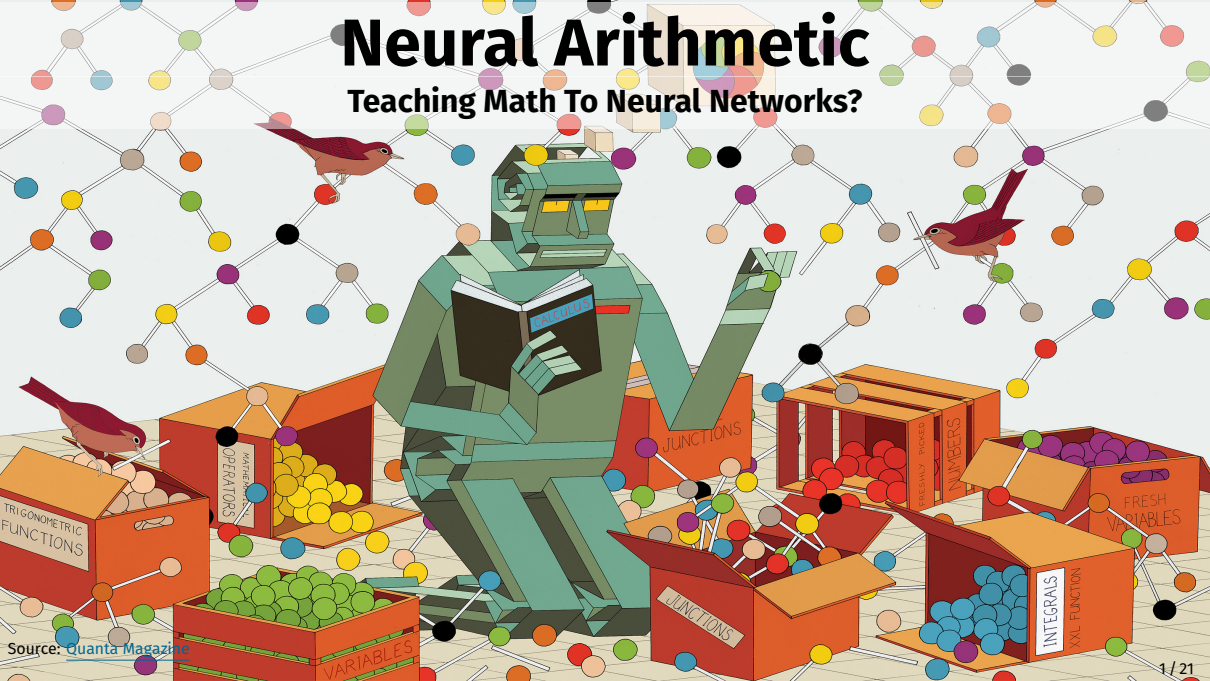
Neural Arithmetic

Teaching Math To Neural Networks?

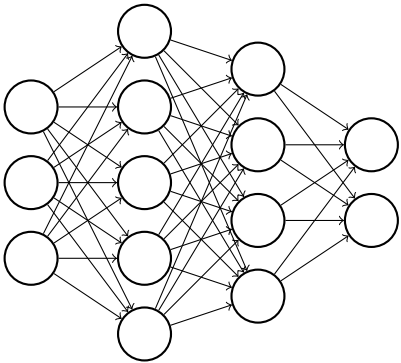


Source: Quanta Magazine

Neural Arithmetic

Extrapolating Beyond The Training Range
& Building **Transparent** Models

Neural Networks

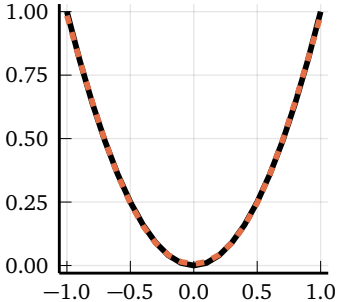


Composed of **Dense** layers:

$$\boldsymbol{y} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

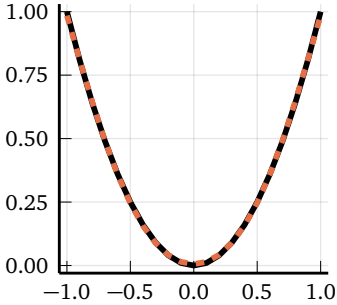
(No convolutions in this talk.)

Neural Networks

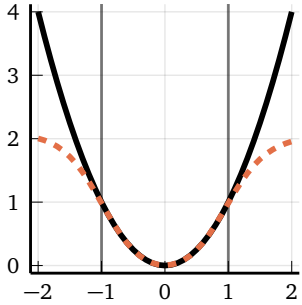


Universal **Approximators**

Neural Networks

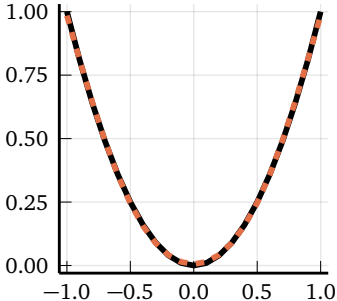


Universal **Approximators**

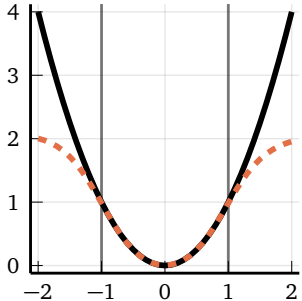


Poor **Extrapolators**

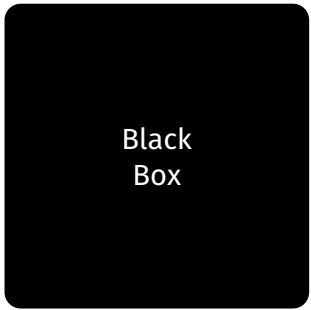
Neural Networks



Universal **Approximators**



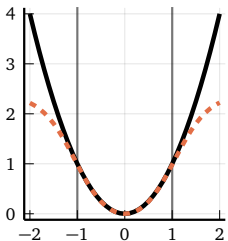
Poor **Extrapolators**



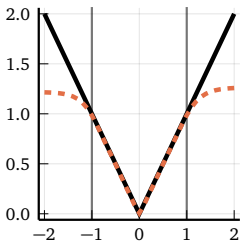
Intransparent

Problems With Extrapolation

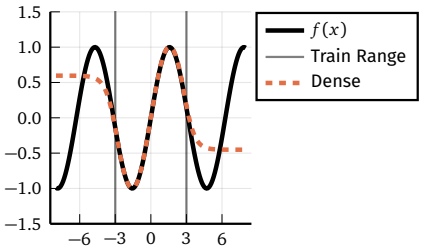
$f(x) = x^2$



$f(x) = |x|$

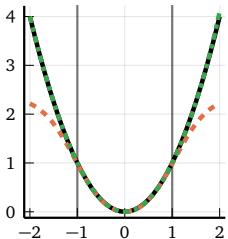


$f(x) = \sin(x)$

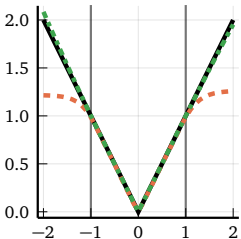


Problems With Extrapolation

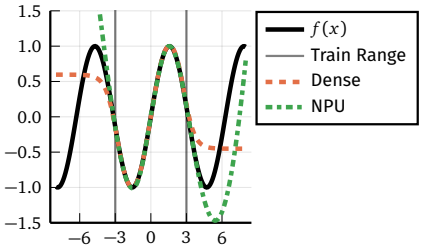
$f(x) = x^2$



$f(x) = |x|$



$f(x) = \sin(x)$



Inductive Bias

1. Assume **composition** of arithmetic operations.
2. Construct layers with **inductive bias** towards arithmetic operations.

Addition of inputs

$$\boldsymbol{x}^\intercal \boldsymbol{w} = x_1 w_1 + x_2 w_2 + \dots$$

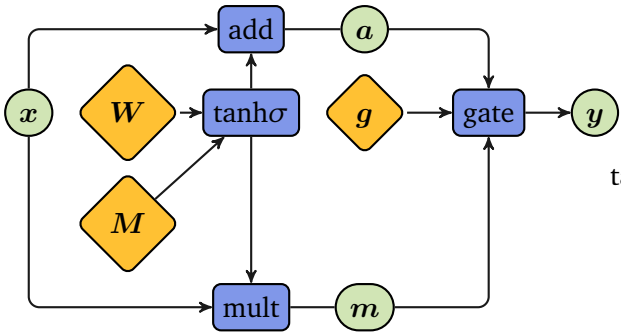
is simple.

Multiplication/power functions

$$\prod_i x_i^{w_i} = \exp(\boldsymbol{w}^\intercal \log \boldsymbol{x})$$

a bit more tricky due to complex outputs...

Neural Arithmetic Logic Unit (NALU)



$$\text{add}(\boldsymbol{x}, \hat{\boldsymbol{W}}) = \hat{\boldsymbol{W}}\boldsymbol{x}$$

$$\text{mult}(\boldsymbol{x}, \hat{\boldsymbol{W}}) = \exp(\hat{\boldsymbol{W}} \log|\boldsymbol{x}| + \epsilon)$$

$$\text{gate}(\boldsymbol{a}, \boldsymbol{m}, \boldsymbol{g}) = \boldsymbol{g} \odot \boldsymbol{a} + (1 - \boldsymbol{g}) \odot \boldsymbol{m}$$

$$\tanh\sigma(\boldsymbol{W}, \boldsymbol{M}) = \tanh(\boldsymbol{W}) \odot \sigma(\boldsymbol{M})$$

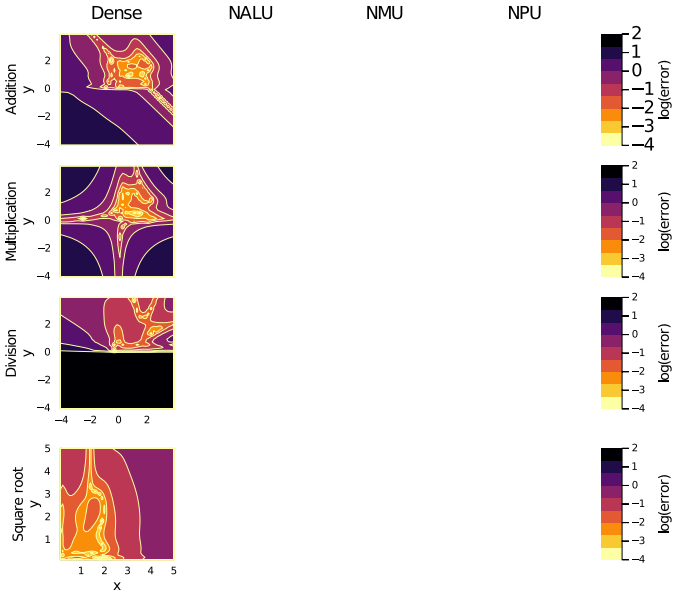
where

$$\boldsymbol{g} = \sigma(\boldsymbol{G}\boldsymbol{x})$$

(*Neural Arithmetic Logic Units* - Trask et al. 2018)

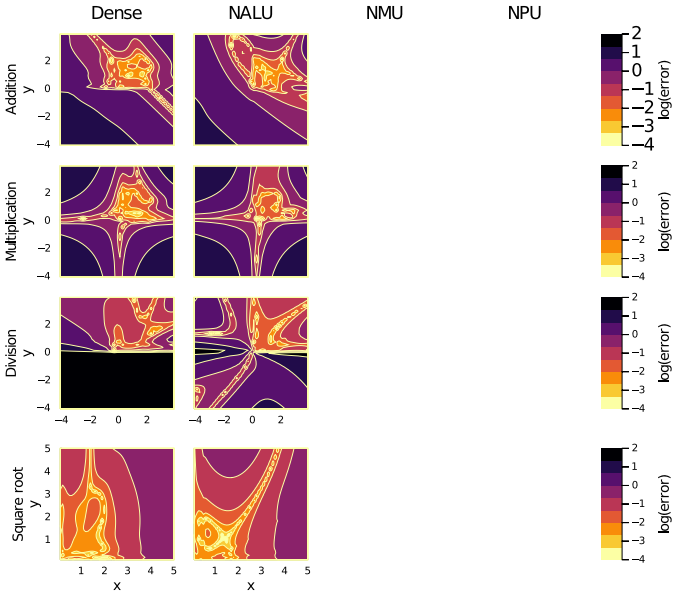
Task: Learn f in *one* model

$$f(x, y) = (x + y, xy, \frac{x}{y}, \sqrt{y})^\intercal$$

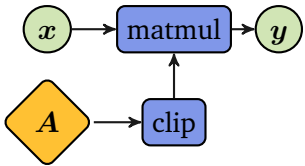


Task: Learn f in *one* model

$$f(x, y) = (x + y, xy, \frac{x}{y}, \sqrt{y})^\intercal$$

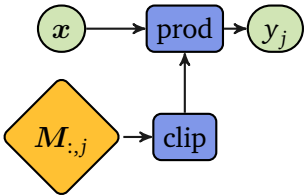


Simpler Arithmetic Units



Neural Addition Unit (NAU)

$$y = \text{clip}(A) \cdot x$$
$$\text{clip}(M) = \min(\max(M, -1), 1)$$



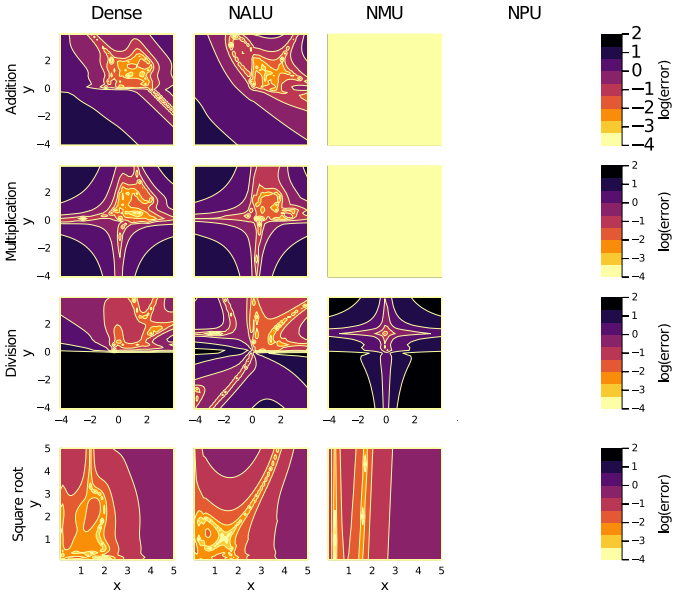
Neural Multiplication Unit (NMU)

$$\text{prod}(x, \hat{M}_{:,j}) = \prod_i \hat{M}_{ij} x_i - 1 + \hat{M}_{ij}$$
$$\text{clip}(M) = \min(\max(M, 0), 1)$$

(*Neural Arithmetic Units* - Madsen & Johansen 2020)

Task: Learn f in *one* model

$$f(x, y) = (x + y, xy, \frac{x}{y}, \sqrt{y})^\intercal$$



Complex space

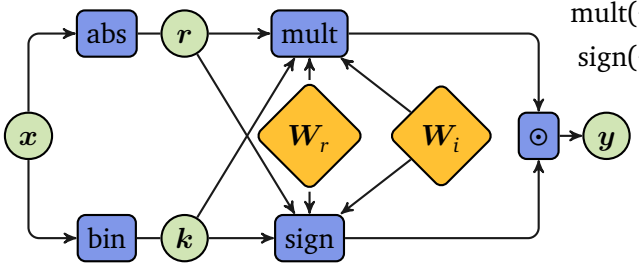
We extend NALU by isolating its multiplication path

$$\boldsymbol{m} = \exp(\boldsymbol{W} \log |\boldsymbol{x}|)$$

and lifting it into complex space ($\log_{\mathbb{C}}$ is the complex logarithm)

$$\boldsymbol{z} = \text{real}(\exp(\boldsymbol{W}_{\mathbb{C}} \log_{\mathbb{C}} \boldsymbol{x})) = \text{real}(\exp\left((\boldsymbol{W}_r + i\boldsymbol{W}_i) \log_{\mathbb{C}} \boldsymbol{x}\right)).$$

Naive Neural Power Unit

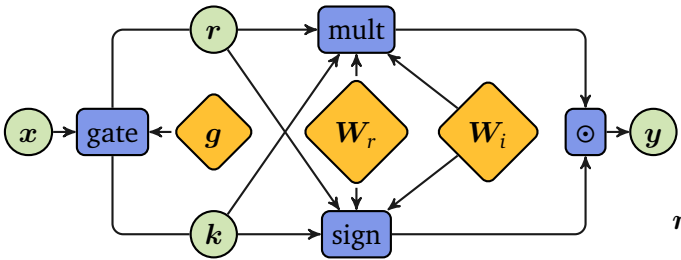


$$\text{mult}(\boldsymbol{r}, \boldsymbol{k}, \boldsymbol{W}_r, \boldsymbol{W}_i) = \exp(\boldsymbol{W}_r \log \boldsymbol{r} - \pi \boldsymbol{W}_i \boldsymbol{k})$$
$$\text{sign}(\boldsymbol{r}, \boldsymbol{k}, \boldsymbol{W}_r, \boldsymbol{W}_i) = \cos(\boldsymbol{W}_i \log \boldsymbol{r} + \pi \boldsymbol{W}_r \boldsymbol{k})$$

$$\text{abs}(\boldsymbol{x}) = \boldsymbol{r} = |\boldsymbol{x}| + \epsilon$$

$$\text{bin}(x_i) = k_i = \begin{cases} 0 & x_i \geq 0 \\ 1 & x_i < 0 \end{cases}$$

Neural Power Unit



$$\text{mult}(\boldsymbol{r}, \boldsymbol{k}, \boldsymbol{W}_r, \boldsymbol{W}_i) = \exp(\boldsymbol{W}_r \log \boldsymbol{r} - \pi \boldsymbol{W}_i \boldsymbol{k})$$

$$\text{sign}(\boldsymbol{r}, \boldsymbol{k}, \boldsymbol{W}_r, \boldsymbol{W}_i) = \cos(\boldsymbol{W}_i \log \boldsymbol{r} + \pi \boldsymbol{W}_r \boldsymbol{k})$$

$$\text{gate}(\boldsymbol{x}, \boldsymbol{g}) = (\boldsymbol{r}, \boldsymbol{k})$$

where

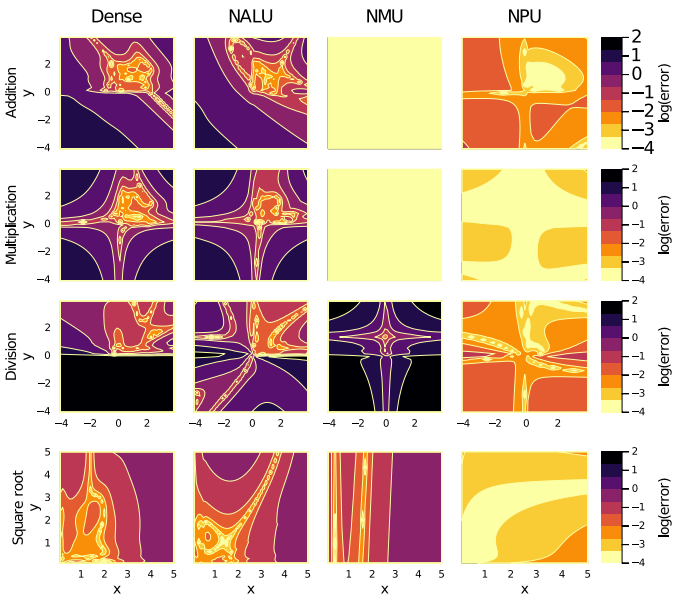
$$\boldsymbol{r} = \hat{\boldsymbol{g}} \odot (|\boldsymbol{x}| + \epsilon) + (1 - \hat{\boldsymbol{g}}),$$

$$k_i = \begin{cases} 0 & x_i \geq 0 \\ \hat{g}_i & x_i < 0 \end{cases},$$

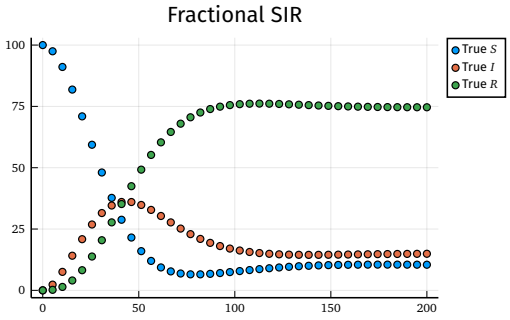
$$\hat{\boldsymbol{g}} = \min(\max(\boldsymbol{g}, 0), 1),$$

Task: Learn f in *one* model

$$f(x, y) = (x + y, xy, \frac{x}{y}, \sqrt{y})^\intercal$$



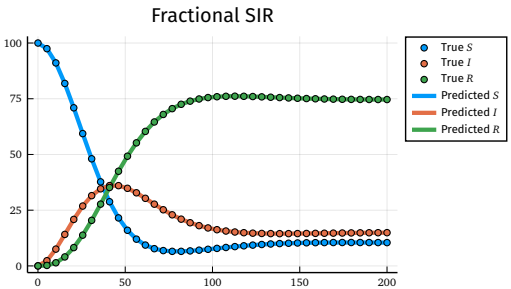
Fractional SIR

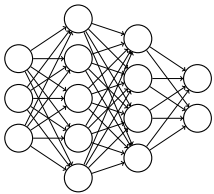


Differential Equation:

$$\begin{bmatrix} \dot{S} \\ \dot{I} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} -\beta & 0 & \eta \\ \beta & -\alpha & 0 \\ 0 & \alpha & -\eta \end{bmatrix} \begin{bmatrix} I^{\gamma} S^{\kappa} \\ I \\ R \end{bmatrix}$$

Black Box Models





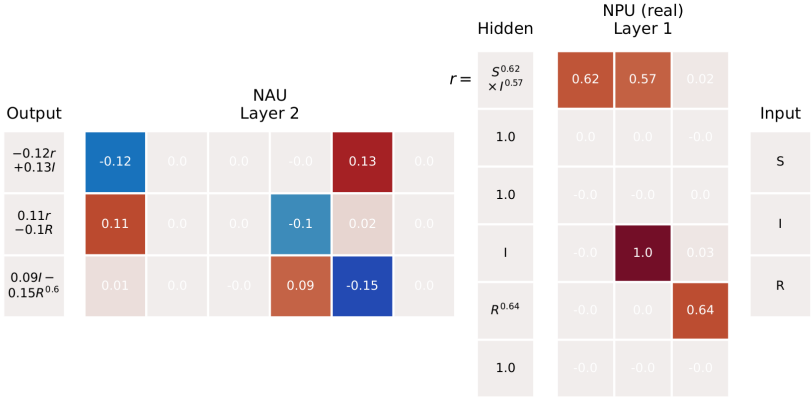
Differential Equation:

$$\begin{bmatrix} \dot{S} \\ \dot{I} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} -\beta & 0 & \eta \\ \beta & -\alpha & 0 \\ 0 & \alpha & -\eta \end{bmatrix} \begin{bmatrix} I^{\gamma} S^{\kappa} \\ I \\ R \end{bmatrix}$$

Neural Differential Equation:

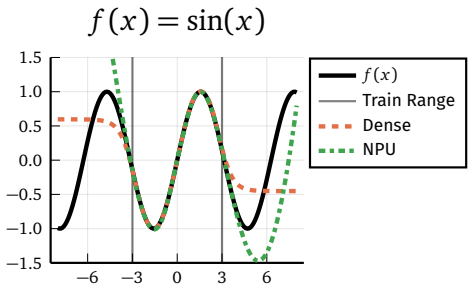
$$\begin{bmatrix} \dot{S} \\ \dot{I} \\ \dot{R} \end{bmatrix} = \text{NeuralODE}\left(\begin{bmatrix} S \\ I \\ R \end{bmatrix}, \boldsymbol{\theta} \right)$$

Fractional SIR

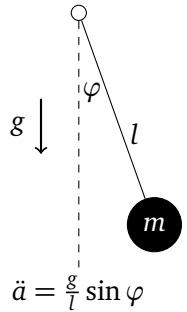


$$\begin{bmatrix} \dot{S} \\ \dot{I} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} -\beta & 0 & \eta \\ \beta & -\alpha & 0 \\ 0 & \alpha & -\eta \end{bmatrix} \begin{bmatrix} I^{\gamma} S^{\kappa} \\ I \\ R \end{bmatrix}$$

Future Work

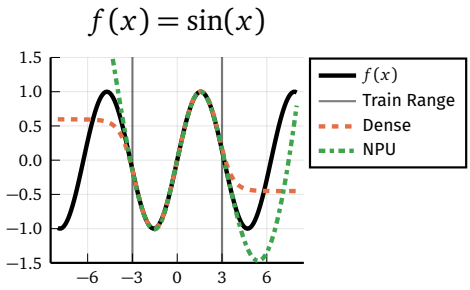


Trigonometric functions

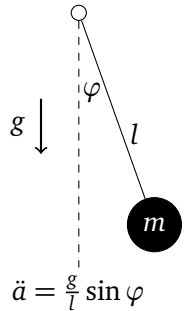


Some serious equation discovery

Future Work



Trigonometric functions



Some serious equation discovery

Do you have an application?

Neural Arithmetic

Extrapolating Beyond The Training Range
& Building **Transparent** Models

Niklas Heim

Tomáš Pevný

Václav Šmídl

https://github.com/nmheim/NeuralArithmetic.jl
heimnikl@fel.cvut.cz