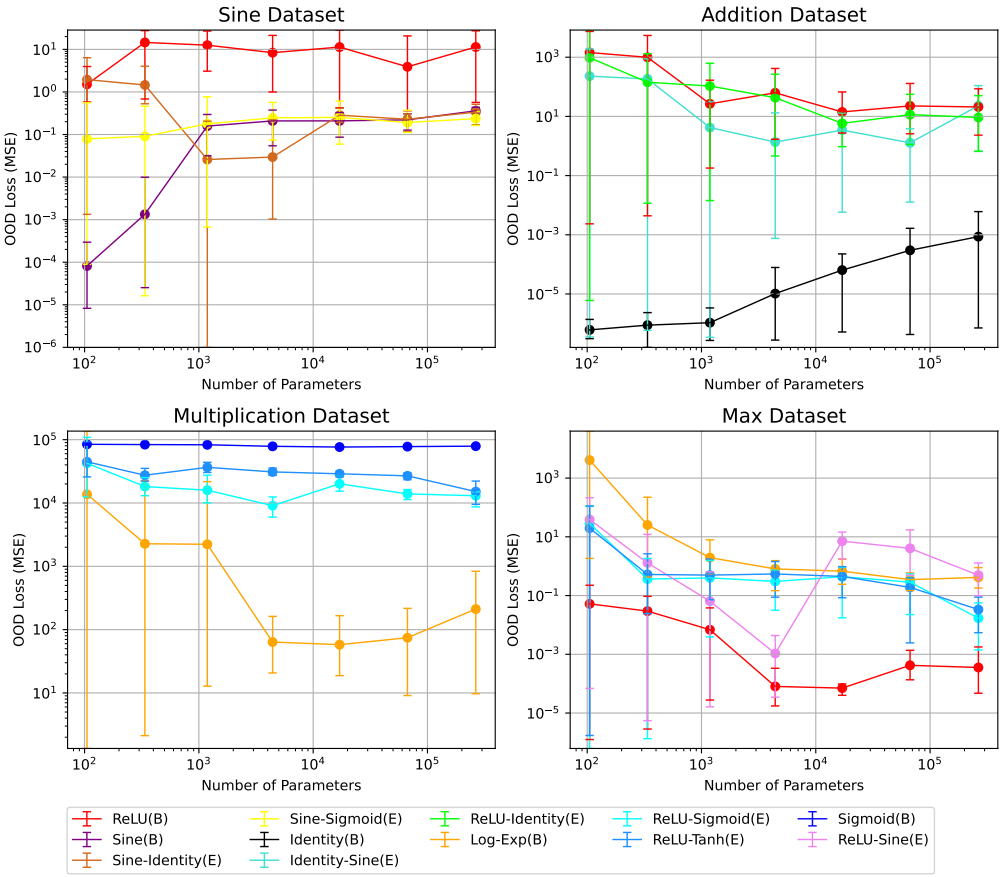

Figure B.9: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (E) indicates that it is a ensembling model with the two layers being divided into two halves, each using a different activation function. (S) is a sequential model with the first layer having a different activation function to the second layer. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the out-of-distribution set as the model complexity increases. Notably, on the `Sine` dataset, none of the ensembling models partially containing the sine activation function are able to compete with the Identity-Sine (S) model. There are however individual ensembling models outperforming the Sine (B) models. Similar things happen on the `Addition` dataset. Only certain instances of the Identity-Sine(E) model, are able to exhibit similar performance to Identity (B).

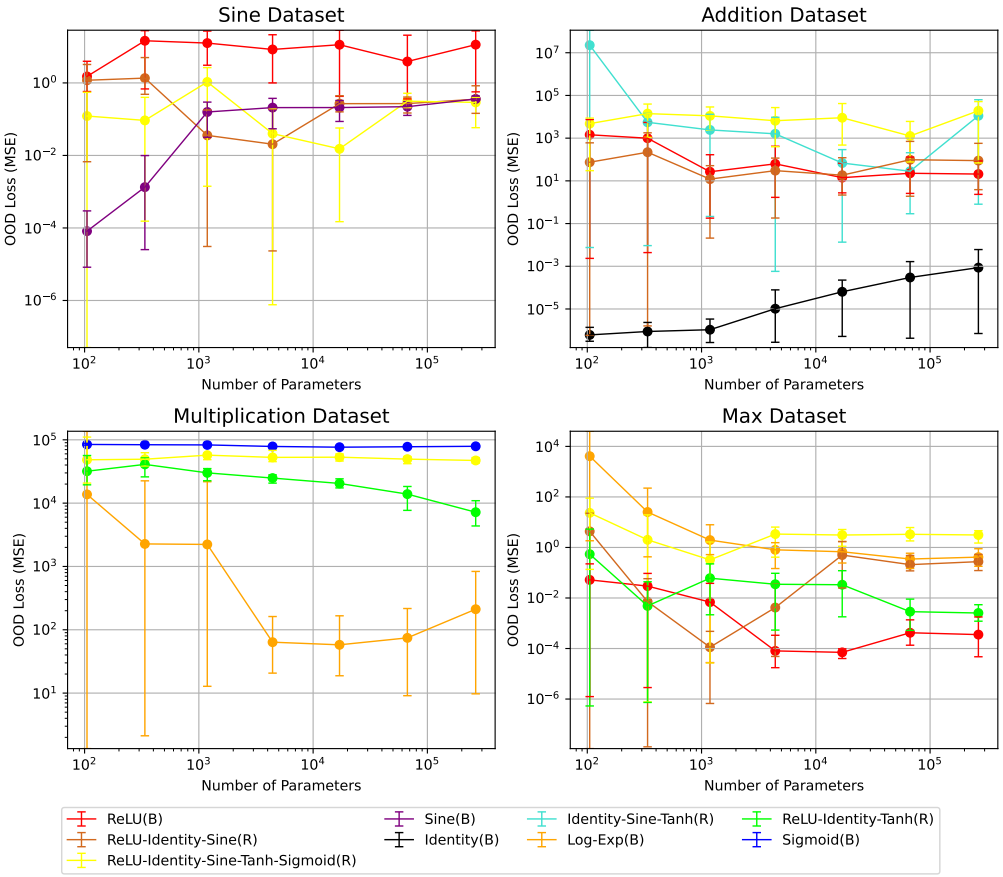

Figure B.10: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (R) indicates that it is a random model, meaning that for every neuron in the network a random activation function is chosen from the given set. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the out-of-distribution set as the model complexity increases. Notably, ReLU-Identity-Tanh (R) and ReLU-Identity-Sine (R) adapt equally strongly to the OOD range than ReLU (B), particularly lower-parameter models. On other `Sine` dataset it is the ReLU-Identity-Sine (R) and ReLU-Identity-Sine-Tanh-Sigmoid (R) models that are able to outperform the Sine (B) models in some cases.

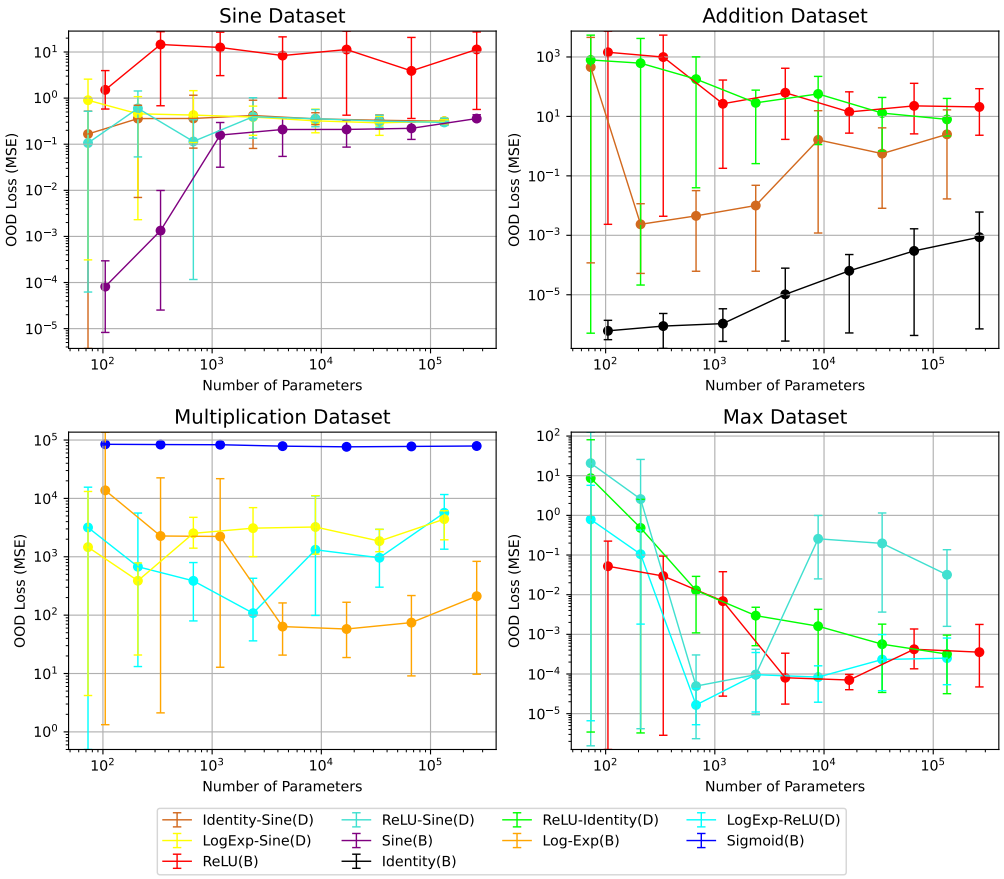

Figure B.11: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (D) indicates that it is a Doublelane model, consisting of two different lanes of neurons (see Figure 4.4). (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the out-of-distribution set as the model complexity increases. Notably, for lower-parameter networks, LogExp-Sine (D) as well as LogExp-ReLU (D) on average adapt better to OOD data than LogExp (B) itself on the `Multiplication` dataset. LogExp-ReLU (D) not only achieves low loss values there, but also on the `Max` dataset.

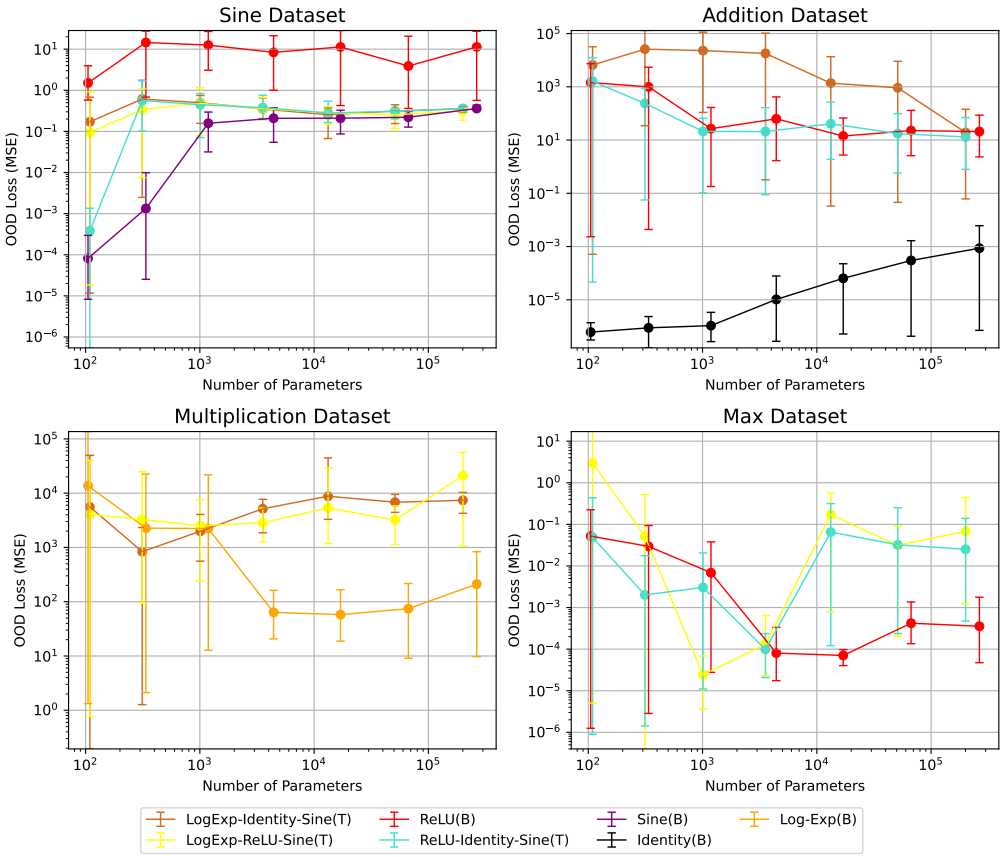

Figure B.12: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (T) indicates that it is a Triplelane model, consisting of three different lanes of neurons. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the out-of-distribution set as the model complexity increases. Unfortunately, the LogExp-Identity-Sine (T) model is not able to carry the low loss values from test to OOD data. However, we see that ReLU-Identity-Sine (T) shows good adaptation abilities for low-parameter networks.

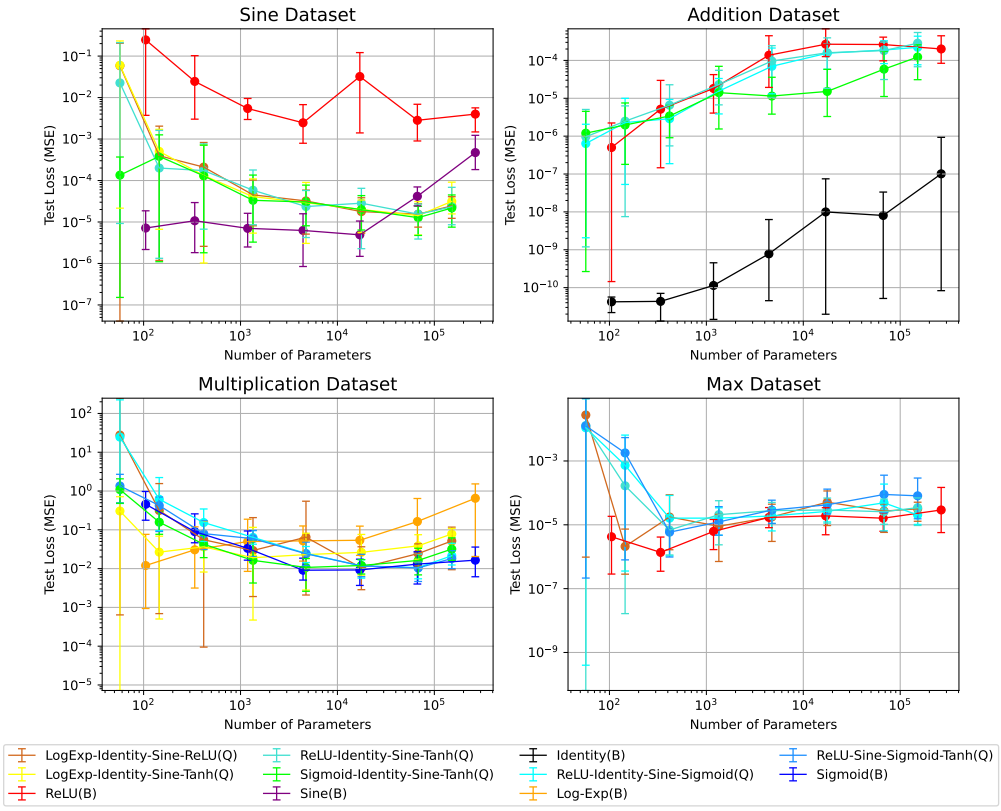

Figure B.13: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (Q) indicates that it is a Quadlane model, consisting of four different lanes of neurons. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases. Notably, LogExp-Identity-Sine-ReLU (Q) performs well on `Multiplication` and `Max` dataset. On the `Sine` dataset the model is not able to consistently perform close to Sine (B).

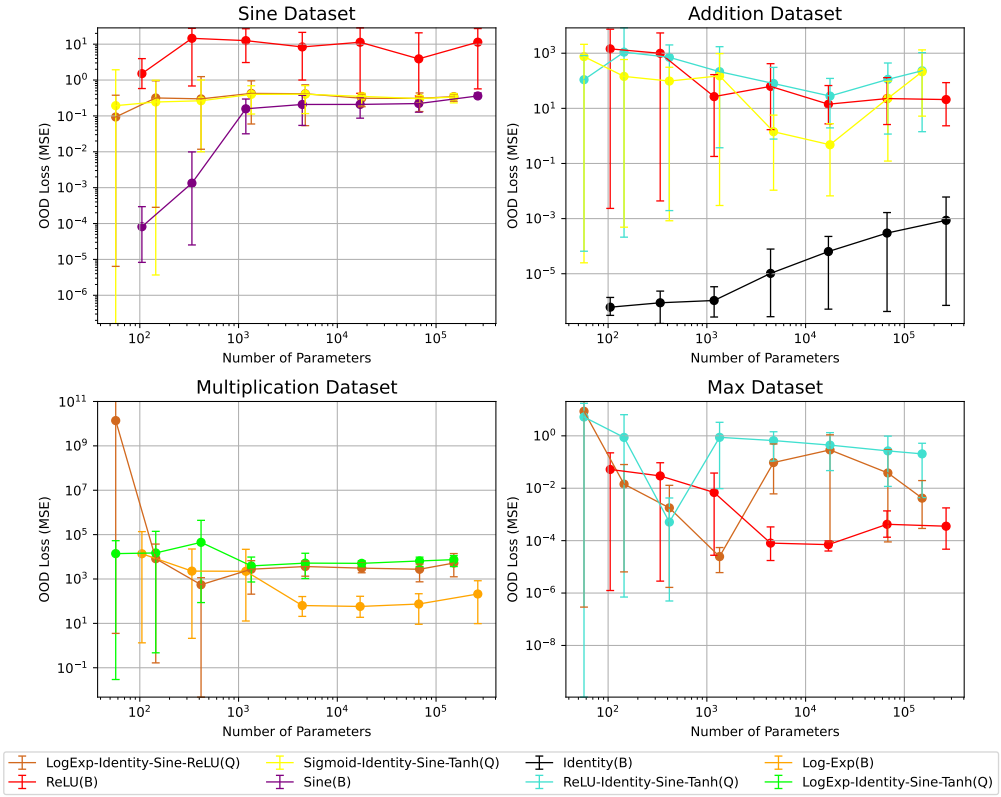

Figure B.14: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (Q) indicates that it is a Quadlane model, consisting of four different lanes of neurons. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the out-of-distribution set as the model complexity increases. LogExp-Identity-Sine-ReLU (Q) achieves similarly low loss values for both `Multiplication` and `Max` dataset. Some instances of Sigmoid-Identity-Sine-Tanh (Q) adapt very well to OOD ranges on both `Sine` and `Addition` dataset.

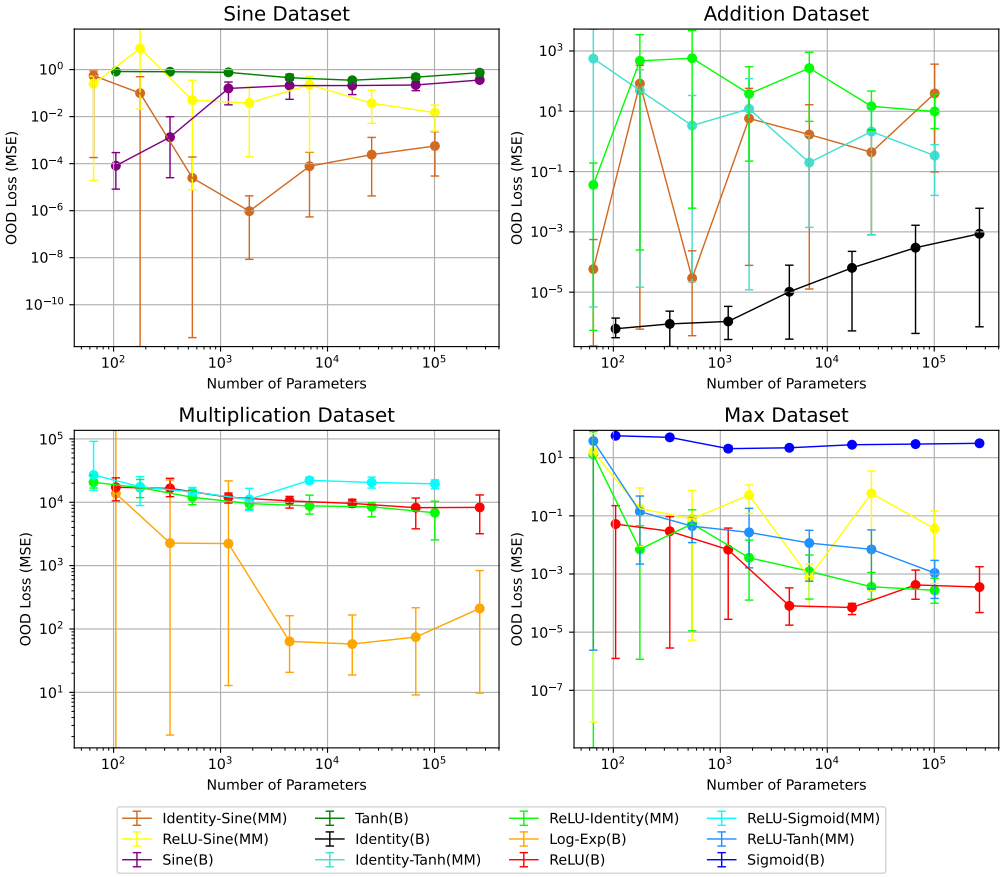

Figure B.15: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets (`Sine`, `Addition`, `Multiplication`, `Max`) as described in Section 5.1.1. (MM) indicates that it is a modified Multi-Lane model. It has a structure as described in Figure 4.5. The graphs show the trend of the MSE loss on the out-of-distribution set as the model complexity increases. Notably, Identity-Sine (MM) also performs well on OOD data. This applies for both the `Sine` and the `Addition` dataset.

B.2.2 Real-World Datasets

This section shows further performances of models trained on the real-world datasets. For more information we refer to Chapter 5.

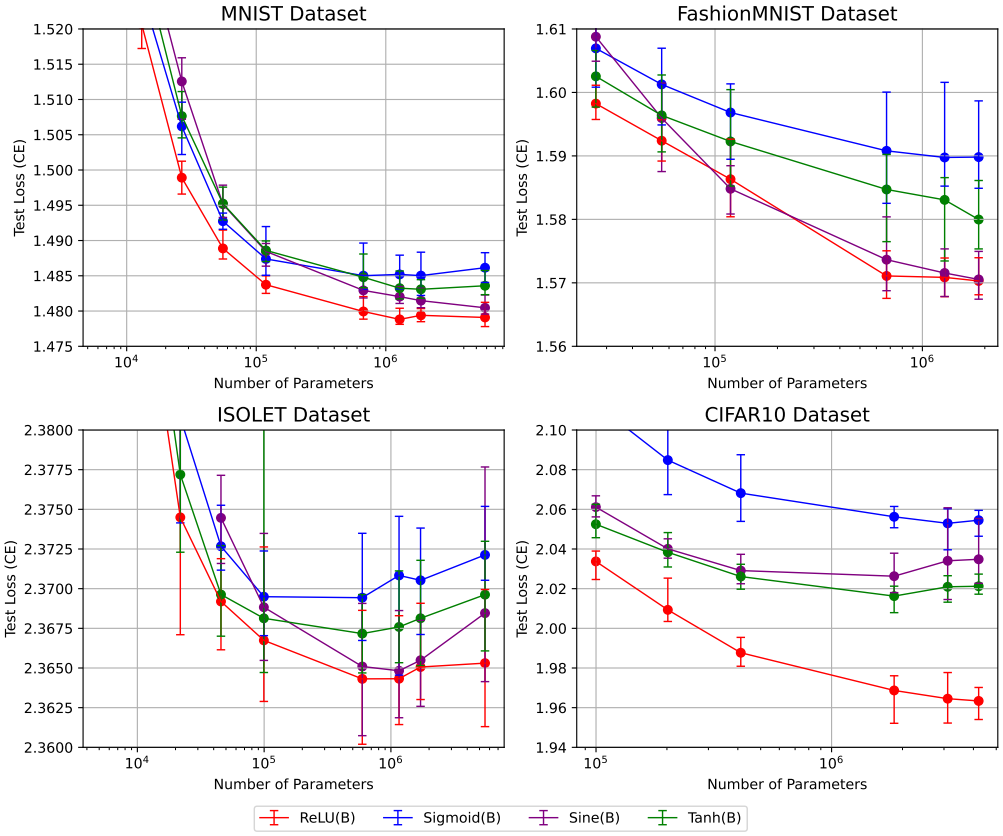

Figure B.16: Comparison of the model performance on the test set relative to the number of parameters for four different datasets (MNIST, FashionMNIST, ISOLET, CIFAR10), as defined in Section 5.1.1. The graphs show the trend of the cross entropy loss on the test set as the model complexity increases. Notably, ReLU consistently outperforms the other activation functions on all the datasets. Tanh generally performs better than Sigmoid. Somewhat surprisingly the sine model is able to compete with ReLU on the ISOLET dataset. Log-Exp (B) did not perform good enough to include it.

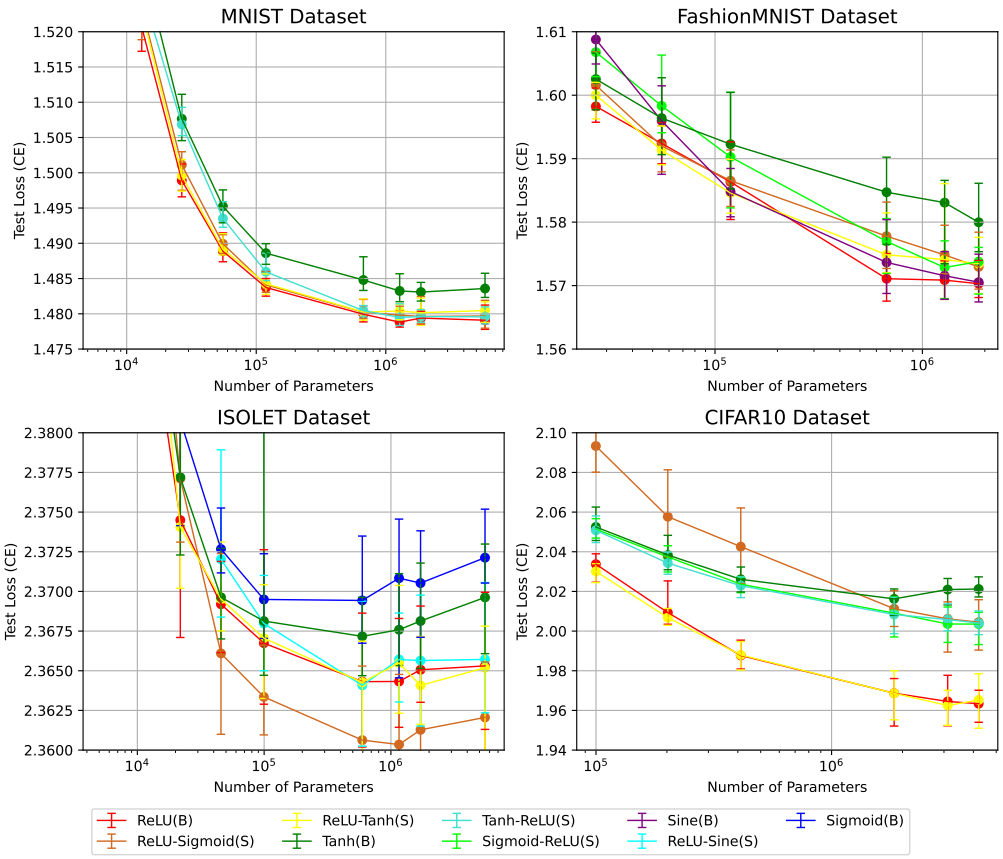

Figure B.17: Comparison of the model performance on the test set relative to the number of parameters for four different datasets (MNIST, FashionMNIST, ISOLET, CIFAR10), as defined in Section 5.1.1. (S) indicates that it is a sequential model with the first layer having a different activation function to the second layer. (B) marks a pure, single activation function model. The graphs show the trend of the cross entropy loss on the test set as the model complexity increases. Notably, most well performing models rely at least partially on ReLU activation function. The ReLU-Sigmoid (S) network does achieve lower losses than ReLU (B) on the ISOLET dataset.

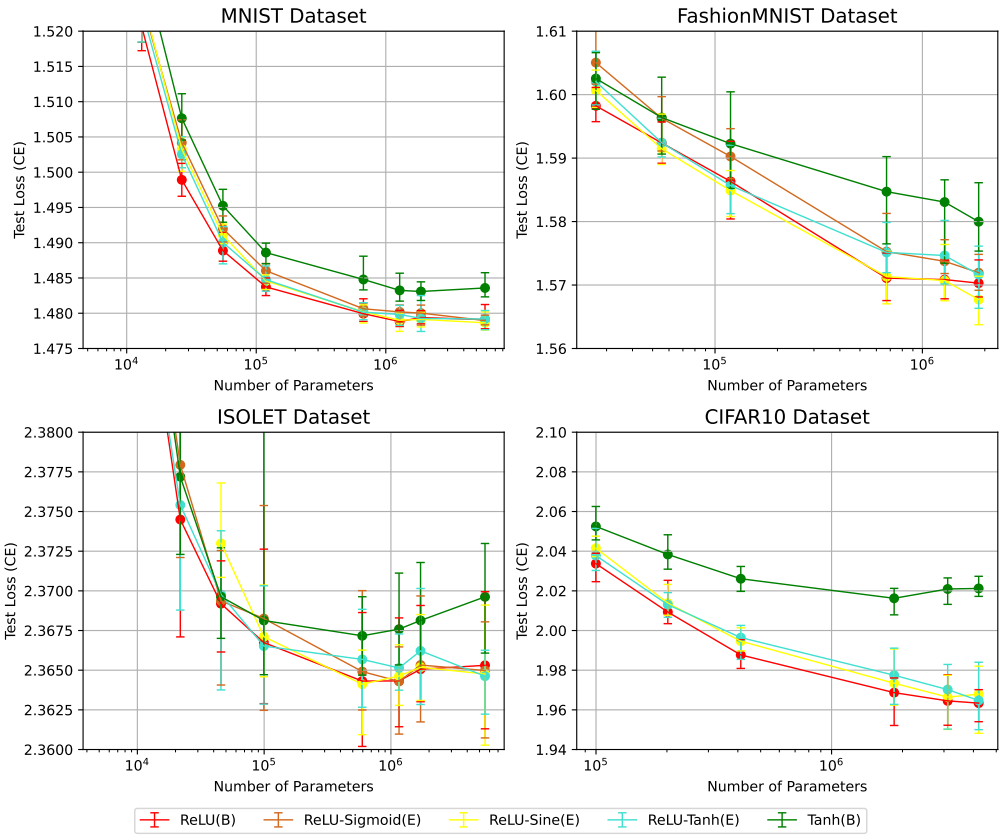

Figure B.18: Comparison of the model performance on the test set relative to the number of parameters for four different datasets (MNIST, FashionMNIST, ISOLET, CIFAR10), as defined in Section 5.1.1. (E) indicates that it is a ensembling model with the two layers being divided into two halves, each using a different activation function. (B) marks a pure, single activation function model. The graphs show the trend of the cross entropy loss on the test set as the model complexity increases. Notably, the ReLU-Sine (E) model consistently challenges the ReLU (B) model on all datasets.

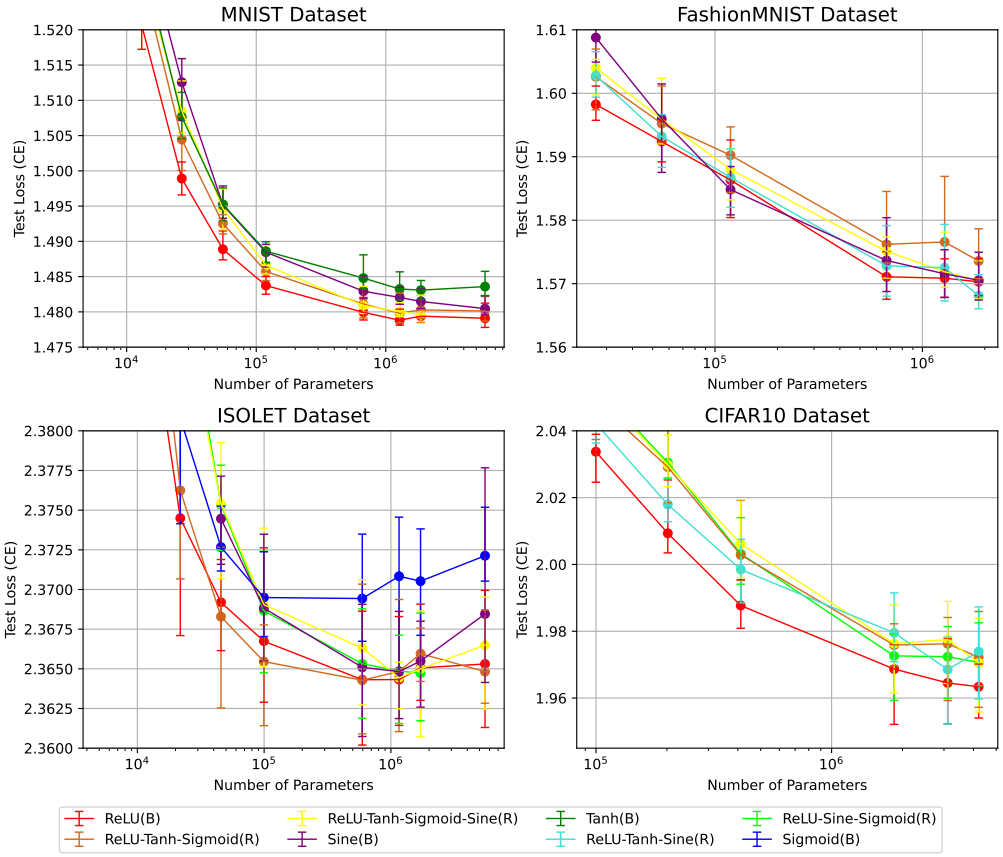

Figure B.19: Comparison of the model performance on the test set relative to the number of parameters for four different datasets (MNIST, FashionMNIST, ISOLET, CIFAR10), as defined in Section 5.1.1. (R) indicates that it is a random model, meaning that for every neuron in the network a random activation functions is chosen from the given set. (B) marks a pure, single activation function model. The graphs show the trend of the cross entropy loss on the test set as the model complexity increases. Notably, the ReLU-Tanh-Sigmoid (R) model performs superior to ReLU (B) for some sizes of networks.

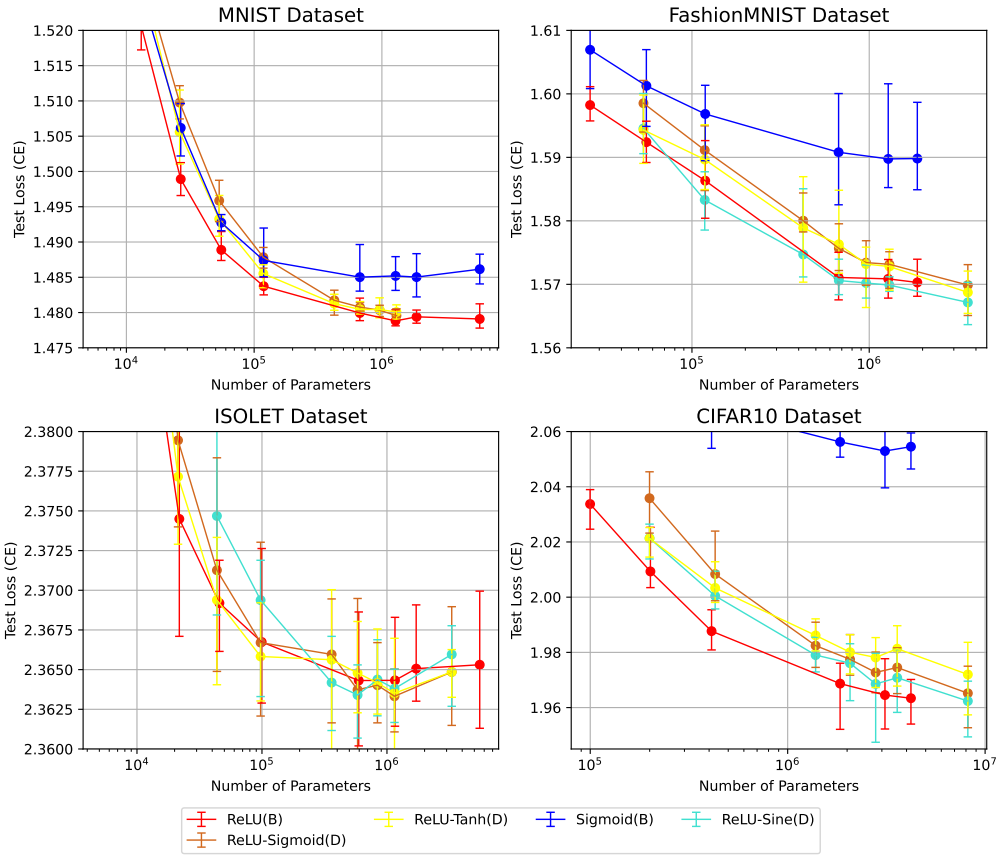

Figure B.20: Comparison of the model performance on the test set relative to the number of parameters for four different datasets (MNIST, FashionMNIST, ISOLET, CIFAR10), as defined in Section 5.1.1. (D) indicates that it is a Double-Lane model containing two different lanes of activation functions as in Figure 4.4. (B) marks a pure, single activation function model. Notably, the ReLU-Sine (D) model achieves superior performance compared to ReLU (B) for some cases on the ISOLET as well as the FashionMNIST dataset.

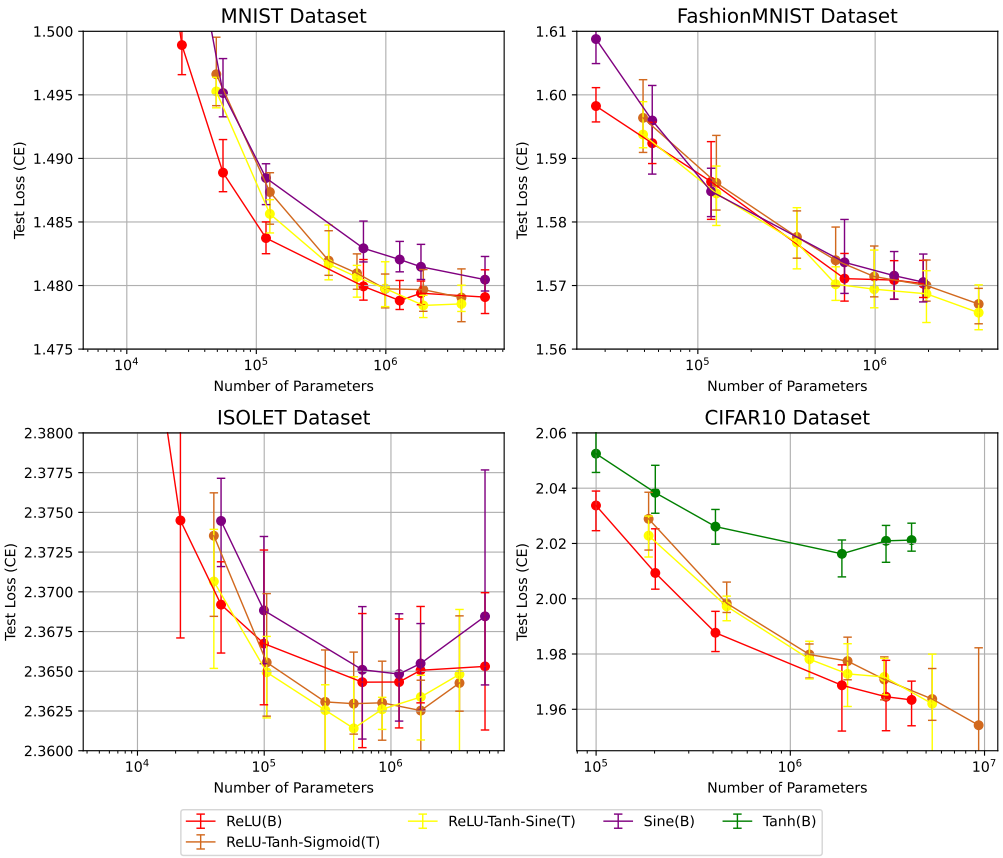

Figure B.21: Comparison of the model performance on the test set relative to the number of parameters for four different datasets (MNIST, FashionMNIST, ISOLET, CIFAR10), as defined in Section 5.1.1. (T) indicates that it is a Triplelane model containing three different lanes of activation functions. (B) marks a pure, single activation function model. Surprisingly, ReLU-Tanh-Sine (T) ourperform ReLU (B) on three of the four datasets. ReLU-Tanh-Sigmoid (T) also shows superior performance to ReLU (B)

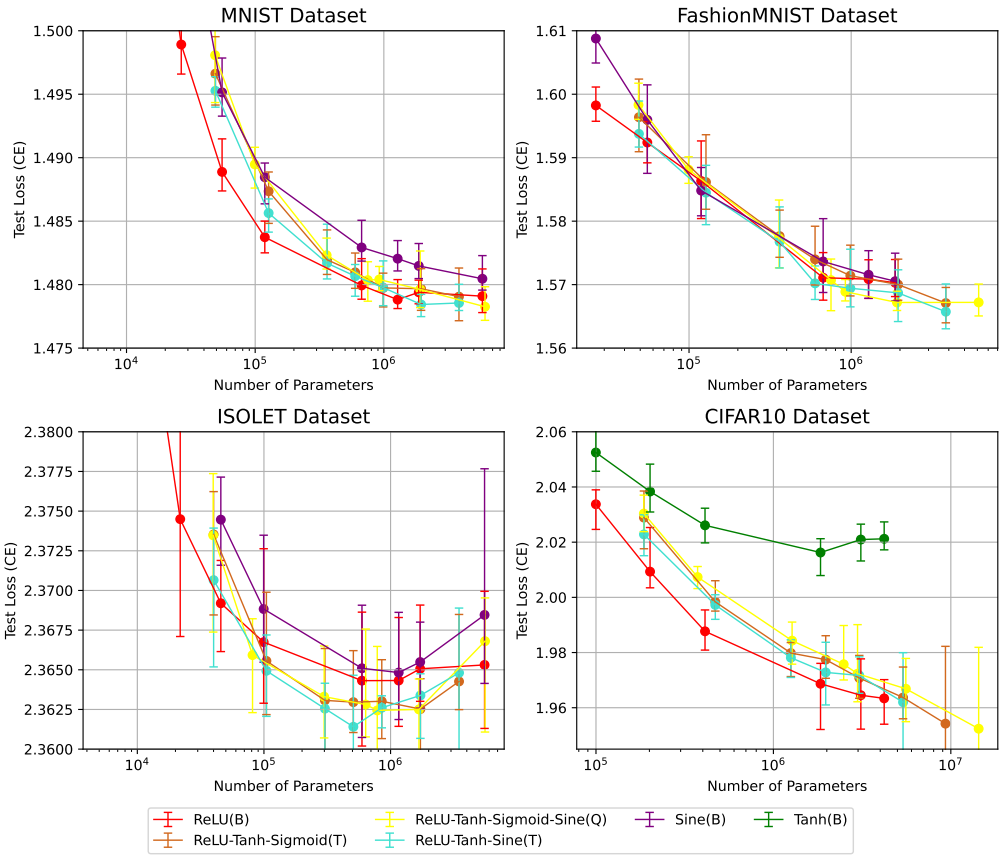

Figure B.22: Comparison of the model performance on the test set relative to the number of parameters for four different datasets (MNIST, FashionMNIST, ISOLET, CIFAR10), as defined in Section 5.1.1. (Q) indicates that it is a Quadlane model containing four different lanes of activation functions. (B) marks a pure, single activation function model. Notably, the best performance on three of the four datasets was achieved by the quadlane model.

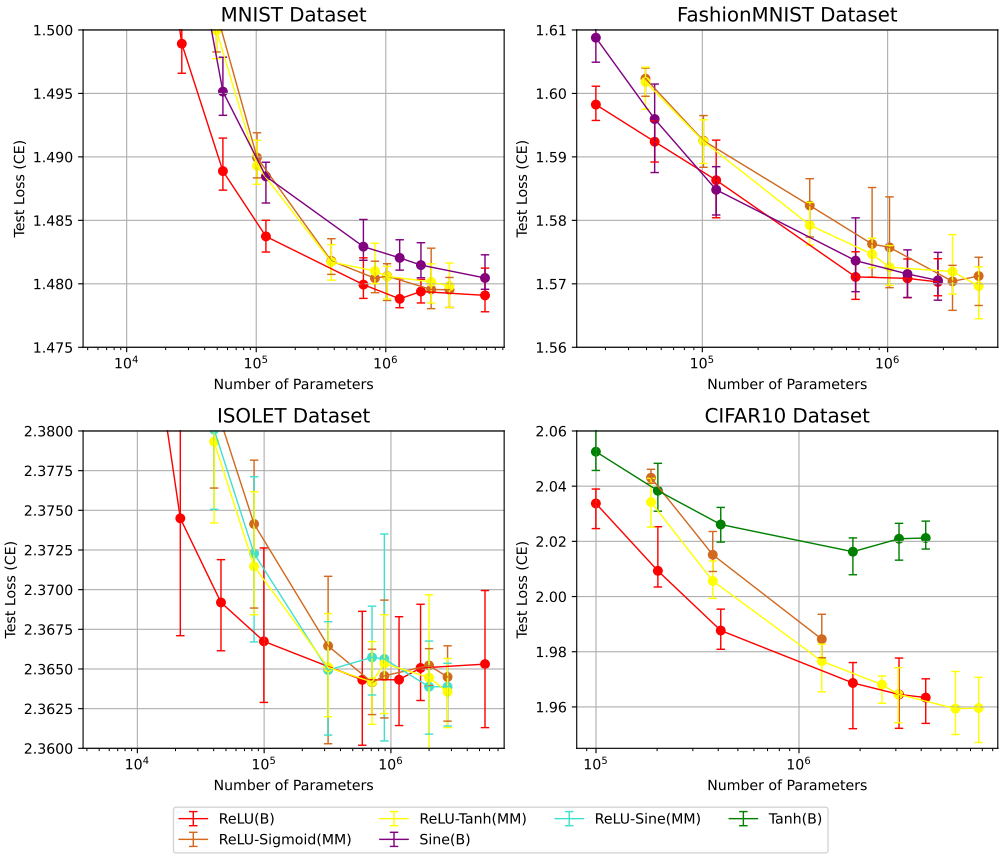

Figure B.23: Comparison of the model performance on the test set relative to the number of parameters for four different datasets (MNIST, FashionMNIST, ISOLET, CIFAR10), as defined in Section 5.1.1. (MM) indicates that it is a modified Multi-Lane model.It has a structure as described in Figure 4.5. (B) marks a pure, single activation function model. Notably, larger ReLU-Tanh (MM) models perform at least as good as ReLU (B) models on all datasets.

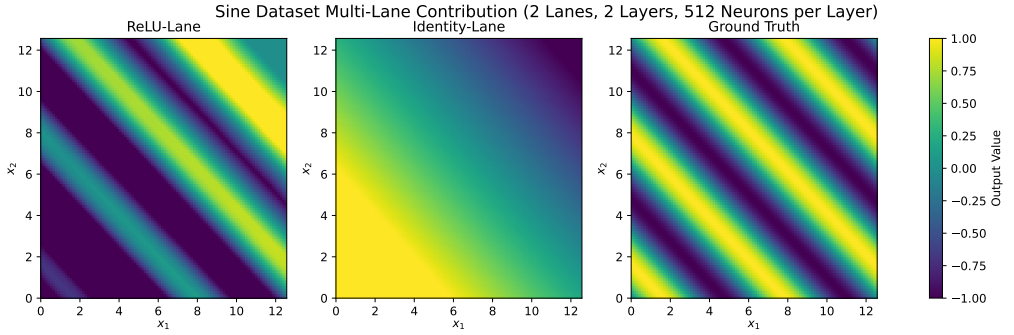

Figure B.24: Comparison of the two lanes within the Doublelane model that is trained on the `Sine` dataset. The x-axis represents the first input while the y-axis represents the second input. The values are color-coded according to the color bar and show the contribution of the lanes to the final output. The first plot shows the contribution of the ReLU lane of the network. The subsequent plot displays the contribution of the Identity lane. The third plot shows the output labels for the input 1 and 2. Notably, the ReLU lane retains most of the structure of the output, whereas the Identity lane contributes with a linear transition.

B.3 Lane Contribution

This section includes figures of the lane contribution. It shows, how much each lane of a Double-Lane models contributes to the output. The right plot shows the labels for the input data x_1 and x_2.

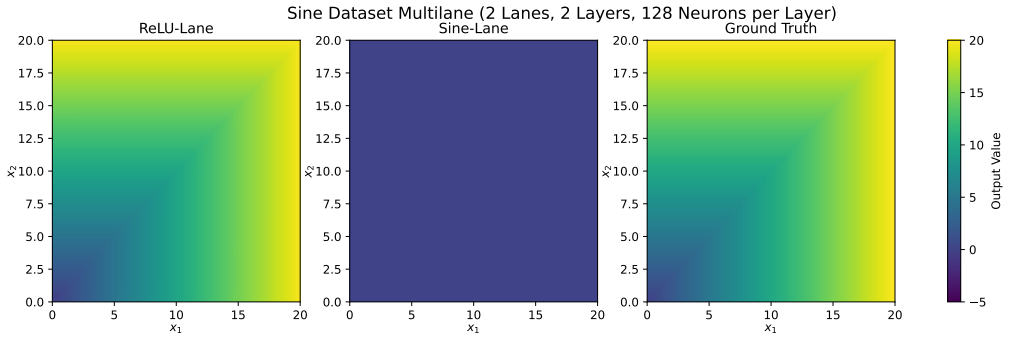

Figure B.25: Comparison of the two lanes within the Doublelane model that is trained on the `Sine` dataset. The x-axis represents the first input while the y-axis represents the second input. The values are color-coded according to the color bar and show the contribution of the lanes to the final output. The first plot shows the contribution of the ReLU lane of the network. The subsequent plot displays the contribution of the sine lane. The third plot shows the output labels for the input 1 and 2.

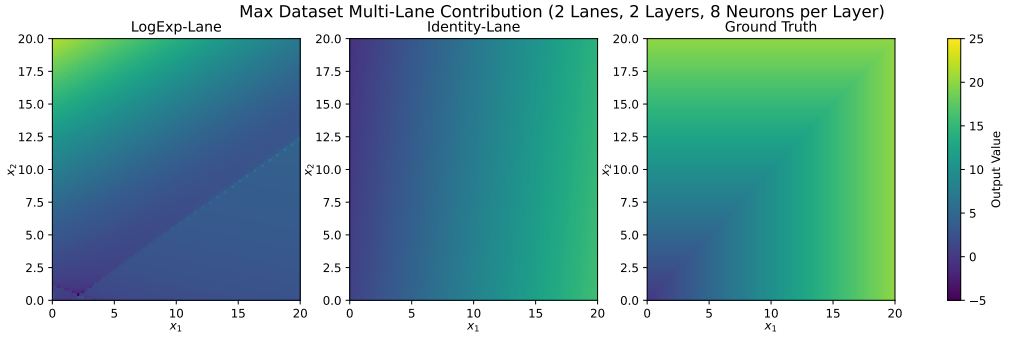

Figure B.26: Comparison of the two lanes within the Doublelane model that is trained on the `Max` dataset. The x-axis represents the first input while the y-axis represents the second input. The values are color-coded according to the color bar and show the contribution of the lanes to the final output. The first plot shows the contribution of the LogExp lane of the network. The subsequent plot displays the contribution of the Identity lane. The third plot shows the output labels for the input 1 and 2.

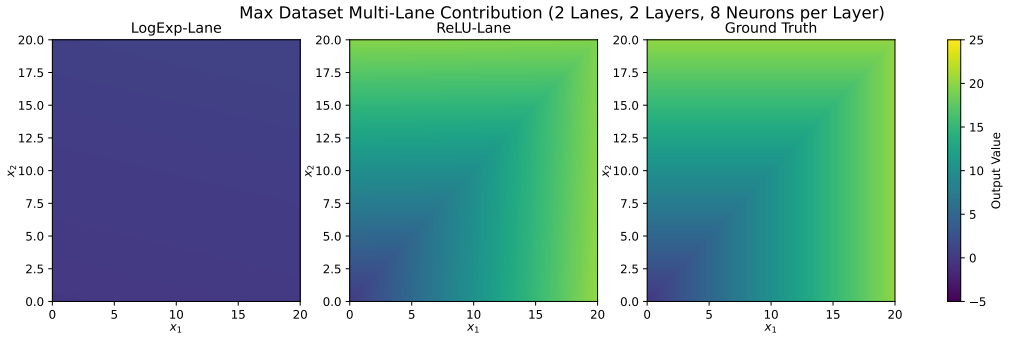

Figure B.27: Comparison of the two lanes within the Doublelane model that is trained on the `Max` dataset. The x-axis represents the first input while the y-axis represents the second input. The values are color-coded according to the color bar and show the contribution of the lanes to the final output. The first plot shows the contribution of the LogExp lane of the network. The subsequent plot displays the contribution of the ReLU lane. The third plot shows the output labels for the input 1 and 2.

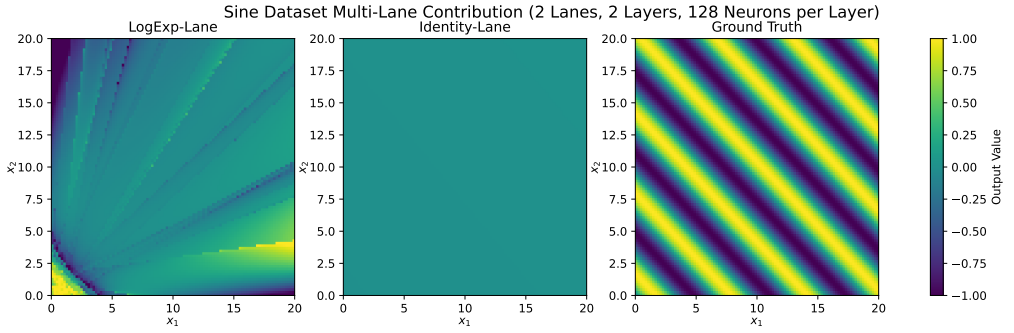

Figure B.28: Comparison of the two lanes within the Doublelane model that is trained on the `Sine` dataset. The x-axis represents the first input while the y-axis represents the second input. The values are color-coded according to the color bar and show the contribution of the lanes to the final output. The first plot shows the contribution of the LogExp lane of the network. The subsequent plot displays the contribution of the Identity lane. The third plot shows the output labels for the input 1 and 2.

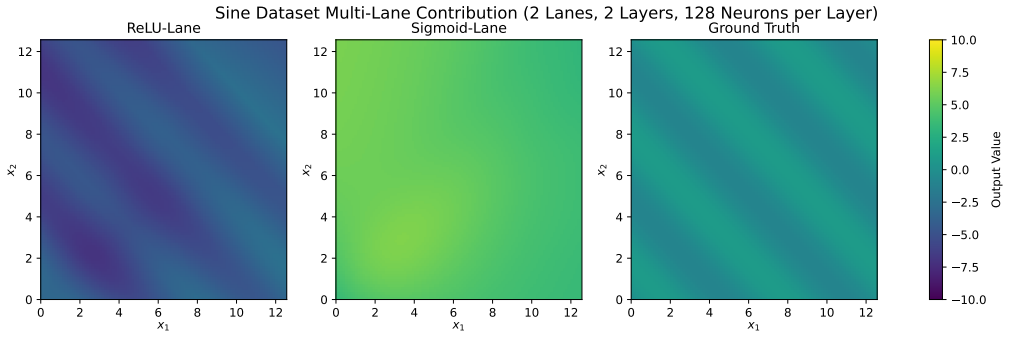

Figure B.29: Comparison of the two lanes within the Doublelane model that is trained on the `Sine` dataset. The x-axis represents the first input while the y-axis represents the second input. The values are color-coded according to the color bar and show the contribution of the lanes to the final output. The first plot shows the contribution of the ReLU lane of the network. The subsequent plot displays the contribution of the Sigmoid lane. The third plot shows the output labels for the input 1 and 2.

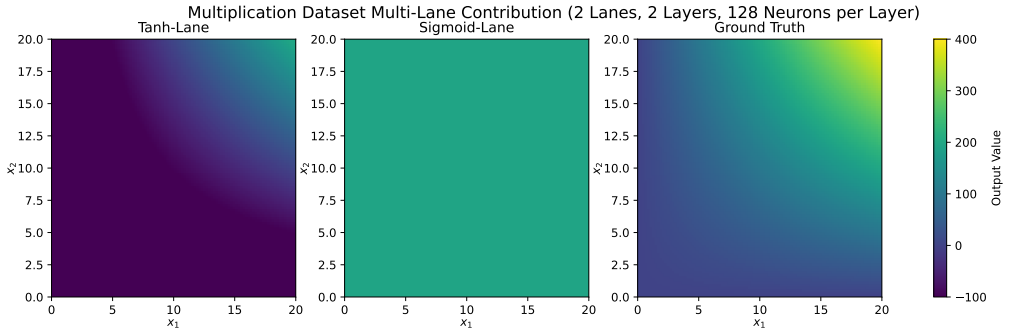

Figure B.30: Comparison of the two lanes within the Doublelane model that is trained on the `Multiplication` dataset. The x-axis represents the first input while the y-axis represents the second input. The values are color-coded according to the color bar and show the contribution of the lanes to the final output. The first plot shows the contribution of the Tanh lane of the network. The subsequent plot displays the contribution of the Sigmoid lane. The third plot shows the output labels for the input 1 and 2.

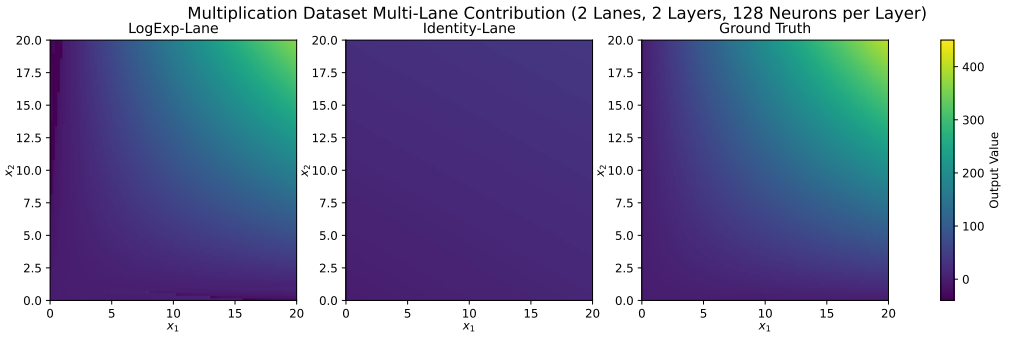

Figure B.31: Comparison of the two lanes within the Doublelane model that is trained on the `Multiplication` dataset. The x-axis represents the first input while the y-axis represents the second input. The values are color-coded according to the color bar and show the contribution of the lanes to the final output. The first plot shows the contribution of the LogExp lane of the network. The subsequent plot displays the contribution of the Identity lane. The third plot shows the output labels for the input 1 and 2.

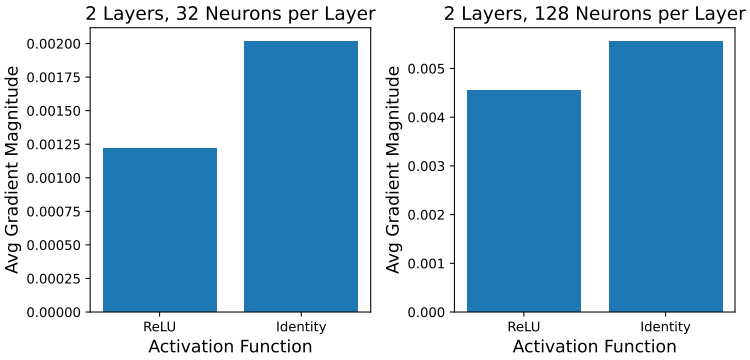

Figure B.32: Comparison of the gradient magnitude for two different lanes within a Double-Lane model. The compared plots were trained on the `Max` dataset, with each one of the models having a different size. The x-axis represents the activation functions of the two lanes. The y-axis indicates the average gradient magnitude of the activations in a lane. Notably, the gradient of the ReLU activations with respect to the loss is smaller than the gradients of identity activations. This does not match the expectation.

B.4 Activation Gradients

This section shows the activation gradients for different Double-Lane models. The expectation is, that the gradient magnitude is higher, when the lane is more important.

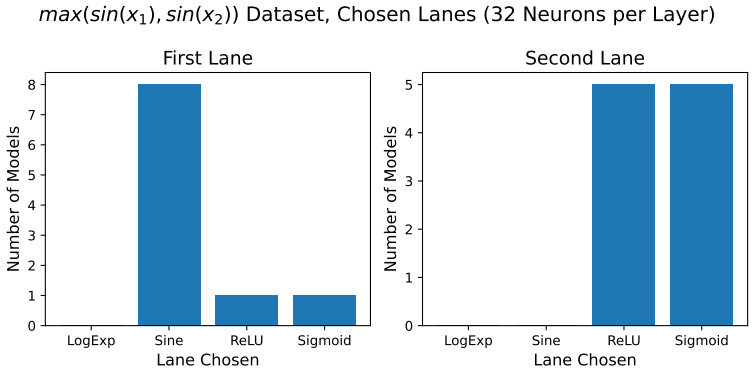

Figure B.33: Shows the lanes chosen from the model with Lane-Loss and the **contribution criterion**. There were ten runs evaluated. The x-axis shows the lanes that were chosen. The y-axis indicates how often the lanes were selected. This model type was trained on the $max(sin(x_1), sin(x_2))$ dataset (see Table 5.3). The model has 32 neurons per layer.

B.5 Lane-Loss

This section provides additional information for our Lane-Loss experiments described in Section 4.3.

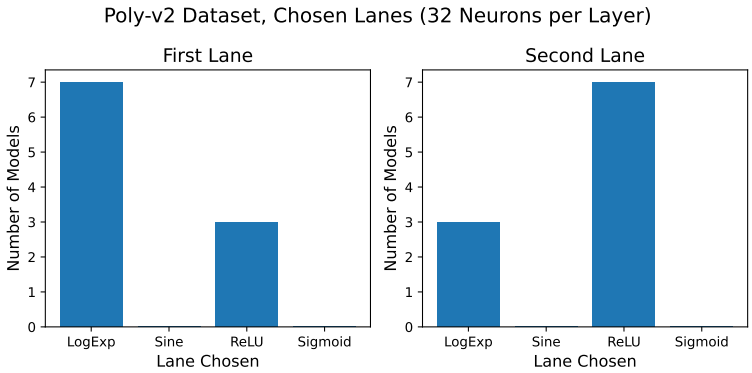

Figure B.34: Shows the lanes chosen from the model with Lane-Loss and the **contribution criterion**. There were ten runs evaluated. The x-axis shows the lanes that were chosen. The y-axis indicates how often the lanes were selected. This model type was trained on the textttPoly-v2 dataset (see Table 5.3). The model has 32 neurons per layer.

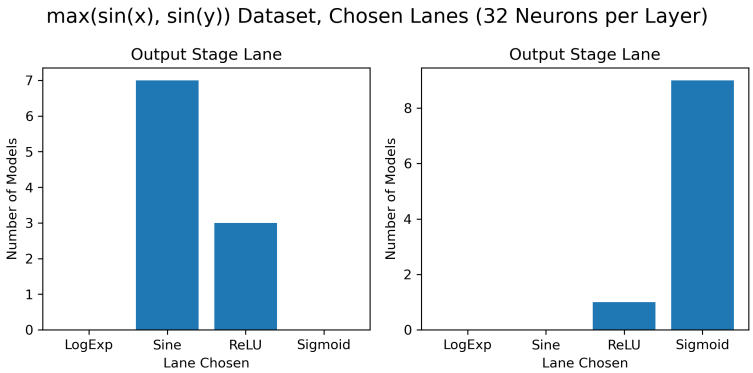

Figure B.35: Shows the lanes chosen from the model with Lane-Loss and the **contribution criterion**. There were ten runs evaluated. The x-axis shows the lanes that were chosen. The y-axis indicates how often the lanes were selected. This model type was trained on the textttPoly-v1 dataset (see Table 5.3). The model has 32 neurons per layer.

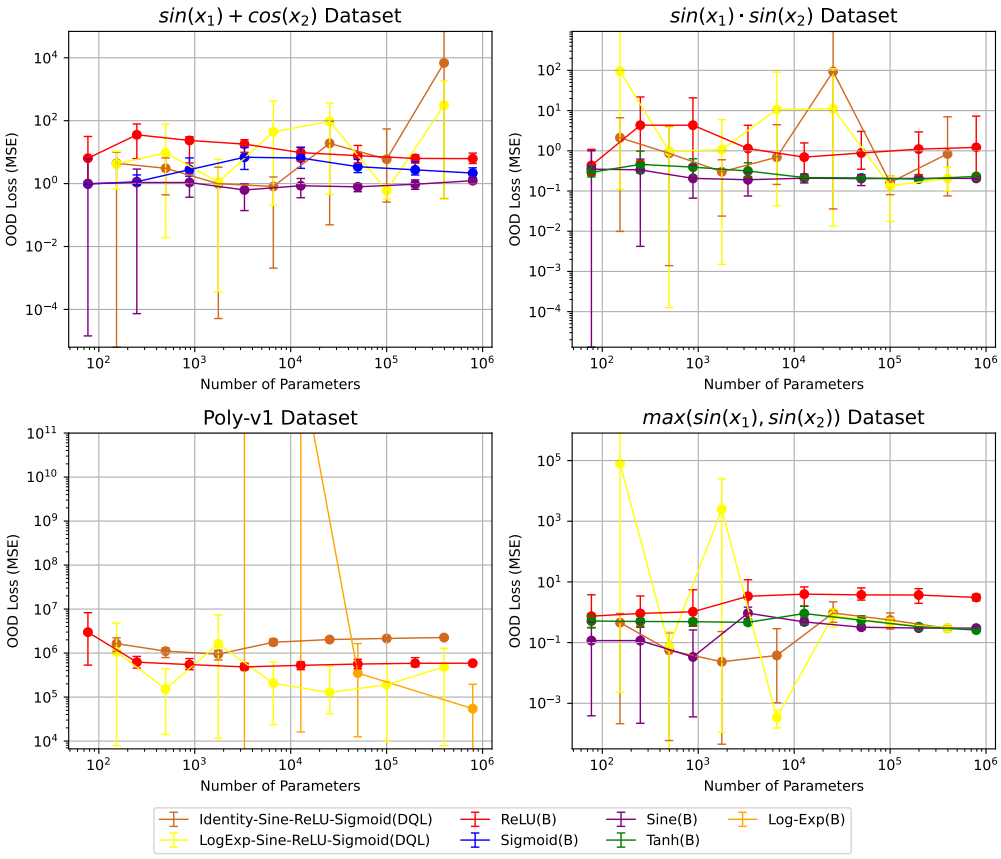

Figure B.36: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets as described in Section 5.3.2. (DQL) indicates that it is a Double Quad-Lane model with Lane-Loss and the **weight** criterion. It has a structure as described in Figure 4.6. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases.

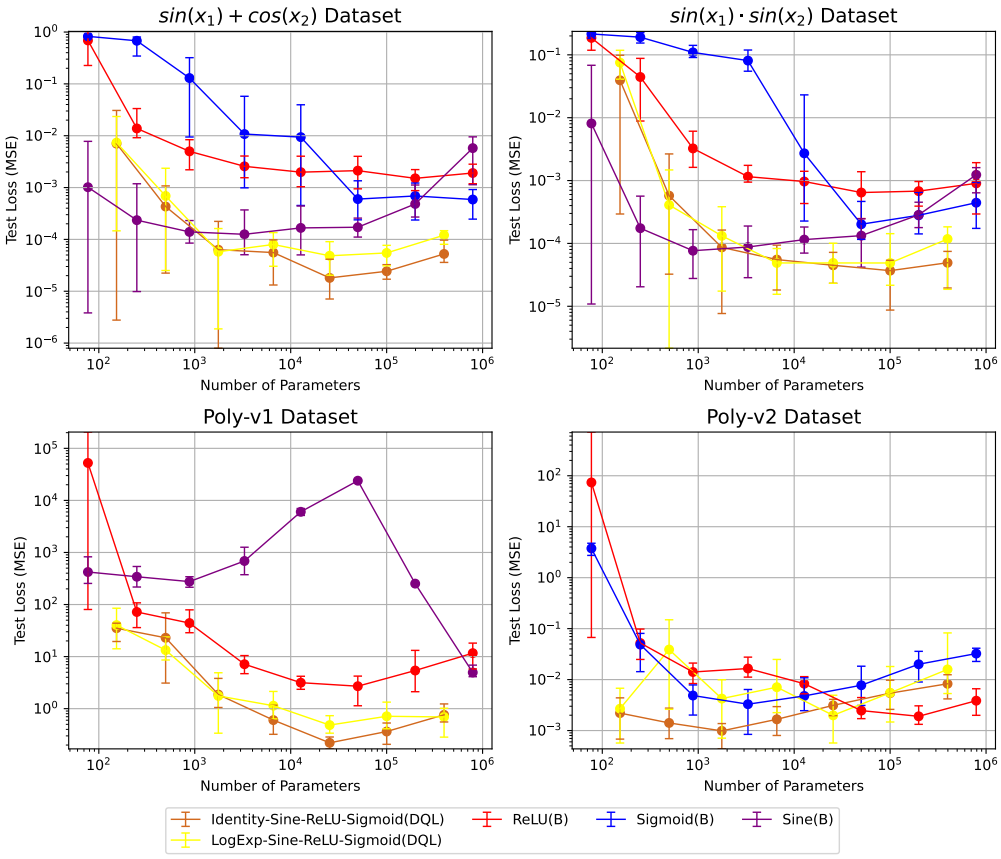

Figure B.37: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets as described in Section 5.3.2. (DQL) indicates that it is a Double Quad-Lane model with Lane-Loss and the **weight** criterion. It has a structure as described in Figure 4.6. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases.

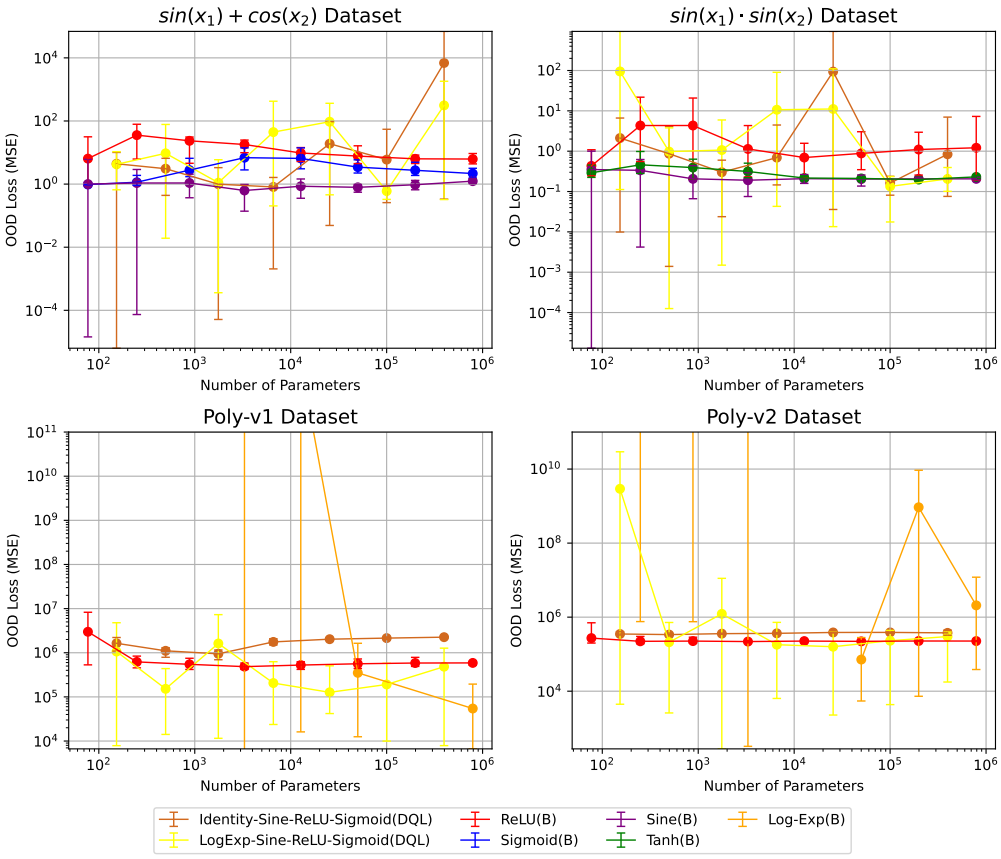

Figure B.38: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets as described in Section 5.3.2. (DQL) indicates that it is a Double Quad-Lane model with Lane-Loss and the **weight** criterion. It has a structure as described in Figure 4.6. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases.

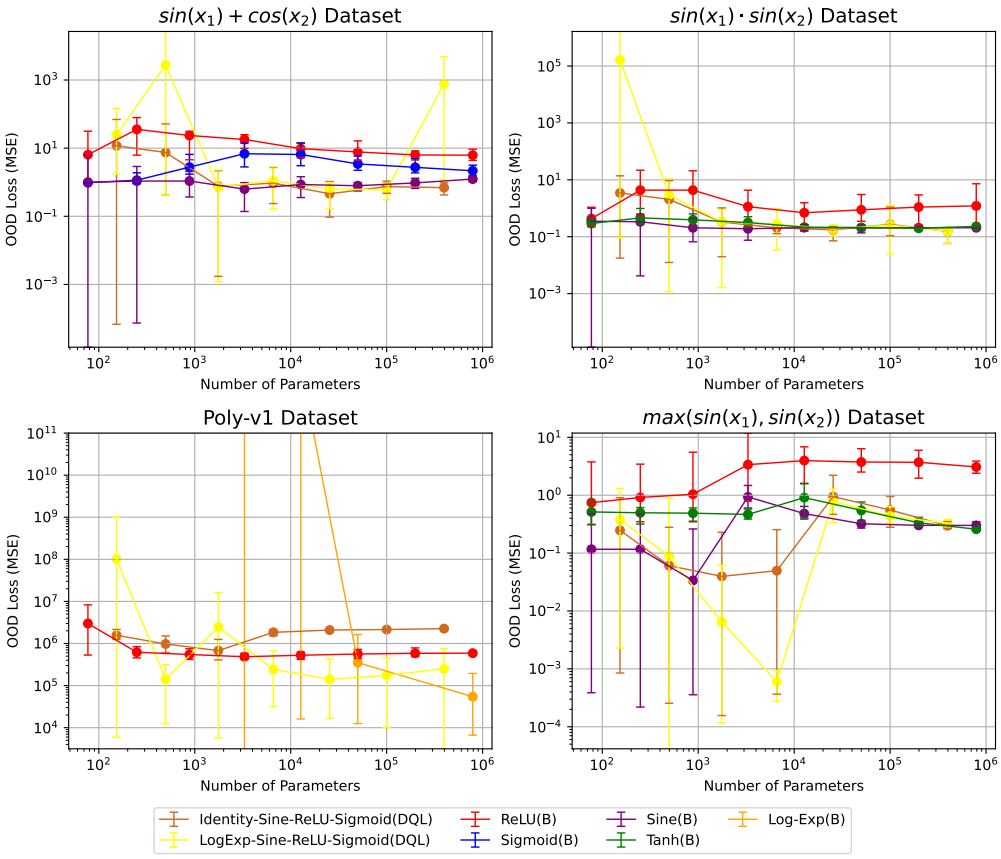

Figure B.39: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets as described in Section 5.3.2. (DQL) indicates that it is a Double Quad-Lane model with Lane-Loss and the **contribution** criterion. It has a structure as described in Figure 4.6. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases.

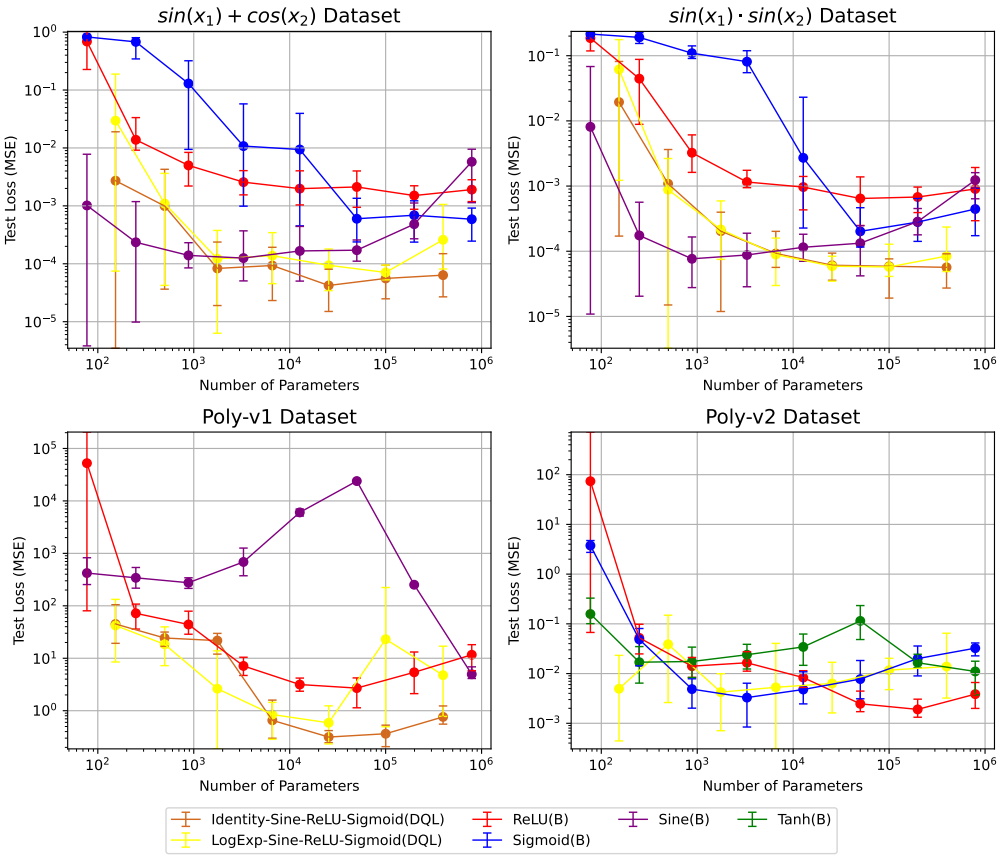

Figure B.40: Comparison of model performance on the **test** set relative to the number of parameters for four different datasets as described in Section 5.3.2. (DQL) indicates that it is a Double Quad-Lane model with Lane-Loss and the **contribution** criterion. It has a structure as described in Figure 4.6. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases.

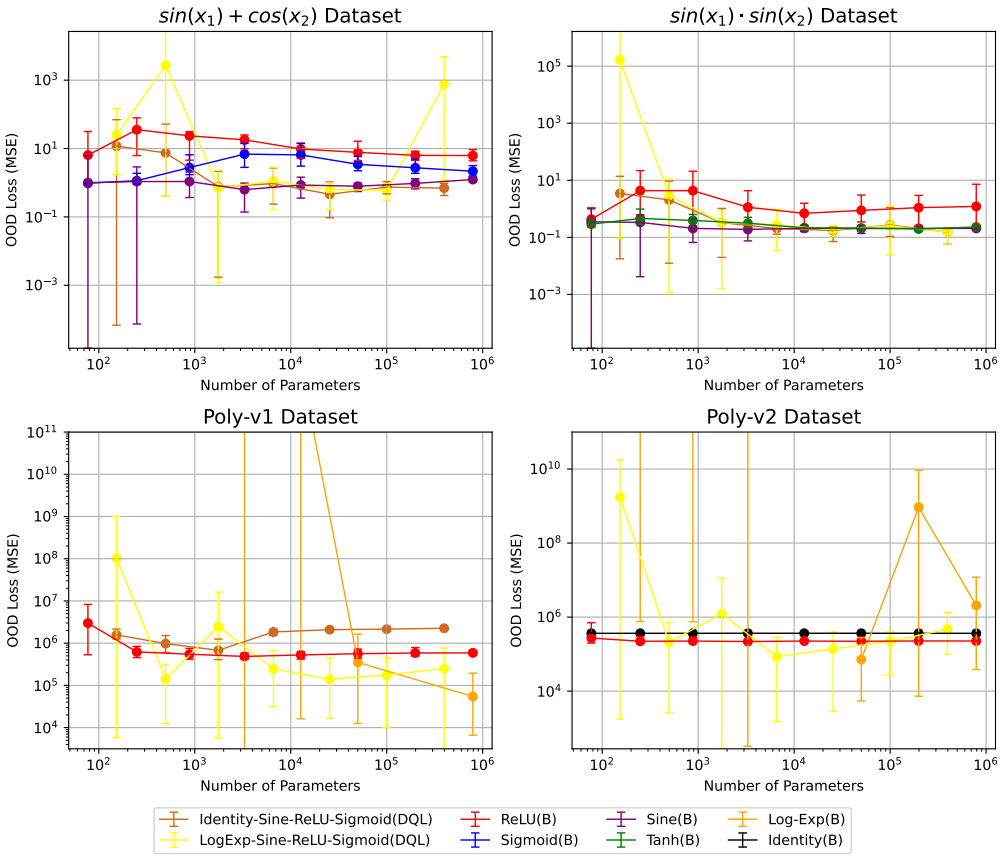

Figure B.41: Comparison of model performance on the **OOD** set relative to the number of parameters for four different datasets as described in Section 5.3.2. (DQL) indicates that it is a Double Quad-Lane model with Lane-Loss and the **contribution** criterion. It has a structure as described in Figure 4.6. (B) marks a pure, single activation function model. The graphs show the trend of the MSE loss on the test set as the model complexity increases.

Appendix C

Overall Results

The following tables show the overall results of the top one hundred models tested on the dataset. They are sorted by 'Test Loss' in ascending order. We also include the number of parameters for each model. For the real-world datasets we add the test accuracy as a reference point.

C.1 CIFAR10 Dataset

Model Type	CE Loss	Acc	Parameters
ReLU-Tanh(S)	1.962488	50.110176	3110410
ReLU(B)	1.963393	50.349559	4210698
ReLU-Tanh-Sigmoid(T)	1.963702	49.600361	5379010
ReLU(B)	1.964530	50.221354	3110410
ReLU-Tanh(MM)	1.964591	49.927885	3082960
ReLU-Tanh(E)	1.964717	49.687500	4210698
ReLU-Sigmoid(D)	1.965195	50.025040	8176010
ReLU-Tanh(S)	1.965273	49.688502	4210698
ReLU-Sine(E)	1.966419	49.910857	3110410
ReLU-Tanh-Sigmoid-Sine(Q)	1.966864	49.203726	5577610
ReLU-Sine(E)	1.967903	49.419071	4210698
ReLU-Tanh(MM)	1.968077	49.720553	2563490
ReLU-Sine(D)	1.968551	49.733574	2790410
ReLU-Tanh-Sine(R)	1.968563	49.294872	3110410
ReLU-Tanh(S)	1.968630	49.602364	1843210
ReLU(B)	1.968685	49.860777	1843210
ReLU-Tanh(E)	1.970221	49.331931	3110410
ReLU-Tanh-Sigmoid-Sine(R)	1.970544	49.028446	4210698
ReLU-Sine-Sigmoid(R)	1.970716	49.496194	4210698
ReLU-Tanh-Sigmoid(T)	1.970783	48.920272	3047410
ReLU-Sine(D)	1.970809	49.565304	3588010
ReLU-Sigmoid-Sine(T)	1.971694	49.074519	3047410

ReLU-Tanh-Sine(T)	1.971694	49.074519	3047410
ReLU-Tanh(D)	1.971991	49.011418	8176010
ReLU-Tanh-Sigmoid(R)	1.972234	49.116587	4210698
ReLU-Tanh-Sigmoid-Sine(Q)	1.972327	48.626803	2979910
ReLU-Sine-Sigmoid(R)	1.972375	49.164663	3110410
ReLU-Sine-Sigmoid(R)	1.972635	49.364984	1843210
ReLU-Sigmoid(D)	1.972747	49.679487	2790410
ReLU-Tanh-Sine(T)	1.972805	48.789062	1971610
ReLU-Sigmoid-Sine(T)	1.972805	48.789062	1971610
ReLU-Sine(E)	1.973419	49.299880	1843210
ReLU-Sigmoid(E)	1.973544	49.167668	4210698
ReLU-Tanh-Sine(R)	1.973885	49.013421	4210698
ReLU-Sigmoid(D)	1.974510	49.221755	3588010
ReLU-Sigmoid(E)	1.975245	49.295873	3110410
ReLU-Tanh-Sigmoid-Sine(Q)	1.975749	48.406450	2489770
ReLU-Tanh-Sigmoid(R)	1.975858	48.886218	1843210
ReLU-Sine(D)	1.976077	49.130609	2069740
ReLU-Tanh-Sigmoid(R)	1.976256	48.792067	3110410
ReLU-Tanh-Sigmoid-Sine(R)	1.976319	49.002404	1843210
ReLU-Tanh(MM)	1.976579	48.911258	1295210
ReLU-Tanh-Sigmoid(T)	1.977429	48.226162	1971610
ReLU-Sigmoid(E)	1.977435	49.034455	1843210
ReLU-Tanh(E)	1.977459	49.061498	1843210
ReLU-Sigmoid(D)	1.977481	49.193710	2069740
ReLU-Tanh-Sigmoid-Sine(R)	1.977531	48.610777	3110410
ReLU-Tanh-Sine(T)	1.978131	48.231170	1254250
ReLU-Sigmoid-Sine(T)	1.978131	48.231170	1254250
ReLU-Tanh(D)	1.978151	48.851162	2790410
ReLU-Sine(D)	1.979025	48.837139	1385170
ReLU-Tanh-Sine(R)	1.979524	48.531651	1843210
ReLU-Tanh-Sigmoid(T)	1.979817	48.113982	1254250
ReLU-Tanh(D)	1.980065	48.599760	2069740
ReLU-Tanh(D)	1.981326	48.488582	3588010
ReLU-Sigmoid(D)	1.982442	48.396434	1385170
ReLU-Tanh-Sigmoid-Sine(Q)	1.984303	47.576122	1274410
ReLU-Sigmoid(MM)	1.984546	47.967748	1295210
ReLU-Tanh(D)	1.986138	48.325321	1385170
ReLU(B)	1.987694	47.887620	411658
ReLU-Tanh(S)	1.987987	47.759415	411658
ReLU-Sine(E)	1.994618	46.893029	411658
ReLU-Tanh(E)	1.996457	46.766827	411658
ReLU-Tanh-Sine(T)	1.997253	46.344151	470410
ReLU-Sigmoid-Sine(T)	1.997542	46.498397	470410
ReLU-Tanh-Sigmoid(T)	1.998358	46.415264	470410

ReLU-Tanh-Sine(R)	1.998475	46.723758	411658
ReLU-Sine(D)	2.000466	46.683694	429226
ReLU-Tanh-Sigmoid(R)	2.002828	46.101763	411658
ReLU-Sine-Sigmoid(R)	2.003175	46.166867	411658
ReLU-Tanh(D)	2.003346	46.244992	429226
Sigmoid-ReLU(S)	2.003527	46.353165	4210698
Sigmoid-ReLU(S)	2.003580	46.361178	3110410
Tanh-ReLU(S)	2.004033	46.279046	4210698
ReLU-Sigmoid(S)	2.004500	46.679688	4210698
Sigmoid-Sine(E)	2.005669	45.709135	4210698
ReLU-Tanh(MM)	2.005671	45.573918	375970
Tanh-ReLU(S)	2.005824	46.200921	3110410
ReLU-Tanh-Sigmoid-Sine(R)	2.006145	45.889423	411658
ReLU-Sigmoid(S)	2.006177	46.721755	3110410
ReLU-Tanh(S)	2.006788	45.535857	201738
Sigmoid-Sine(D)	2.007227	45.518830	8176010
ReLU-Tanh-Sigmoid-Sine(Q)	2.007354	45.756210	373930
Sigmoid-Sine(D)	2.007817	45.490785	3588010
ReLU-Sigmoid(D)	2.008373	45.564904	429226
Tanh-ReLU(S)	2.008376	45.816306	1843210
Tanh-Sine(E)	2.008643	45.677083	1843210
Sigmoid-ReLU(S)	2.009046	46.007612	1843210
Sigmoid-Sine(E)	2.009299	45.312500	3110410
ReLU(B)	2.009320	45.355569	201738
Sigmoid-Sine(E)	2.009430	45.528846	1843210
ReLU-Sigmoid(E)	2.010163	45.400641	411658
Sigmoid-Sine(D)	2.010603	45.337540	2790410
Tanh-Sine(D)	2.011138	45.206330	2069740
Sigmoid-Sine(D)	2.011147	45.529848	2069740
ReLU-Sigmoid(S)	2.011289	46.288061	1843210
Tanh-Sine(E)	2.012050	45.034054	4210698
Tanh-Sine(D)	2.012484	44.968950	2790410
Tanh-Sine(D)	2.012712	45.083133	3588010
Tanh-Sine(D)	2.012991	44.667468	8176010

C.2 ISOLET Dataset

Model Type	CE Loss	Acc	Parameters
ReLU-Tanh-Sigmoid-Sine(Q)	2.362502	96.198630	1675226
ReLU-Tanh-Sigmoid(T)	2.362526	96.172945	1720526
ReLU-Sigmoid-Sine(T)	2.362551	96.232877	303056

ReLU-Tanh-Sine(T)	2.362551	96.232877	303056
ReLU-Sigmoid-Sine(T)	2.362609	96.232877	852326
ReLU-Tanh-Sine(T)	2.362609	96.232877	852326
ReLU-Tanh-Sigmoid-Sine(Q)	2.362810	96.104452	636146
ReLU-Tanh-Sigmoid(T)	2.362962	96.190068	508226
ReLU-Tanh-Sigmoid(T)	2.363016	96.164384	852326
ReLU-Tanh-Sigmoid(T)	2.363073	96.155822	303056
ReLU-Sigmoid-Sine(T)	2.363195	96.087329	1720526
ReLU-Tanh-Sigmoid-Sine(Q)	2.363307	96.344178	298826
ReLU-Sigmoid(D)	2.363322	96.190068	1149026
ReLU-Sigmoid(S)	2.363352	96.104452	99482
ReLU-Tanh-Sine(T)	2.363383	96.130137	1720526
ReLU-Sine(D)	2.363402	96.181507	581966
ReLU-Tanh(D)	2.363515	96.198630	1149026
ReLU-Tanh(MM)	2.363552	96.010274	2798026
ReLU-Sigmoid(D)	2.363731	96.138699	581966
Sigmoid-Sine(E)	2.363797	96.087329	1159226
ReLU-Sine(D)	2.363801	96.070205	1149026
ReLU-Sine(MM)	2.363878	96.078767	1998426
ReLU-Sine(MM)	2.363884	96.087329	2798026
ReLU-Sigmoid-Sine(T)	2.364000	96.121575	3472826
ReLU-Sigmoid(D)	2.364017	96.190068	839226
ReLU-Tanh(S)	2.364076	96.104452	1713178
ReLU-Sine(S)	2.364082	96.104452	594458
ReLU-Sine(E)	2.364119	96.095890	594458
Sigmoid-Sine(D)	2.364129	96.035959	839226
ReLU-Sigmoid(MM)	2.364154	96.147260	709866
ReLU-Tanh(D)	2.364164	95.984589	839226
ReLU-Tanh(MM)	2.364179	96.053082	709866
ReLU-Sine(D)	2.364184	96.095890	360806
ReLU-Tanh-Sigmoid(T)	2.364265	96.104452	3472826
ReLU-Tanh(S)	2.364272	96.001712	594458
ReLU-Tanh-Sigmoid(R)	2.364281	96.138699	594458
ReLU-Sigmoid(E)	2.364295	96.104452	1159226
ReLU(B)	2.364314	96.061644	594458
ReLU(B)	2.364324	95.958904	1159226
ReLU-Sine(D)	2.364375	96.078767	839226
ReLU-Tanh-Sigmoid-Sine(R)	2.364436	96.155822	1159226
ReLU-Tanh(MM)	2.364459	95.984589	1998426
ReLU-Sigmoid(MM)	2.364496	96.061644	2798026
ReLU-Sigmoid(MM)	2.364566	96.035959	887876
ReLU-Tanh(E)	2.364625	95.916096	5523482
ReLU-Sine(E)	2.364632	95.984589	1159226
Tanh-Sigmoid(S)	2.364712	96.044521	594458

ReLU-Sine-Sigmoid(R)	2.364725	96.001712	1713178
ReLU-Sine(E)	2.364759	96.001712	5523482
Sigmoid-Sine(D)	2.364765	96.053082	3298026
ReLU-Tanh(D)	2.364789	96.087329	581966
ReLU-Tanh-Sine(T)	2.364808	95.890411	3472826
ReLU-Tanh-Sigmoid(R)	2.364823	95.984589	5523482
Sine(B)	2.364825	96.095890	1159226
ReLU-Tanh-Sigmoid(R)	2.364830	96.044521	1159226
ReLU-Sigmoid(D)	2.364854	95.984589	3298026
ReLU-Tanh(D)	2.364854	96.018836	3298026
ReLU-Sine-Sigmoid(R)	2.364861	95.967466	1159226
ReLU-Sigmoid(E)	2.364913	96.010274	594458
ReLU-Sine(MM)	2.364929	96.018836	319626
ReLU-Tanh-Sine(T)	2.364938	96.053082	104576
ReLU-Sigmoid-Sine(T)	2.364938	96.053082	104576
ReLU-Sigmoid(E)	2.364979	95.950342	5523482
ReLU-Tanh-Sigmoid-Sine(R)	2.365003	96.018836	1713178
ReLU-Tanh-Sine(R)	2.365048	96.035959	1159226
ReLU(B)	2.365063	95.967466	1713178
Sine(B)	2.365094	96.044521	594458
ReLU-Tanh(MM)	2.365102	96.001712	319626
ReLU-Tanh(E)	2.365140	95.916096	1159226
ReLU-Sine(E)	2.365186	95.958904	1713178
ReLU-Tanh(S)	2.365188	95.890411	5523482
ReLU-Sigmoid(MM)	2.365223	95.941781	1998426
Sine-ReLU(S)	2.365240	95.941781	1713178
ReLU(B)	2.365307	95.941781	5523482
ReLU-Sine-Sigmoid(R)	2.365321	96.035959	594458
ReLU-Sigmoid(E)	2.365325	96.035959	1713178
ReLU-Tanh(MM)	2.365343	95.821918	887876
Tanh-Sine(D)	2.365402	95.976027	1149026
Sine-ReLU(S)	2.365428	96.087329	594458
ReLU-Tanh-Sigmoid(R)	2.365456	96.010274	99482
ReLU-Tanh(S)	2.365489	95.907534	1159226
Sine(B)	2.365489	95.933219	1713178
Tanh-Sine(D)	2.365539	96.070205	839226
ReLU-Tanh-Sigmoid(T)	2.365559	95.933219	104576
ReLU-Tanh(D)	2.365622	95.779110	360806
ReLU-Sine(MM)	2.365632	95.933219	887876
ReLU-Sine(S)	2.365644	95.933219	1713178
ReLU-Tanh(E)	2.365680	95.967466	594458
ReLU-Sine(S)	2.365708	95.881849	1159226
ReLU-Sine(S)	2.365716	95.856164	5523482
ReLU-Sine(MM)	2.365733	95.847603	709866

Tanh-Sigmoid(S)	2.365809	96.035959	1713178
ReLU-Tanh(D)	2.365828	95.950342	97538
Sigmoid-Sine(E)	2.365845	95.967466	5523482
Sigmoid-Sine(D)	2.365861	96.018836	360806
Tanh-Sigmoid(S)	2.365886	96.001712	1159226
ReLU-Tanh-Sigmoid-Sine(Q)	2.365927	96.044521	81266
ReLU-Sigmoid(D)	2.365959	95.856164	360806
ReLU-Tanh-Sigmoid(R)	2.365959	95.864726	1713178
ReLU-Sine(D)	2.365961	95.864726	3298026

C.3 FashionMNIST Dataset

Model Type	CE Loss	Acc	Parameters
ReLU-Tanh-Sine(T)	1.565733	89.576322	3835210
ReLU-Tanh-Sigmoid(T)	1.567089	89.405048	3835210
ReLU-Sine(D)	1.567153	89.373998	3600010
ReLU-Tanh-Sigmoid-Sine(Q)	1.567188	89.476162	1916810
ReLU-Tanh-Sigmoid-Sine(Q)	1.567202	89.437099	6112810
ReLU-Sine(E)	1.567705	89.384014	1867786
ReLU-Tanh-Sine(R)	1.568096	89.291867	1867786
Tanh-Sine(D)	1.568134	89.268830	3600010
ReLU-Tanh-Sine(T)	1.568700	89.263822	1947010
ReLU-Tanh(D)	1.568733	89.226763	3600010
ReLU-Tanh-Sigmoid-Sine(Q)	1.568884	89.206731	920710
ReLU-Tanh-Sine(T)	1.569425	89.214744	988210
ReLU-Tanh(MM)	1.569652	89.111579	3100010
Sigmoid-Sine(D)	1.569712	89.126603	3600010
ReLU-Sine(D)	1.569877	89.094551	1300010
ReLU-Sigmoid(D)	1.569885	89.083534	3600010
ReLU-Tanh-Sigmoid(T)	1.570005	89.119591	1947010
ReLU-Tanh-Sine(T)	1.570204	89.135617	598810
ReLU-Sine(D)	1.570210	89.095553	960010
Tanh-Sine(E)	1.570255	89.063502	1867786
ReLU(B)	1.570297	89.057492	1867786
ReLU-Sigmoid(MM)	1.570385	89.046474	2240010
ReLU-Tanh-Sigmoid-Sine(R)	1.570511	89.021434	1867786
Sine(B)	1.570538	89.047476	1867786
ReLU-Sine(D)	1.570598	89.045473	674060
ReLU-Sine(E)	1.570709	89.028446	1280010
ReLU-Tanh-Sigmoid-Sine(Q)	1.570715	89.075521	750890
ReLU(B)	1.570861	89.013421	1280010

Sigmoid-Sine(E)	1.571016	88.997396	1867786
ReLU(B)	1.571086	88.985377	671754
ReLU-Sigmoid(MM)	1.571236	88.989383	3100010
ReLU-Sine(E)	1.571329	88.983373	671754
ReLU-Tanh-Sigmoid(T)	1.571410	88.985377	988210
Sine(B)	1.571551	88.915264	1280010
ReLU-Tanh(E)	1.571644	88.934295	1867786
ReLU-Tanh-Sigmoid-Sine(R)	1.571894	88.902244	1280010
Tanh-Sine(E)	1.571918	88.946314	1280010
ReLU-Tanh(MM)	1.571938	88.897236	2240010
ReLU-Sigmoid(E)	1.571940	88.871194	1867786
Sigmoid-Sine(E)	1.572476	88.839143	1280010
ReLU-Tanh-Sine(R)	1.572511	88.850160	1280010
ReLU-Tanh(MM)	1.572613	88.822115	1023760
ReLU-Tanh(D)	1.572805	88.820112	1300010
ReLU-Tanh-Sine(R)	1.572815	88.835136	671754
Sigmoid-ReLU(S)	1.572844	88.865184	1280010
ReLU-Sigmoid(S)	1.572952	88.815104	1867786
ReLU-Sigmoid(D)	1.573153	88.781050	1300010
ReLU-Tanh(D)	1.573188	88.782051	960010
ReLU-Sigmoid(D)	1.573422	88.739984	960010
ReLU-Tanh(S)	1.573456	88.725962	1867786
Tanh-Sine(D)	1.573568	88.731971	1300010
ReLU-Tanh-Sigmoid(R)	1.573576	88.738982	1867786
Sine(B)	1.573651	88.730970	671754
ReLU-Sigmoid(E)	1.573739	88.730970	1280010
Sigmoid-ReLU(S)	1.573780	88.814103	1867786
Tanh-Sine(D)	1.573854	88.721955	674060
ReLU-Tanh-Sigmoid(T)	1.573978	88.694912	598810
Sigmoid-Sine(D)	1.574030	88.708934	1300010
Tanh-ReLU(S)	1.574031	88.776042	1867786
ReLU-Tanh(S)	1.574100	88.699920	1280010
Tanh-Sine(D)	1.574171	88.686899	960010
ReLU-Tanh(E)	1.574650	88.629808	1280010
ReLU-Tanh(MM)	1.574665	88.585737	824610
ReLU-Sine(D)	1.574701	88.679888	424210
ReLU-Sigmoid(S)	1.574826	88.611779	1280010
ReLU-Tanh(S)	1.574854	88.597756	671754
Tanh-ReLU(S)	1.575039	88.578726	1280010
ReLU-Tanh-Sigmoid-Sine(R)	1.575100	88.572716	671754
ReLU-Tanh(E)	1.575139	88.591747	671754
ReLU-Sigmoid(E)	1.575265	88.613782	671754
Sigmoid-Sine(D)	1.575434	88.556691	960010
Sigmoid-Sine(E)	1.575442	88.544671	671754

Sigmoid-Tanh(S)	1.575503	88.636819	1867786
ReLU-Sigmoid(D)	1.575652	88.558694	674060
ReLU-Sigmoid(MM)	1.575744	88.544671	1023760
Tanh-Sine(E)	1.575969	88.532652	671754
ReLU-Tanh-Sigmoid(R)	1.576200	88.445513	671754
ReLU-Sigmoid(MM)	1.576266	88.469551	824610
ReLU-Tanh(D)	1.576307	88.453526	674060
ReLU-Tanh-Sigmoid(R)	1.576571	88.434495	1280010
ReLU-Tanh-Sine(T)	1.576758	88.469551	361930
Sigmoid-ReLU(S)	1.576957	88.504607	671754
ReLU-Tanh-Sigmoid-Sine(Q)	1.577326	88.431490	359210
ReLU-Tanh-Sigmoid(T)	1.577645	88.343349	361930
ReLU-Sigmoid(S)	1.577789	88.395433	671754
Tanh-ReLU(S)	1.578113	88.324319	671754
ReLU-Tanh(D)	1.578894	88.233173	424210
Sigmoid-Sine(D)	1.579038	88.211138	674060
Sigmoid-Tanh(S)	1.579225	88.208133	1280010
ReLU-Tanh(MM)	1.579255	88.190104	380010
Tanh-Sine(D)	1.579580	88.113982	424210
Sigmoid-Sine(D)	1.579770	88.118990	424210
Tanh(B)	1.579978	88.109976	1867786
ReLU-Sigmoid(D)	1.580019	88.115986	424210
Tanh-Sigmoid(S)	1.582334	87.893630	671754
ReLU-Sigmoid(MM)	1.582337	87.894631	380010
Tanh-Sigmoid(D)	1.582413	87.844551	3600010
Sigmoid-Tanh(S)	1.582808	87.893630	671754
Tanh-Sigmoid(S)	1.582850	87.825521	1867786
Tanh(B)	1.583071	87.782452	1280010

C.4 MNIST Dataset

Model Type	CE Loss	Acc	Parameters
ReLU-Tanh-Sigmoid-Sine(Q)	1.478258	98.306290	6112810
ReLU-Tanh-Sine(T)	1.478435	98.296274	1947010
ReLU-Tanh-Sine(T)	1.478555	98.262220	3835210
ReLU-Sine(E)	1.478647	98.249199	5832714
ReLU(B)	1.478817	98.270232	1280010
ReLU-Sigmoid(E)	1.478965	98.242188	5832714
ReLU-Sine(E)	1.479002	98.217147	1280010
ReLU-Tanh-Sigmoid(T)	1.479052	98.241186	3835210
ReLU(B)	1.479091	98.230168	5832714

ReLU-Sine(E)	1.479128	98.228165	1867786
ReLU-Sine(D)	1.479134	98.224159	1300010
ReLU-Tanh(E)	1.479181	98.210136	5832714
ReLU-Tanh(E)	1.479262	98.226162	1867786
ReLU(B)	1.479395	98.219151	1867786
ReLU-Sine(D)	1.479476	98.205128	960010
Tanh-ReLU(S)	1.479478	98.218149	1280010
ReLU-Sigmoid(MM)	1.479532	98.182091	3100010
ReLU-Sigmoid(MM)	1.479555	98.194111	2240010
ReLU-Sigmoid(D)	1.479563	98.206130	1300010
Tanh-ReLU(S)	1.479572	98.219151	5832714
ReLU-Tanh-Sigmoid-Sine(Q)	1.479616	98.168069	1916810
ReLU-Sigmoid(S)	1.479617	98.190104	1867786
ReLU-Tanh-Sigmoid-Sine(R)	1.479618	98.201122	1867786
Tanh-ReLU(S)	1.479618	98.236178	1867786
ReLU-Tanh-Sigmoid(T)	1.479679	98.194111	1947010
ReLU-Sigmoid(S)	1.479688	98.202123	5832714
ReLU-Tanh-Sigmoid(T)	1.479739	98.172075	988210
ReLU-Tanh-Sine(T)	1.479766	98.170072	988210
ReLU-Sigmoid(S)	1.479806	98.220152	1280010
ReLU-Tanh-Sigmoid(R)	1.479817	98.173077	1280010
ReLU-Tanh(MM)	1.479821	98.141026	3100010
ReLU-Tanh(E)	1.479829	98.137019	1280010
ReLU(B)	1.479937	98.158053	671754
ReLU-Tanh-Sigmoid-Sine(R)	1.480023	98.137019	1280010
ReLU-Sigmoid(E)	1.480023	98.184095	1867786
ReLU-Tanh(D)	1.480080	98.149038	1300010
ReLU-Tanh-Sigmoid(R)	1.480122	98.149038	5832714
ReLU-Tanh(E)	1.480147	98.149038	671754
ReLU-Tanh(S)	1.480150	98.107973	1867786
ReLU-Tanh(MM)	1.480153	98.114984	2240010
ReLU-Sigmoid(E)	1.480188	98.141026	1280010
ReLU-Sine(E)	1.480208	98.136018	671754
ReLU-Sigmoid(S)	1.480225	98.151042	671754
ReLU-Tanh(S)	1.480248	98.117989	671754
ReLU-Tanh-Sigmoid(R)	1.480281	98.118990	1867786
ReLU-Sigmoid(D)	1.480340	98.123998	960010
ReLU-Sine(D)	1.480343	98.113982	674060
ReLU-Tanh-Sigmoid-Sine(Q)	1.480357	98.133013	750890
ReLU-Tanh(S)	1.480392	98.102965	1280010
ReLU-Tanh-Sigmoid-Sine(Q)	1.480405	98.115986	920710
ReLU-Tanh(D)	1.480426	98.109976	960010
ReLU-Tanh(D)	1.480441	98.126002	674060
ReLU-Sigmoid(MM)	1.480442	98.120994	824610

Tanh-ReLU(S)	1.480454	98.135016	671754
ReLU-Tanh(S)	1.480462	98.065905	5832714
Sine(B)	1.480462	98.096955	5832714
ReLU-Tanh(MM)	1.480583	98.072917	1023760
ReLU-Sigmoid(MM)	1.480608	98.120994	1023760
ReLU-Sigmoid(E)	1.480631	98.104968	671754
ReLU-Sine(D)	1.480634	98.089944	424210
ReLU-Tanh-Sine(T)	1.480664	98.081931	598810
ReLU-Sigmoid(D)	1.480770	98.085938	674060
Sigmoid-ReLU(S)	1.480898	98.116987	1867786
ReLU-Tanh-Sigmoid-Sine(R)	1.480909	98.051883	671754
ReLU-Tanh-Sigmoid(T)	1.480967	98.064904	598810
ReLU-Tanh(MM)	1.481008	98.058894	824610
ReLU-Tanh-Sigmoid(R)	1.481105	98.057893	671754
ReLU-Tanh(D)	1.481262	98.046875	424210
Sigmoid-ReLU(S)	1.481458	98.056891	1280010
Sine(B)	1.481475	97.987780	1867786
ReLU-Tanh-Sine(T)	1.481689	97.986779	361930
Sigmoid-ReLU(S)	1.481708	98.003806	671754
ReLU-Tanh(MM)	1.481722	97.999800	380010
ReLU-Sigmoid(D)	1.481737	98.004808	424210
ReLU-Sigmoid(MM)	1.481802	97.970753	380010
ReLU-Tanh-Sigmoid(T)	1.481960	97.968750	361930
Sine(B)	1.482045	97.933694	1280010
Sigmoid-Tanh(S)	1.482130	97.959736	5832714
ReLU-Tanh-Sigmoid-Sine(Q)	1.482320	97.935697	359210
Tanh-Sine(E)	1.482562	97.879607	1867786
Tanh-Sine(E)	1.482575	97.888622	1280010
Tanh-Sigmoid(S)	1.482674	97.901643	1867786
Sine(B)	1.482929	97.859575	671754
Tanh-Sigmoid(S)	1.483014	97.862580	1280010
Sigmoid-Tanh(S)	1.483023	97.858574	1867786
Tanh(B)	1.483084	97.841546	1867786
Tanh-Sigmoid(S)	1.483134	97.878606	671754
Tanh(B)	1.483242	97.838542	1280010
Tanh-Sine(D)	1.483258	97.800481	1300010
Tanh-Sine(D)	1.483470	97.794471	960010
Tanh-Sine(D)	1.483541	97.804487	674060
Tanh-Sine(E)	1.483561	97.779447	671754
Tanh-Sigmoid(E)	1.483567	97.793470	1867786
Tanh(B)	1.483586	97.766426	5832714
ReLU(B)	1.483735	97.823518	118794
Sigmoid-Tanh(S)	1.483792	97.808494	1280010
Tanh-Sigmoid(E)	1.484068	97.730369	5832714

Tanh-Sigmoid(S)	1.484072	97.741386	5832714
ReLU-Sigmoid(S)	1.484103	97.785457	118794
ReLU-Tanh(S)	1.484271	97.745393	118794

C.5 Addition Dataset

Model Type	MSE Loss	Parameters
Identity(B)	4.218541e-11	105
Identity(B)	4.334937e-11	337
Identity(B)	1.139856e-10	1185
Identity-Sine(MM)	1.251248e-10	65
Identity-ReLU(S)	1.340950e-10	105
Identity-Sine(MM)	4.224948e-10	545
Identity(B)	7.803106e-10	4417
Identity(B)	8.050292e-09	66817
Identity(B)	9.973964e-09	17025
Identity-ReLU(S)	2.738191e-08	264705
ReLU-Identity(MM)	2.799264e-08	65
ReLU-Identity(D)	7.440819e-08	73
Identity(B)	1.010980e-07	264705
ReLU-Identity(S)	1.880120e-07	105
ReLU-Identity-Sine(R)	2.888514e-07	105
Identity-ReLU(S)	3.143518e-07	337
Identity-Sine(E)	3.491331e-07	1185
ReLU-Identity(E)	3.958540e-07	105
Identity-Sine(MM)	4.873608e-07	177
ReLU(B)	5.008985e-07	105
Identity-Sine(MM)	5.107796e-07	1857
Identity-Tanh(MM)	5.252917e-07	545
Identity-Sine(E)	5.575056e-07	105
Identity-Sine(D)	5.629378e-07	73
ReLU-Identity-Sine-Sigmoid(Q)	6.376322e-07	57
Identity-Tanh(MM)	6.442939e-07	65
ReLU-Identity-Sine(T)	6.859600e-07	109
Identity-Sine(D)	7.289137e-07	673
Identity-Tanh(D)	9.142937e-07	73
ReLU-Identity-Sine-Tanh(Q)	9.618030e-07	57
Identity-ReLU(S)	1.013747e-06	66817
Identity-Sigmoid(D)	1.035564e-06	73
Identity-Sigmoid(MM)	1.037650e-06	65
Sigmoid-Identity-Sine-Tanh(Q)	1.198273e-06	57

Identity-Tanh(MM)	1.310169e-06	177
Identity-Sine(D)	1.430256e-06	209
Identity-Sine(E)	1.472089e-06	4417
ReLU-Sine(MM)	1.547298e-06	65
Identity-Sine-Tanh(T)	1.719859e-06	109
Identity-Sine(E)	1.804996e-06	337
Sigmoid-Identity-Sine-Tanh(Q)	1.969518e-06	145
Identity-Sine(D)	1.991259e-06	2369
Identity-ReLU(S)	2.164832e-06	17025
Identity-Sigmoid(MM)	2.281560e-06	177
ReLU-Identity-Sine-Sigmoid(Q)	2.282133e-06	145
Identity-ReLU(S)	2.365801e-06	4417
ReLU-Identity-Sine-Tanh(Q)	2.486831e-06	145
ReLU-Identity(D)	2.491241e-06	209
ReLU-Sine(D)	2.563428e-06	209
Identity-Tanh(D)	2.679652e-06	209
ReLU-Identity-Sine-Sigmoid(Q)	2.886712e-06	417
Identity-Sine(MM)	3.350699e-06	6785
Identity-Tanh(MM)	3.372365e-06	1857
ReLU-Identity(E)	3.392316e-06	337
Sigmoid-Identity-Sine-Tanh(Q)	3.408533e-06	417
Identity-Tanh(D)	3.486892e-06	673
ReLU-Sine(E)	3.555582e-06	337
Identity-Sine-Tanh(T)	3.655686e-06	313
ReLU-Identity-Sine(T)	3.675826e-06	313
ReLU-Identity-Tanh(T)	3.727607e-06	109
Identity-Sigmoid(D)	3.740584e-06	209
Identity-Sigmoid(E)	4.103480e-06	105
ReLU-Identity-Tanh(R)	4.190486e-06	105
Identity-Sigmoid(MM)	4.309065e-06	545
ReLU-Identity(MM)	4.569288e-06	177
ReLU(B)	5.157250e-06	337
Identity-Tanh(D)	5.394374e-06	2369
Identity-Tanh(E)	5.436198e-06	105
Identity-Tanh(E)	5.849986e-06	1185
ReLU-Identity-Sine(R)	6.448754e-06	337
ReLU-Identity-Sine-Tanh(Q)	6.580093e-06	417
Identity-ReLU(S)	6.834402e-06	1185
Identity-Sigmoid(D)	6.938043e-06	673
ReLU-Identity(S)	7.033896e-06	337
Identity-Sine-Tanh(T)	7.236945e-06	3553
ReLU-Identity-Sine-Tanh-Sigmoid(R)	7.277392e-06	105
ReLU-Identity-Sine(R)	8.197271e-06	1185
ReLU-Sine(D)	8.515829e-06	673

ReLU-Identity(D)	8.692030e-06	673
Identity-Sine-Tanh(R)	8.732929e-06	337
Identity-Sine-Tanh(T)	9.186107e-06	1009
Identity-Sigmoid(D)	9.483295e-06	2369
Identity-Sine-Tanh(R)	9.648039e-06	1185
Identity-Sigmoid(MM)	9.770781e-06	1857
Identity-Tanh(D)	1.036943e-05	8833
Sigmoid-Identity-Sine-Tanh(Q)	1.145824e-05	4737
Identity-Tanh(MM)	1.147140e-05	6785
ReLU-Identity-Tanh(T)	1.169524e-05	313
Identity-Tanh(E)	1.203589e-05	337
Identity-Sigmoid(D)	1.249842e-05	8833
Identity-Sigmoid(E)	1.283536e-05	337
ReLU-Identity(S)	1.362445e-05	1185
ReLU-Identity-Sine(T)	1.362587e-05	1009
Sigmoid-Identity-Sine-Tanh(Q)	1.408885e-05	1345
ReLU-Sigmoid-Tanh(T)	1.411536e-05	313
Identity-Tanh(D)	1.428864e-05	133633
Identity-Sigmoid(MM)	1.433660e-05	25857
ReLU-Identity(MM)	1.452601e-05	545
Sigmoid-Identity-Sine-Tanh(Q)	1.517340e-05	17665
ReLU-Identity-Sine-Sigmoid(Q)	1.542206e-05	1345

C.6 Sine Dataset

Model Type	MSE Loss	Parameters
Identity-Sine(MM)	5.339271e-08	545
Sine-Identity(S)	5.603382e-08	4417
Identity-Sine(S)	1.138129e-07	4417
Identity-Sine(MM)	1.283310e-07	6785
Identity-Sine(S)	2.723283e-07	17025
Sine-Identity(S)	2.914424e-07	17025
Identity-Sine(MM)	3.532560e-07	25857
Identity-Sine(MM)	1.584273e-06	100865
Identity-Sine(S)	3.268685e-06	66817
Sine(B)	4.903959e-06	17025
Sine(B)	6.255450e-06	4417
Sine(B)	7.029013e-06	1185
Sine(B)	7.146952e-06	105
ReLU-Sine(S)	7.222555e-06	4417
Sine-Identity(S)	8.536661e-06	264705

Sine-Identity(S)	1.004256e-05	66817
ReLU-Sine(S)	1.032311e-05	1185
Sine(B)	1.071452e-05	337
Sigmoid-Identity-Sine-Tanh(Q)	1.272606e-05	68097
ReLU-Sine(MM)	1.275749e-05	6785
Sine-Sigmoid(E)	1.352560e-05	66817
LogExp-Identity-Sine-Tanh(Q)	1.428260e-05	68097
ReLU-Identity-Sine-Tanh(Q)	1.542056e-05	68097
LogExp-Identity-Sine-ReLU(Q)	1.544548e-05	68097
LogExp-Sine(D)	1.628308e-05	34049
ReLU-Identity-Sine-Sigmoid(Q)	1.644714e-05	68097
LogExp-ReLU-Sine(T)	1.662348e-05	51073
ReLU-Identity-Sine(T)	1.666095e-05	51073
Identity-Sine-Tanh(T)	1.681932e-05	51073
Identity-Sine(D)	1.737286e-05	673
LogExp-Identity-Sine(T)	1.778383e-05	51073
LogExp-Identity-Sine-ReLU(Q)	1.780640e-05	17665
ReLU-Identity-Sine-Sigmoid(Q)	1.905118e-05	17665
Identity-Sine-Tanh(T)	1.925196e-05	3553
ReLU-Sine(S)	1.933193e-05	17025
Identity-Sine(D)	1.950528e-05	34049
Sigmoid-Identity-Sine-Tanh(Q)	1.988305e-05	17665
LogExp-Identity-Sine(T)	1.991400e-05	13249
ReLU-Sine(MM)	1.996472e-05	25857
LogExp-Identity-Sine-Tanh(Q)	2.001167e-05	17665
ReLU-Sine-Sigmoid-Tanh(Q)	2.017167e-05	68097
ReLU-Identity-Sine(T)	2.033012e-05	13249
Tanh-Sine(D)	2.042091e-05	34049
Tanh-Sine(D)	2.055280e-05	673
ReLU-Identity-Sine(R)	2.141294e-05	4417
LogExp-ReLU-Sine(T)	2.165569e-05	3553
Sigmoid-Identity-Sine-Tanh(Q)	2.176657e-05	151297
ReLU-Identity-Sine-Tanh(Q)	2.322451e-05	151297
ReLU-Sine(D)	2.332722e-05	34049
ReLU-Identity-Sine-Sigmoid(Q)	2.358708e-05	151297
ReLU-Identity-Sine-Tanh(Q)	2.358768e-05	4737
Sine-Identity(E)	2.403891e-05	1185
LogExp-ReLU-Sine(T)	2.414902e-05	13249
Identity-Sine-Tanh(R)	2.423174e-05	1185
LogExp-Sine(D)	2.424149e-05	8833
ReLU-Sine(MM)	2.439882e-05	1857
LogExp-Identity-Sine-ReLU(Q)	2.446927e-05	151297
LogExp-Sine(D)	2.465490e-05	673
ReLU-Sine(MM)	2.548502e-05	100865

Sine-Identity(E)	2.630055e-05	4417
Sine-Sigmoid(E)	2.632855e-05	17025
LogExp-Identity-Sine(T)	2.653514e-05	3553
ReLU-Sine-Sigmoid-Tanh(Q)	2.654410e-05	151297
Tanh-Sine(D)	2.716923e-05	2369
ReLU-Identity-Sine-Sigmoid(Q)	2.735593e-05	417
ReLU-Identity-Sine-Tanh(Q)	2.839464e-05	17665
LogExp-Sine(D)	2.865882e-05	2369
ReLU-Sine(E)	2.875951e-05	17025
LogExp-Identity-Sine-Tanh(Q)	2.925066e-05	4737
ReLU-Identity-Sine-Tanh-Sigmoid(R)	2.985358e-05	17025
Sigmoid-Identity-Sine-Tanh(Q)	3.029778e-05	4737
Identity-Sine-Tanh(T)	3.109996e-05	13249
ReLU-Identity-Sine(R)	3.113047e-05	1185
LogExp-Identity-Sine-Tanh(Q)	3.143679e-05	151297
LogExp-Identity-Sine-ReLU(Q)	3.221574e-05	4737
Sigmoid-Identity-Sine-Tanh(Q)	3.335646e-05	1345
ReLU-Identity-Sine(T)	3.385209e-05	3553
Sigmoid-Sine(D)	3.423116e-05	8833
ReLU-Sine(S)	3.498403e-05	66817
ReLU-Sine(E)	3.527506e-05	4417
Sigmoid-Sine(D)	3.548204e-05	2369
Identity-Sine-Tanh(T)	3.583437e-05	313
Sine-Sigmoid(E)	3.700374e-05	4417
LogExp-Identity-Sine(T)	3.718695e-05	1009
Identity-Sine(D)	3.737231e-05	8833
Identity-Sine(D)	3.809437e-05	2369
Tanh-Sine(D)	3.827695e-05	8833
Identity-Sine-Tanh(R)	3.958827e-05	4417
Sigmoid-Sine(D)	3.962899e-05	34049
ReLU-Sine(D)	3.998362e-05	8833
ReLU-Identity-Sine(R)	4.006778e-05	66817
LogExp-Identity-Sine-Tanh(Q)	4.080366e-05	1345
Sine-Identity(E)	4.142248e-05	17025
ReLU-Identity-Sine-Sigmoid(Q)	4.144473e-05	4737
Sine(B)	4.162520e-05	66817
Sigmoid-Sine(D)	4.306660e-05	133633
Sine-Identity(E)	4.352261e-05	66817
ReLU-Sine(D)	4.493540e-05	209
Sigmoid(B)	4.497494e-05	17025
LogExp-Identity-Sine-ReLU(Q)	4.510810e-05	1345
Identity-Sine(D)	4.513386e-05	133633

C.7 Max Dataset

Model Type	MSE Loss	Parameters
ReLU-Identity(MM)	0.000004	545
ReLU(B)	0.000004	105
ReLU-Identity-Sine(R)	0.000004	337
ReLU-Sine(D)	0.000005	673
ReLU-Sigmoid(MM)	0.000005	545
ReLU-Sine-Sigmoid-Tanh(Q)	0.000006	417
ReLU-Identity(D)	0.000006	2369
LogExp-ReLU(D)	0.000006	673
ReLU(B)	0.000006	1185
ReLU-Identity-Sine(T)	0.000006	109
ReLU-Identity-Tanh(T)	0.000006	1009
ReLU-Sigmoid(E)	0.000007	1185
ReLU-Identity-Sine-Tanh(Q)	0.000007	417
ReLU-Tanh(D)	0.000007	673
ReLU-Identity-Sine(T)	0.000007	313
ReLU-Sigmoid(D)	0.000008	2369
LogExp-ReLU-Identity(T)	0.000008	1009
ReLU-Tanh(MM)	0.000008	545
ReLU-Identity(MM)	0.000009	177
ReLU-Identity-Tanh(T)	0.000009	3553
LogExp-Identity-Sine-ReLU(Q)	0.000009	1345
LogExp-ReLU-Sine(T)	0.000009	1009
ReLU-Sigmoid(MM)	0.000010	1857
ReLU-Sigmoid-Tanh(T)	0.000010	1009
ReLU-Identity-Tanh(R)	0.000010	337
ReLU-Identity(MM)	0.000012	1857
ReLU-Tanh(MM)	0.000012	177
ReLU-Identity-Sine(T)	0.000012	13249
ReLU-Sine-Sigmoid-Tanh(Q)	0.000012	1345
ReLU-Sigmoid(D)	0.000013	209
ReLU-Identity(D)	0.000013	209
ReLU-Sigmoid(E)	0.000013	4417
ReLU-Sine(MM)	0.000013	545
ReLU-Sigmoid-Tanh(T)	0.000014	13249
ReLU-Identity-Tanh(R)	0.000014	1185
ReLU-Identity(D)	0.000014	34049
ReLU-Tanh(MM)	0.000014	1857
ReLU-Identity(D)	0.000014	8833
ReLU-Identity(MM)	0.000014	6785

ReLU-Identity-Sine(R)	0.000015	1185
ReLU-Identity-Sine(T)	0.000015	1009
ReLU-Sigmoid(D)	0.000015	8833
ReLU-Sigmoid(E)	0.000015	17025
ReLU-Identity(D)	0.000015	133633
ReLU-Identity(MM)	0.000015	100865
ReLU-Sigmoid-Tanh(T)	0.000015	3553
ReLU(B)	0.000016	66817
ReLU-Identity-Sine-Sigmoid(Q)	0.000016	417
ReLU-Tanh(MM)	0.000016	25857
ReLU-Identity-Sine-Sigmoid(Q)	0.000016	1345
ReLU-Tanh(D)	0.000016	8833
LogExp-ReLU-Identity(T)	0.000017	13249
ReLU(B)	0.000017	4417
ReLU-Sine(MM)	0.000017	1857
ReLU-Identity-Tanh(R)	0.000017	4417
ReLU-Sine(D)	0.000018	2369
ReLU-Identity-Tanh(T)	0.000018	13249
LogExp-Identity-Sine-ReLU(Q)	0.000018	417
LogExp-Identity-Sine-ReLU(Q)	0.000018	4737
ReLU-Tanh(D)	0.000018	2369
ReLU-Identity-Sine(T)	0.000018	3553
ReLU-Tanh(E)	0.000018	1185
ReLU-Sigmoid(MM)	0.000018	25857
ReLU-Sigmoid(MM)	0.000018	6785
ReLU-Sine(MM)	0.000019	177
ReLU-Sigmoid-Tanh(R)	0.000019	1185
ReLU-Sigmoid(MM)	0.000019	100865
LogExp-ReLU-Identity(T)	0.000019	3553
ReLU-Tanh(MM)	0.000019	6785
ReLU-Identity-Sine-Sigmoid(Q)	0.000019	4737
ReLU(B)	0.000019	17025
ReLU-Tanh(D)	0.000019	133633
ReLU-Tanh(D)	0.000019	34049
ReLU-Sigmoid-Tanh(T)	0.000020	51073
LogExp-ReLU-Sine(T)	0.000020	3553
LogExp-ReLU-Sine(T)	0.000020	313
LogExp-ReLU(D)	0.000020	209
LogExp-ReLU(D)	0.000020	8833
ReLU-Identity-Sine-Sigmoid(Q)	0.000020	151297
LogExp-ReLU-Identity(T)	0.000021	200449
ReLU-Identity-Sine-Tanh(Q)	0.000021	1345
ReLU-Identity-Sine(T)	0.000021	51073
ReLU-Sigmoid(D)	0.000022	133633

ReLU-Identity-Tanh(T)	0.000022	200449
ReLU-Identity(MM)	0.000022	25857
LogExp-ReLU(D)	0.000022	2369
ReLU-Tanh(MM)	0.000023	100865
ReLU-Identity-Sine(T)	0.000024	200449
ReLU-Sigmoid-Tanh(T)	0.000024	200449
ReLU-Identity-Tanh(R)	0.000024	66817
ReLU-Identity-Sine-Tanh(Q)	0.000026	68097
ReLU-Identity-Sine-Sigmoid(Q)	0.000026	17665
LogExp-ReLU(D)	0.000027	133633
ReLU-Sigmoid(MM)	0.000027	177
ReLU-Tanh(E)	0.000027	17025
LogExp-Identity-Sine-ReLU(Q)	0.000027	68097
LogExp-ReLU(D)	0.000027	34049
LogExp-ReLU-Sine(T)	0.000028	51073
ReLU-Sigmoid-Tanh(R)	0.000028	17025
ReLU-Identity-Sine-Tanh(Q)	0.000028	4737

C.8 Multiplication Dataset

Model Type	MSE Loss	Parameters
Sigmoid(B)	0.009232	17025
Tanh-Sigmoid(S)	0.009527	66817
ReLU-Identity-Sine(T)	0.009674	51073
LogExp-ReLU-Sine(T)	0.010022	3553
Tanh-Sigmoid(E)	0.010188	4417
ReLU-Sine-Sigmoid-Tanh(Q)	0.010388	68097
LogExp-Identity-Sine-ReLU(Q)	0.010442	17665
ReLU-Identity-Sine-Sigmoid(Q)	0.010580	68097
Sigmoid-Identity-Sine-Tanh(Q)	0.010727	4737
ReLU-Identity-Sine-Tanh(Q)	0.011583	17665
Identity-Sine-Tanh(T)	0.011622	13249
LogExp-Identity-Sine(T)	0.011690	3553
ReLU-Identity-Sine-Sigmoid(Q)	0.011729	17665
ReLU-Sine-Sigmoid-Tanh(Q)	0.011754	17665
Log-Exp(B)	0.012004	105
Sigmoid-Identity-Sine-Tanh(Q)	0.012036	17665
LogExp-Sine(D)	0.012048	2369
LogExp-ReLU-Sine(T)	0.012180	1009
ReLU-Sine(D)	0.012249	34049
Identity-Sine-Tanh(T)	0.012416	1009

ReLU-Identity-Sine-Tanh(Q)	0.013035	68097
Sigmoid(B)	0.013061	66817
Tanh-Sigmoid(E)	0.013066	17025
Tanh(B)	0.013601	4417
LogExp-ReLU-Sine(T)	0.013605	313
ReLU-Identity-Sine(T)	0.014206	13249
LogExp-Sine(D)	0.014635	34049
Sigmoid-ReLU(S)	0.014783	264705
Tanh-Sigmoid(D)	0.014878	34049
Sigmoid-Tanh(S)	0.014901	17025
Sigmoid-Tanh(S)	0.015261	4417
LogExp-ReLU-Identity(T)	0.016044	13249
Sigmoid-Identity-Sine-Tanh(Q)	0.016254	68097
Sigmoid-Identity-Sine-Tanh(Q)	0.016326	1345
ReLU-Sine(D)	0.016470	8833
Sigmoid(B)	0.016499	264705
Tanh-Sigmoid(E)	0.016843	1185
Identity-Sine-Tanh(R)	0.017251	1185
ReLU-Identity-Sine-Tanh(Q)	0.017462	151297
Identity-Sine-Tanh(R)	0.017507	4417
ReLU-Sigmoid(E)	0.017702	264705
LogExp-Identity-Sine(T)	0.018196	313
ReLU-Sine-Sigmoid-Tanh(Q)	0.018232	151297
LogExp-Sine(D)	0.018238	209
LogExp-Identity-Sine-Tanh(Q)	0.019096	1345
Tanh-Sigmoid(D)	0.019820	673
ReLU-Identity-Sine-Tanh(Q)	0.020487	4737
Tanh-Sigmoid(S)	0.020639	4417
Sine-Sigmoid(E)	0.020798	66817
ReLU-Sine(E)	0.020906	66817
ReLU-Sine(E)	0.021086	17025
ReLU-Sigmoid(E)	0.021174	66817
ReLU-Identity-Sine-Sigmoid(Q)	0.021485	151297
Sine-Sigmoid(E)	0.021504	17025
ReLU-Identity-Sine-Tanh-Sigmoid(R)	0.021623	264705
Sigmoid-Tanh(S)	0.021760	66817
LogExp-Sine(D)	0.022663	673
LogExp-Identity-Sine-Tanh(Q)	0.022987	4737
ReLU-Sine(D)	0.023165	133633
Tanh(B)	0.023500	17025
Sigmoid-Tanh(S)	0.023660	264705
ReLU-Identity-Sine(T)	0.023871	3553
ReLU-Identity-Sine(T)	0.023968	200449
Sine-ReLU(S)	0.023974	4417

Identity-Sine-Tanh(T)	0.023975	51073
Sigmoid-ReLU(S)	0.024132	66817
ReLU-Sine-Sigmoid-Tanh(Q)	0.024151	4737
ReLU-Identity-Sine-Tanh-Sigmoid(R)	0.024216	66817
LogExp-Identity-Sine-ReLU(Q)	0.024422	68097
ReLU-Identity-Sine-Sigmoid(Q)	0.024639	4737
LogExp-ReLU-Identity(T)	0.025441	3553
Identity-Sine-Tanh(R)	0.025618	17025
ReLU-Identity-Sine(R)	0.026219	4417
LogExp-Identity-Sine-Tanh(Q)	0.026226	17665
LogExp-Identity-Sine-Tanh(Q)	0.026621	145
Tanh-Sigmoid(E)	0.026744	66817
LogExp-ReLU(D)	0.027879	8833
ReLU-Sigmoid(D)	0.027915	133633
Sine-Sigmoid(E)	0.028452	1185
Sine-Sigmoid(E)	0.028592	4417
LogExp-Identity-Sine-ReLU(Q)	0.029275	1345
ReLU-Identity-Sine(R)	0.029568	17025
ReLU-Sine(D)	0.029835	2369
Sine-ReLU(S)	0.029902	17025
LogExp-ReLU-Sine(T)	0.030419	51073
LogExp-Identity-Sine(T)	0.030505	1009
Log-Exp(B)	0.030554	337
ReLU-Sine(E)	0.030802	4417
ReLU-Identity-Sine(R)	0.030874	66817
Sigmoid-Tanh(S)	0.030984	1185
Tanh(B)	0.031610	1185
Sigmoid-Identity-Sine-Tanh(Q)	0.031878	151297
ReLU-Sine(E)	0.032054	264705
ReLU-Identity-Sine-Tanh-Sigmoid(R)	0.032175	17025
ReLU-Sigmoid-Tanh(R)	0.032465	264705
Tanh-ReLU(S)	0.032775	17025
ReLU-Sigmoid(D)	0.033011	34049
Sigmoid(B)	0.033318	1185
ReLU-Sigmoid-Tanh(T)	0.033347	51073
ReLU-Identity-Tanh(T)	0.034015	51073

C.9 $sin(x_1) \cdot sin(x_2)$ Dataset

Model Type	MSE Loss	Parameters
Sigmoid-Identity-ReLU-Sine(DQ)	1.002394e-04	25473

LogExp-Sine-ReLU-Identity(DQ)	1.102212e-04	100097
Sine(B)	1.150198e-04	12737
LogExp-Sine-ReLU-Identity(DQ)	1.254939e-04	25473
Sine(B)	1.328861e-04	50049
Tanh-Identity-ReLU-Sine(DQ)	1.372053e-04	6593
LogExp-Sine-ReLU-Identity(DQ)	1.528628e-04	6593
Sigmoid-Identity-ReLU-Sine(DQ)	1.551288e-04	6593
Sine(B)	1.746948e-04	249
LogExp-Sine-ReLU-Identity(DQ)	1.802996e-04	1761
Sigmoid(B)	2.019531e-04	50049
Sigmoid-Identity-ReLU-Sine(DQ)	2.062674e-04	1761
Tanh-Identity-ReLU-Sine(DQ)	2.253995e-04	1761
Sigmoid(B)	2.801080e-04	198401
Sine(B)	2.846046e-04	198401
LogExp-Sine-ReLU-Identity(DQ)	3.681662e-04	497
Sigmoid(B)	4.443539e-04	790017
LogExp-Sine-ReLU-Identity(DQ)	5.578428e-04	396801
Tanh(B)	6.089346e-04	3297
ReLU(B)	6.449642e-04	50049
ReLU(B)	6.787748e-04	198401
Tanh(B)	7.637935e-04	881
Tanh(B)	7.936276e-04	12737
ReLU(B)	9.025322e-04	790017
Tanh-Identity-ReLU-Sine(DQ)	9.271109e-04	497
ReLU(B)	9.655557e-04	12737
Sigmoid-Identity-ReLU-Sine(DQ)	9.821363e-04	497
Tanh(B)	1.094252e-03	50049
ReLU(B)	1.153568e-03	3297
Sine(B)	1.234811e-03	790017
Tanh(B)	1.707253e-03	198401
Sigmoid(B)	2.711949e-03	12737
ReLU(B)	3.245658e-03	881
Sine(B)	8.108883e-03	77
Tanh(B)	1.012798e-02	790017
Tanh-Identity-ReLU-Sine(DQ)	2.274837e-02	153
Sigmoid-Identity-ReLU-Sine(DQ)	3.317213e-02	153
ReLU(B)	4.442213e-02	249
Tanh(B)	4.829999e-02	249
LogExp-Sine-ReLU-Identity(DQ)	5.979945e-02	153
Sigmoid(B)	8.100235e-02	3297
Sigmoid(B)	1.093509e-01	881
Tanh(B)	1.246651e-01	77
ReLU(B)	1.861619e-01	77
Sigmoid(B)	1.910648e-01	249

Sigmoid(B)	2.148804e-01	77
Identity(B)	2.360977e-01	12737
Identity(B)	2.361340e-01	881
Identity(B)	2.361583e-01	198401
Identity(B)	2.361595e-01	3297
Identity(B)	2.361682e-01	249
Identity(B)	2.361684e-01	77
Identity(B)	2.363096e-01	50049
Identity(B)	2.364899e-01	790017
LogExp(B)	6.406162e-01	50049
LogExp(B)	1.026950e+00	3297
LogExp(B)	1.154908e+02	198401
LogExp(B)	2.610127e+13	881
LogExp(B)	3.998643e+18	12737

C.10 $sin(x_1) + cos(x_2)$ Dataset

Model Type	MSE Loss	Parameters
Sigmoid-Identity-ReLU-Sine(DQ)	0.000086	25473
Tanh-Identity-ReLU-Sine(DQ)	0.000098	396801
Sigmoid-Identity-ReLU-Sine(DQ)	0.000100	396801
Sine(B)	0.000125	3297
LogExp-Sine-ReLU-Identity(DQ)	0.000133	6593
Sine(B)	0.000139	881
Tanh-Identity-ReLU-Sine(DQ)	0.000153	1761
Sigmoid-Identity-ReLU-Sine(DQ)	0.000154	1761
LogExp-Sine-ReLU-Identity(DQ)	0.000154	1761
LogExp-Sine-ReLU-Identity(DQ)	0.000163	396801
Sine(B)	0.000167	12737
Sine(B)	0.000172	50049
Sine(B)	0.000235	249
Tanh-Identity-ReLU-Sine(DQ)	0.000353	497
Sine(B)	0.000483	198401
Sigmoid-Identity-ReLU-Sine(DQ)	0.000587	497
Sigmoid(B)	0.000588	790017
Sigmoid(B)	0.000599	50049
Sigmoid-Identity-ReLU-Sine(DQ)	0.000624	153
Sigmoid(B)	0.000688	198401
Sine(B)	0.001016	77
Tanh-Identity-ReLU-Sine(DQ)	0.001165	153
ReLU(B)	0.001506	198401

ReLU(B)	0.001908	790017
ReLU(B)	0.001982	12737
Tanh(B)	0.002100	3297
ReLU(B)	0.002120	50049
Tanh(B)	0.002162	881
Tanh(B)	0.002263	12737
ReLU(B)	0.002568	3297
Tanh(B)	0.003113	50049
ReLU(B)	0.004998	881
LogExp-Sine-ReLU-Identity(DQ)	0.005412	497
Tanh(B)	0.005704	198401
Sine(B)	0.005767	790017
Sigmoid(B)	0.009394	12737
Sigmoid(B)	0.010771	3297
Tanh(B)	0.012822	249
ReLU(B)	0.013824	249
Tanh(B)	0.019589	790017
LogExp-Sine-ReLU-Identity(DQ)	0.023513	153
Sigmoid(B)	0.129553	881
Tanh(B)	0.337096	77
Sigmoid(B)	0.678403	249
ReLU(B)	0.688930	77
Sigmoid(B)	0.825396	77
Identity(B)	0.938015	77
Identity(B)	0.938529	249
Identity(B)	0.938794	12737
Identity(B)	0.939042	881
Identity(B)	0.939228	50049
Identity(B)	0.939312	790017
Identity(B)	0.939348	3297
Identity(B)	0.940748	198401

C.11 $max(sin(x_1), sin(x_2))$ Dataset

Model Type	MSE Loss	Parameters
Sigmoid-Identity-ReLU-Sine(DQ)	3.551311e-04	25473
Sine(B)	3.886810e-04	881
Tanh-Identity-ReLU-Sine(DQ)	4.001738e-04	25473
Sigmoid-Identity-ReLU-Sine(DQ)	4.719596e-04	6593
Sine(B)	5.259567e-04	12737
Tanh-Identity-ReLU-Sine(DQ)	5.494741e-04	1761

Tanh(B)	6.101574e-04	198401
Sigmoid-Identity-ReLU-Sine(DQ)	6.544009e-04	1761
Tanh-Identity-ReLU-Sine(DQ)	7.187343e-04	6593
Tanh(B)	9.072367e-04	790017
Tanh(B)	9.674739e-04	50049
ReLU(B)	9.720830e-04	790017
ReLU(B)	9.743124e-04	198401
LogExp-Sine-ReLU-Identity(DQ)	9.809652e-04	1761
ReLU(B)	1.121718e-03	50049
LogExp-Sine-ReLU-Identity(DQ)	1.369716e-03	6593
Tanh-Identity-ReLU-Sine(DQ)	1.493707e-03	497
Sigmoid-Identity-ReLU-Sine(DQ)	1.498983e-03	497
Sine(B)	1.570057e-03	3297
ReLU(B)	2.065045e-03	12737
LogExp-Sine-ReLU-Identity(DQ)	2.086330e-03	497
Sine(B)	2.098511e-03	249
Sigmoid(B)	2.697506e-03	12737
Sigmoid(B)	2.699803e-03	3297
Sigmoid(B)	3.900797e-03	881
Sigmoid(B)	4.873247e-03	50049
LogExp-Sine-ReLU-Identity(DQ)	5.102780e-03	25473
Tanh-Identity-ReLU-Sine(DQ)	6.108182e-03	153
Sigmoid-Identity-ReLU-Sine(DQ)	6.291096e-03	153
Sine(B)	8.765030e-03	77
ReLU(B)	9.435077e-03	3297
Sigmoid(B)	9.441121e-03	790017
Tanh(B)	1.224971e-02	881
ReLU(B)	1.902999e-02	881
Tanh(B)	2.268122e-02	3297
Tanh(B)	2.480021e-02	12737
LogExp-Sine-ReLU-Identity(DQ)	2.912080e-02	153
Tanh(B)	3.403129e-02	249
Sigmoid(B)	4.288222e-02	249
Sigmoid(B)	5.128191e-02	198401
ReLU(B)	5.980527e-02	249
Tanh(B)	1.265159e-01	77
Sigmoid(B)	2.060013e-01	77
ReLU(B)	2.390016e-01	77
Identity(B)	3.048867e-01	77
Identity(B)	3.050236e-01	881
Identity(B)	3.050546e-01	249
Identity(B)	3.051976e-01	12737
Identity(B)	3.052031e-01	3297
Identity(B)	3.052634e-01	198401

Identity(B)	3.052870e-01	50049
Identity(B)	3.055089e-01	790017
LogExp-Sine-ReLU-Identity(DQ)	3.378067e+29	100097

C.12 Poly-v1 Dataset

Model Type	MSE Loss	Parameters
Tanh-Identity-ReLU-Sine(DQ)	1.110121	396801
Tanh-Identity-ReLU-Sine(DQ)	1.141925	1761
Sigmoid-Identity-ReLU-Sine(DQ)	2.380271	1761
ReLU(B)	2.699300	50049
ReLU(B)	3.155248	12737
Sine(B)	4.954150	790017
ReLU(B)	5.399961	198401
ReLU(B)	7.138938	3297
ReLU(B)	11.655133	790017
Tanh-Identity-ReLU-Sine(DQ)	12.152509	497
Sigmoid-Identity-ReLU-Sine(DQ)	43.278779	497
Tanh(B)	43.526353	790017
ReLU(B)	44.056655	881
Sigmoid(B)	45.695821	790017
LogExp-Sine-ReLU-Identity(DQ)	48.443759	497
LogExp-Sine-ReLU-Identity(DQ)	49.499772	1761
LogExp-Sine-ReLU-Identity(DQ)	62.492322	153
LogExp-Sine-ReLU-Identity(DQ)	64.565363	6593
Sigmoid-Identity-ReLU-Sine(DQ)	65.272677	153
Sigmoid(B)	68.127695	881
ReLU(B)	71.925055	249
Sigmoid(B)	72.077288	3297
Tanh-Identity-ReLU-Sine(DQ)	83.545288	153
Sigmoid(B)	84.062144	12737
LogExp-Sine-ReLU-Identity(DQ)	177.789175	25473
Sigmoid(B)	182.461387	50049
LogExp(B)	234.897464	50049
Sine(B)	252.130892	198401
Sine(B)	275.711224	881
Sine(B)	341.824558	249
LogExp(B)	348.711442	12737
LogExp(B)	374.233008	790017
Sine(B)	421.341707	77
LogExp-Sine-ReLU-Identity(DQ)	612.339858	100097

Sine(B)	686.336498	3297
Tanh(B)	1525.450182	198401
Tanh(B)	2378.287794	3297
Sine(B)	6038.778704	12737
LogExp(B)	13321.988124	3297
Tanh(B)	20799.141507	249
Tanh(B)	20839.607328	77
Sine(B)	23903.131978	50049
Identity(B)	26468.443276	77
Identity(B)	26471.557520	249
Identity(B)	26475.241571	3297
Identity(B)	26477.589237	881
Identity(B)	26483.199508	12737
Identity(B)	26494.772761	198401
Identity(B)	26499.633797	50049
Identity(B)	26546.770696	790017
Tanh(B)	34689.751730	50049
ReLU(B)	52427.050539	77
Tanh(B)	61718.953449	881
Tanh(B)	83844.690144	12737
Sigmoid(B)	184318.890588	77
Sigmoid(B)	204698.548770	249
Sigmoid(B)	204722.445421	198401

C.13 Poly-v2 Dataset

Model Type	MSE Loss	Parameters
ReLU(B)	2.464118e-03	50049
Sigmoid-Identity-ReLU-Sine(DQ)	2.492162e-03	497
Tanh-Identity-ReLU-Sine(DQ)	2.529449e-03	497
Sigmoid-Identity-ReLU-Sine(DQ)	2.783454e-03	153
Tanh-Identity-ReLU-Sine(DQ)	3.082667e-03	153
Tanh-Identity-ReLU-Sine(DQ)	3.190937e-03	1761
Sigmoid(B)	3.288851e-03	3297
Sigmoid-Identity-ReLU-Sine(DQ)	3.627595e-03	1761
LogExp-Sine-ReLU-Identity(DQ)	3.639907e-03	497
ReLU(B)	3.854643e-03	790017
Tanh-Identity-ReLU-Sine(DQ)	4.110414e-03	6593
Sigmoid(B)	4.778981e-03	12737
Sigmoid(B)	4.869028e-03	881
Sigmoid-Identity-ReLU-Sine(DQ)	5.215905e-03	6593

LogExp-Sine-ReLU-Identity(DQ)	6.816202e-03	6593
Sigmoid(B)	7.744870e-03	50049
LogExp-Sine-ReLU-Identity(DQ)	7.781561e-03	1761
ReLU(B)	8.299285e-03	12737
LogExp-Sine-ReLU-Identity(DQ)	8.440077e-03	25473
LogExp-Sine-ReLU-Identity(DQ)	9.341818e-03	153
Tanh(B)	1.103039e-02	790017
Sine(B)	1.141513e-02	198401
Sine(B)	1.141605e-02	249
Sine(B)	1.206936e-02	790017
ReLU(B)	1.408698e-02	881
Sine(B)	1.524963e-02	881
ReLU(B)	1.649235e-02	3297
Tanh(B)	1.650499e-02	198401
Tanh(B)	1.692497e-02	249
Tanh(B)	1.748832e-02	881
Sigmoid(B)	2.000268e-02	198401
Sine(B)	2.250853e-02	3297
Tanh(B)	2.391386e-02	3297
Sine(B)	3.206739e-02	12737
Sigmoid(B)	3.240024e-02	790017
Tanh(B)	3.435221e-02	12737
Sine(B)	4.124608e-02	50049
LogExp-Sine-ReLU-Identity(DQ)	4.175243e-02	100097
Sigmoid(B)	4.855275e-02	249
ReLU(B)	5.218624e-02	249
LogExp-Sine-ReLU-Identity(DQ)	5.923935e-02	396801
LogExp(B)	8.271052e-02	3297
LogExp(B)	9.080861e-02	12737
Sine(B)	1.072994e-01	77
Tanh(B)	1.145237e-01	50049
Tanh(B)	1.576821e-01	77
LogExp(B)	2.724432e+00	50049
Sigmoid(B)	3.757239e+00	77
LogExp(B)	3.925604e+00	198401
Identity(B)	6.602226e+01	77
Identity(B)	6.605630e+01	12737
Identity(B)	6.607456e+01	249
Identity(B)	6.608727e+01	790017
Identity(B)	6.609774e+01	50049
Identity(B)	6.609914e+01	3297
Identity(B)	6.611546e+01	198401
Identity(B)	6.616614e+01	881
ReLU(B)	7.390096e+01	77

LogExp(B)	8.659283e+01	790017
LogExp(B)	1.698540e+08	249
LogExp(B)	1.778329e+09	881