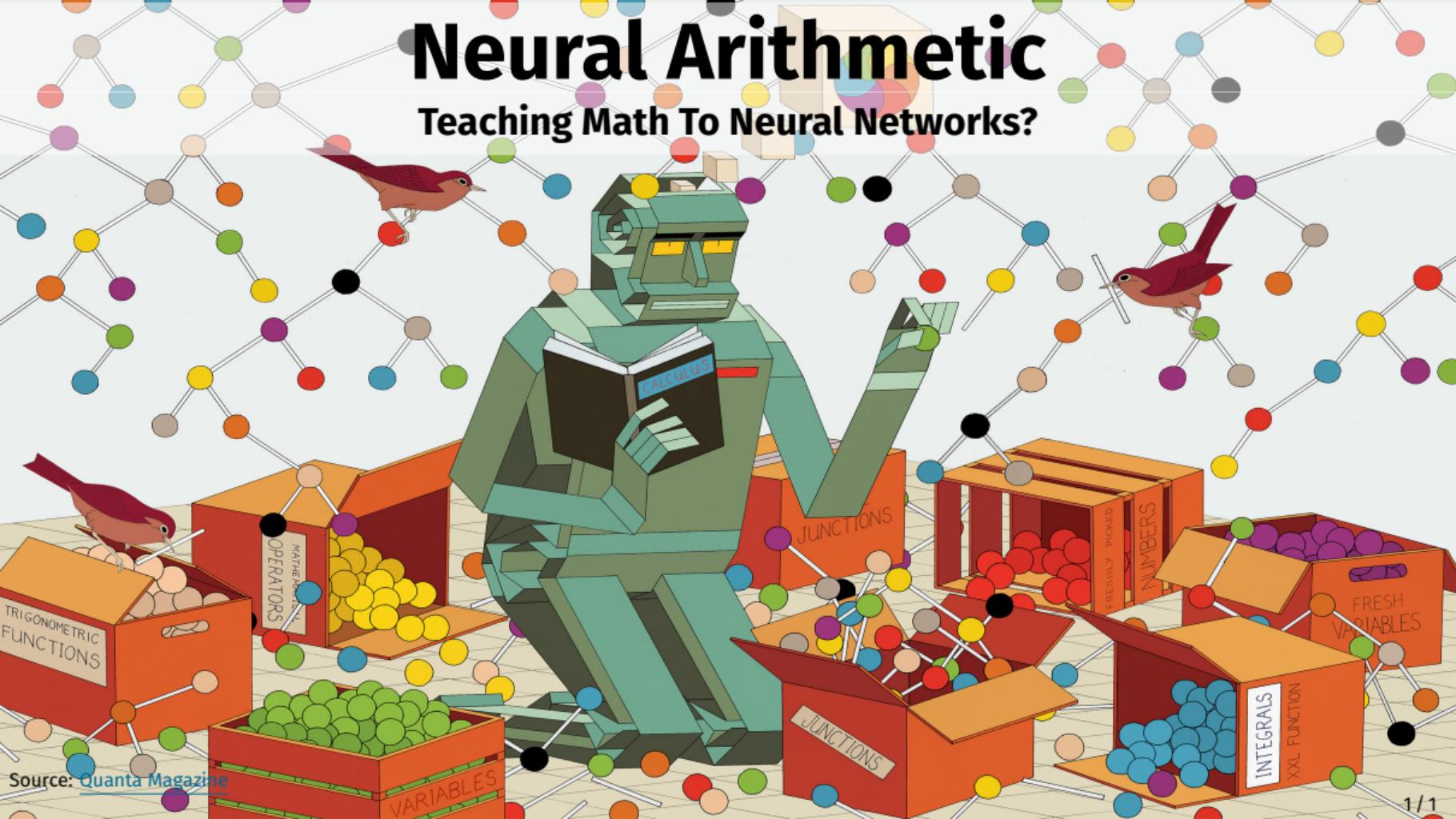


Neural Arithmetic

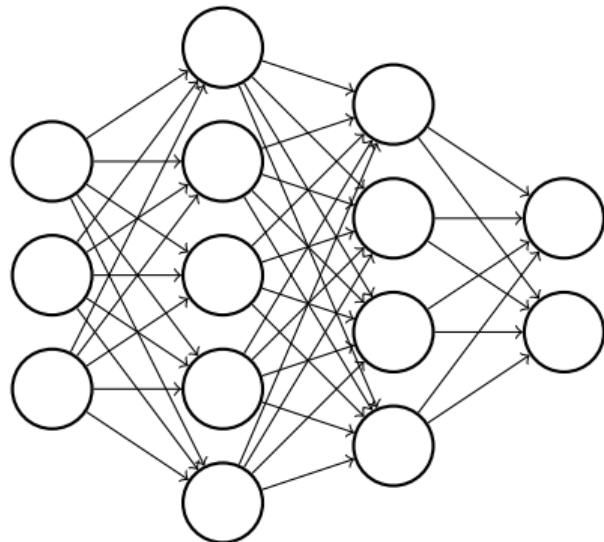
Teaching Math To Neural Networks?



Neural Arithmetic

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

Neural Networks

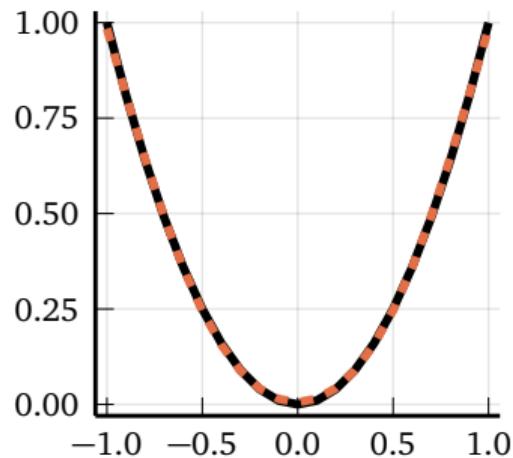


Composed of **Dense** layers:

$$y = \sigma(Wx + b)$$

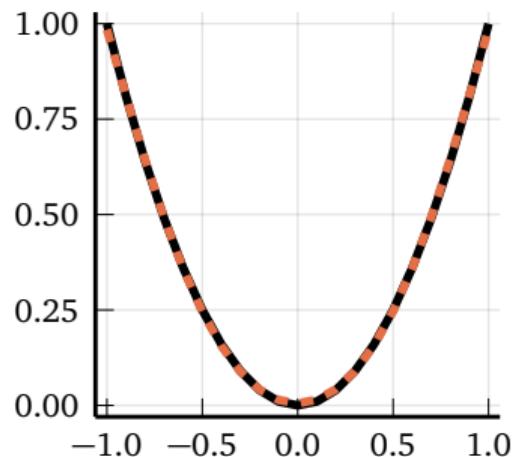
(No convolutions in this talk.)

Neural Networks

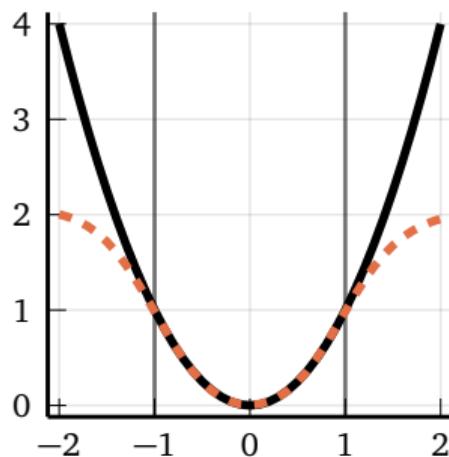


Universal **Approximators**

Neural Networks

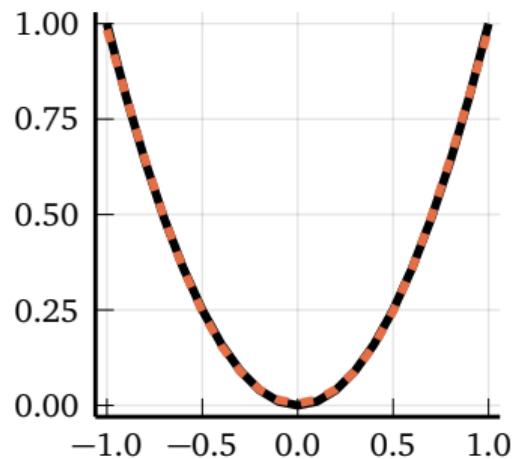


Universal Approximators

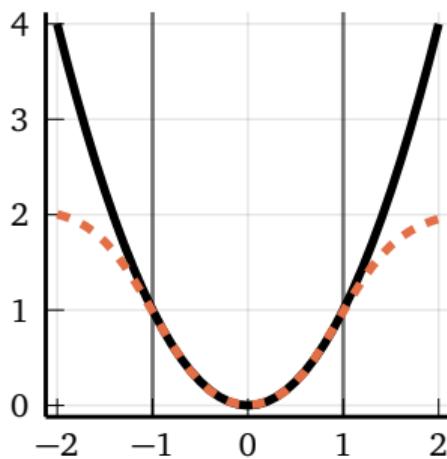


Poor Extrapolators

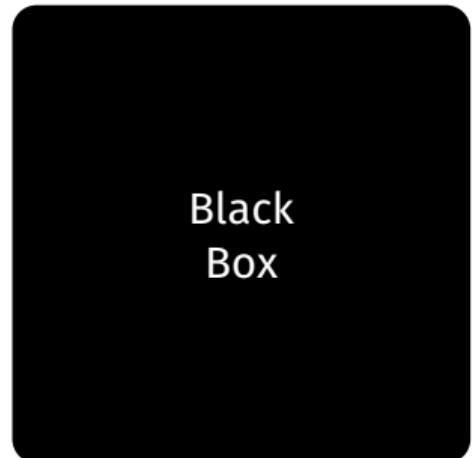
Neural Networks



Universal Approximators



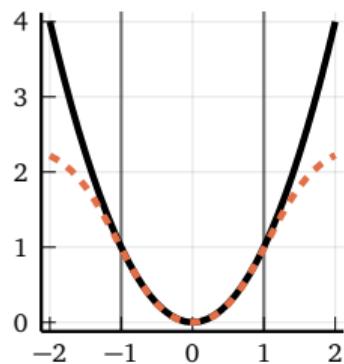
Poor Extrapolators



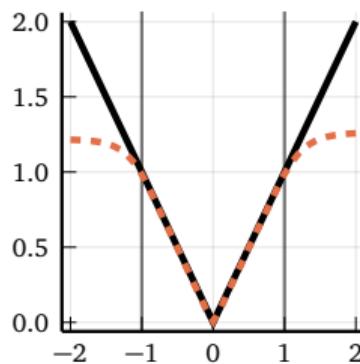
Intransparent

Problems With Extrapolation

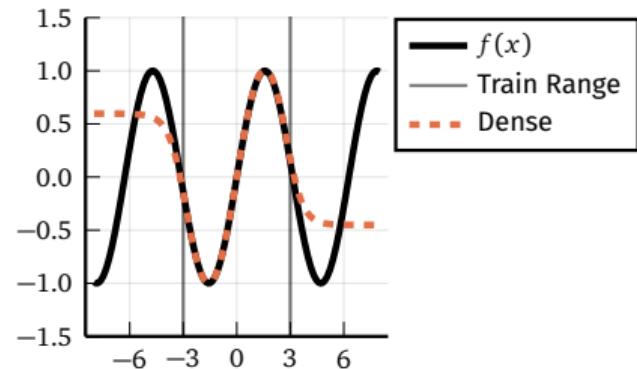
$$f(x) = x^2$$



$$f(x) = |x|$$

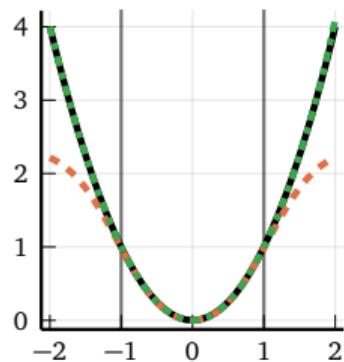


$$f(x) = \sin(x)$$

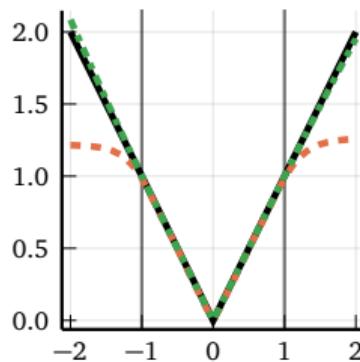


Problems With Extrapolation

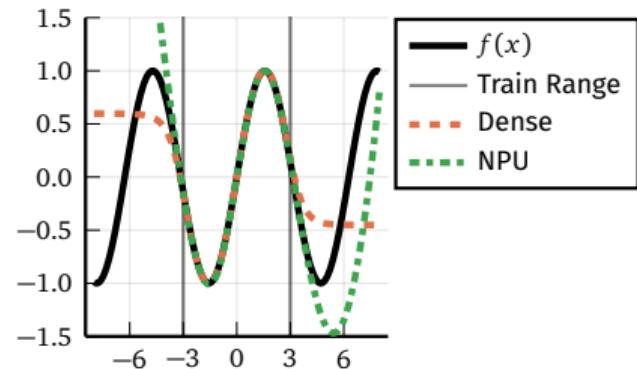
$$f(x) = x^2$$



$$f(x) = |x|$$



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



Inductive Bias

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

Addition of inputs

$$x^\top w = x_1 w_1 + x_2 w_2 + \dots$$

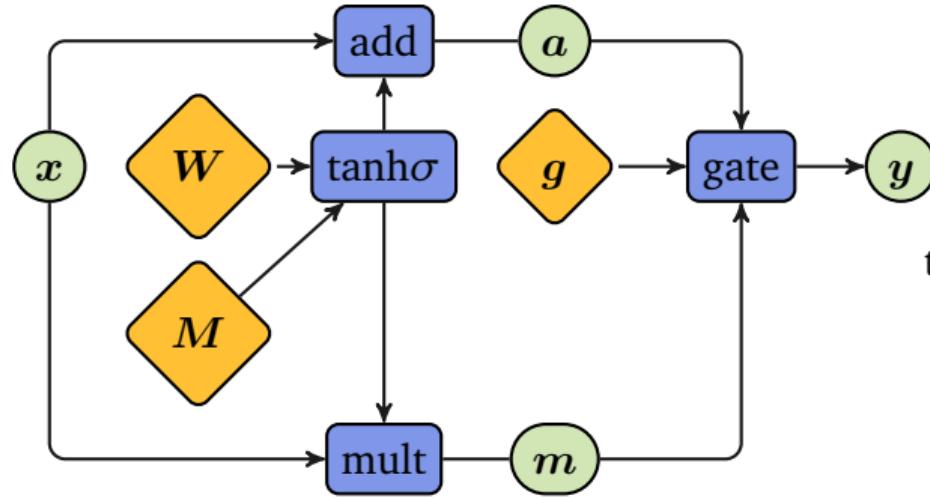
is simple.

Multiplication/power functions

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

a bit more tricky due to complex outputs...

Neural Arithmetic Logic Unit (NALU)



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

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

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

$$\tanh\sigma(W, M) = \tanh(W) \odot \sigma(M)$$

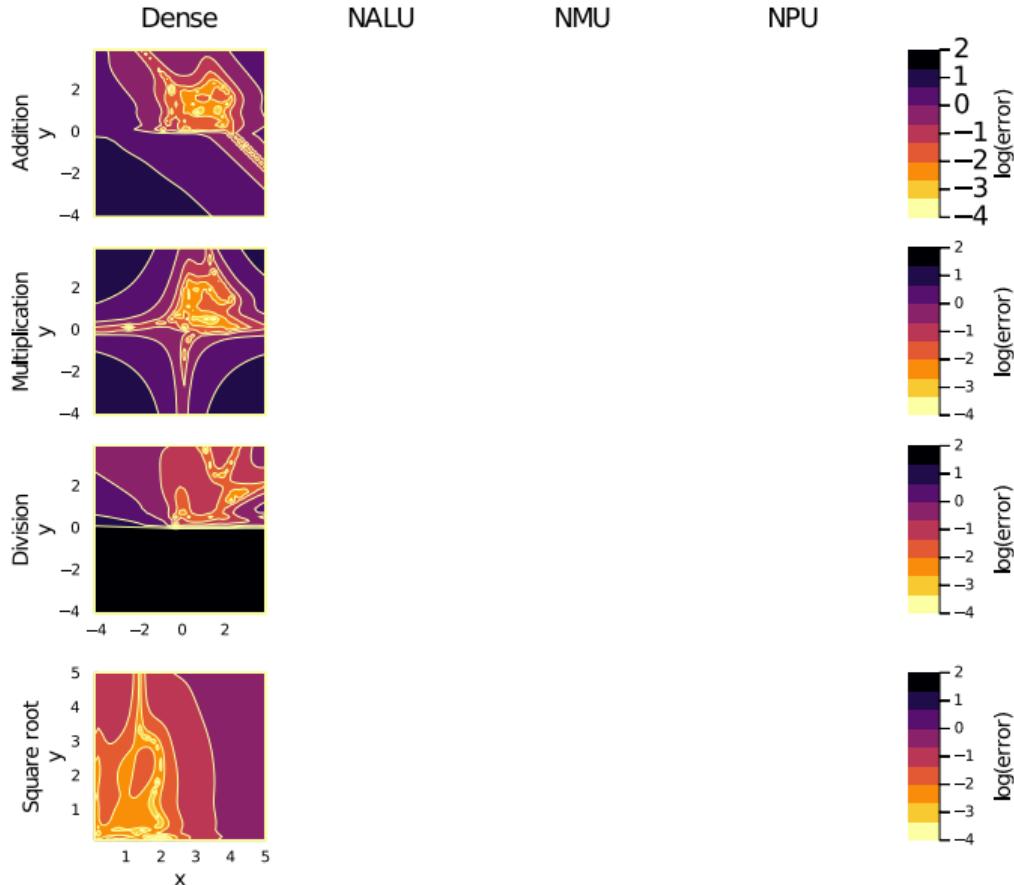
where

$$g = \sigma(Gx)$$

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

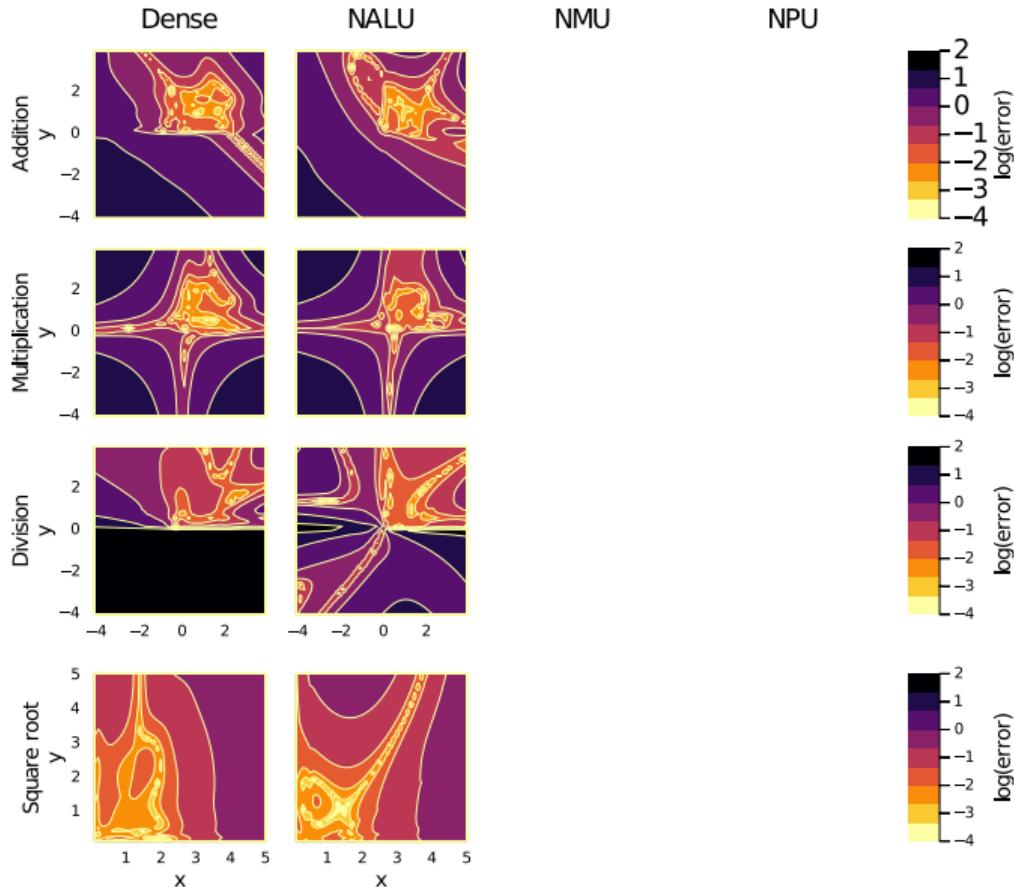
Task: Learn f in one model

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

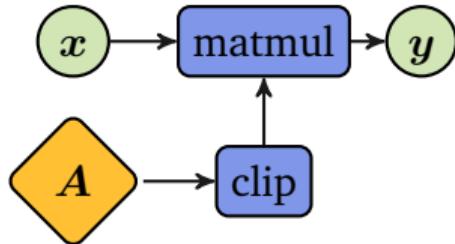


Task: Learn f in one model

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



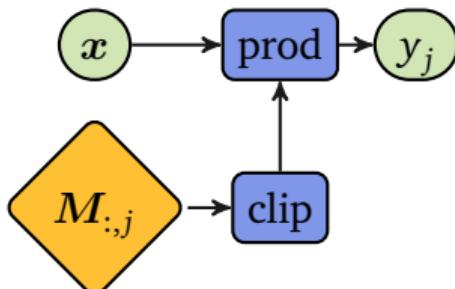
Simpler Arithmetic Units



Neural Addition Unit (NAU)

$$y = \text{clip}(A) \cdot x$$

$$\text{clip}(M) = \min(\max(M, -1), 1)$$



Neural Multiplication Unit (NMU)

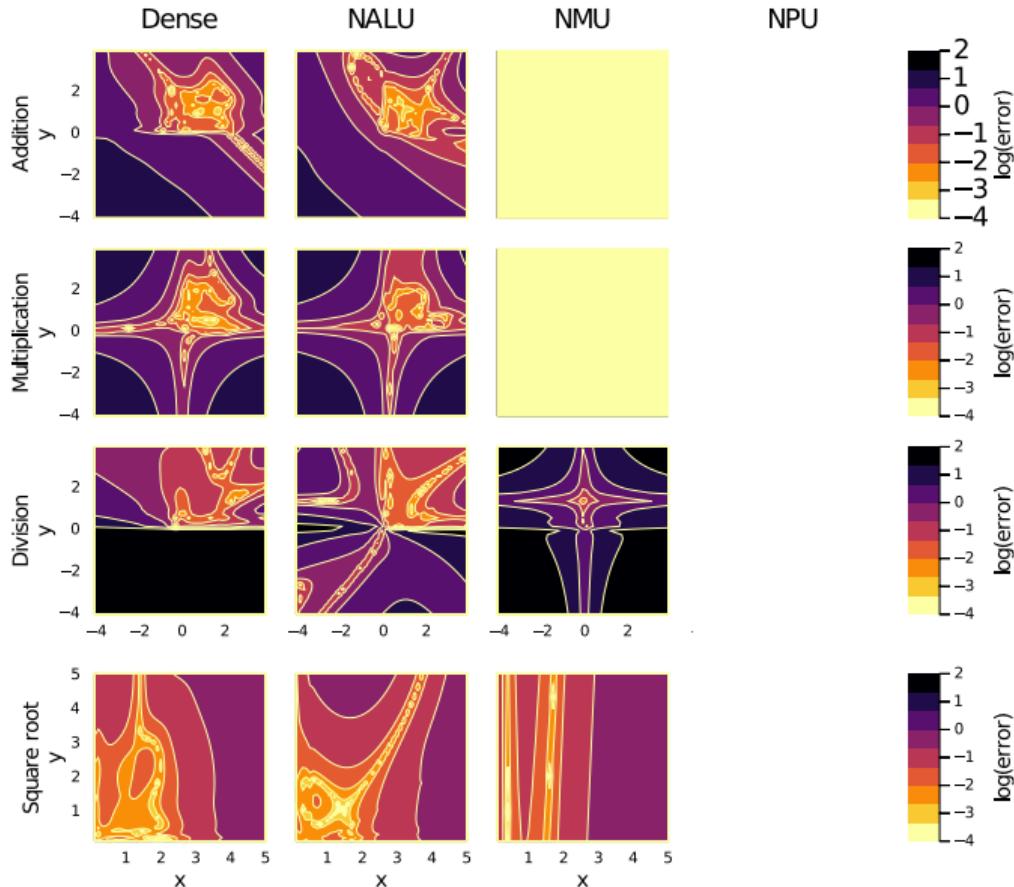
$$\text{prod}(x, \hat{M}_{:,j}) = \prod_i \hat{M}_{ij} x_i - 1 + \hat{M}_{ij}$$

$$\text{clip}(M) = \min(\max(M, 0), 1)$$

(Neural Arithmetic Units - Madsen & Johansen 2020)

Task: Learn f in one model

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



Complex space

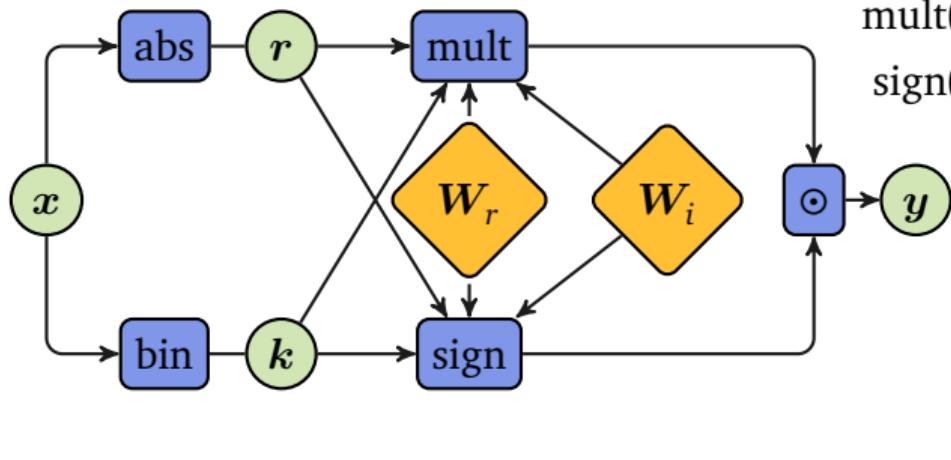
We extend NALU by isolating its multiplication path

$$m = \exp(W \log |x|)$$

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

$$z = \text{real}(\exp(W_{\mathbb{C}} \log_{\mathbb{C}} x)) = \text{real}(\exp((W_r + iW_i) \log_{\mathbb{C}} x)).$$

Naive Neural Power Unit



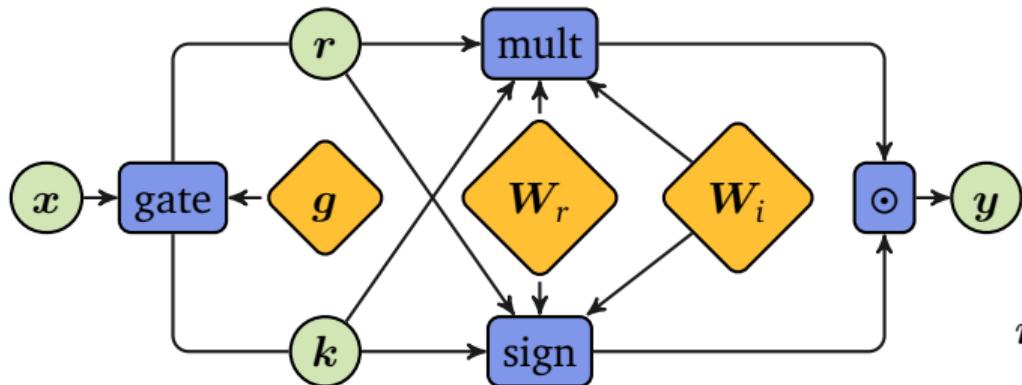
$$\text{mult}(r, k, W_r, W_i) = \exp(W_r \log r - \pi W_i k)$$

$$\text{sign}(r, k, W_r, W_i) = \cos(W_i \log r + \pi W_r k)$$

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

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

Neural Power Unit



$$\text{mult}(r, k, W_r, W_i) = \exp(W_r \log r - \pi W_i k)$$

$$\text{sign}(r, k, W_r, W_i) = \cos(W_i \log r + \pi W_r k)$$

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

where

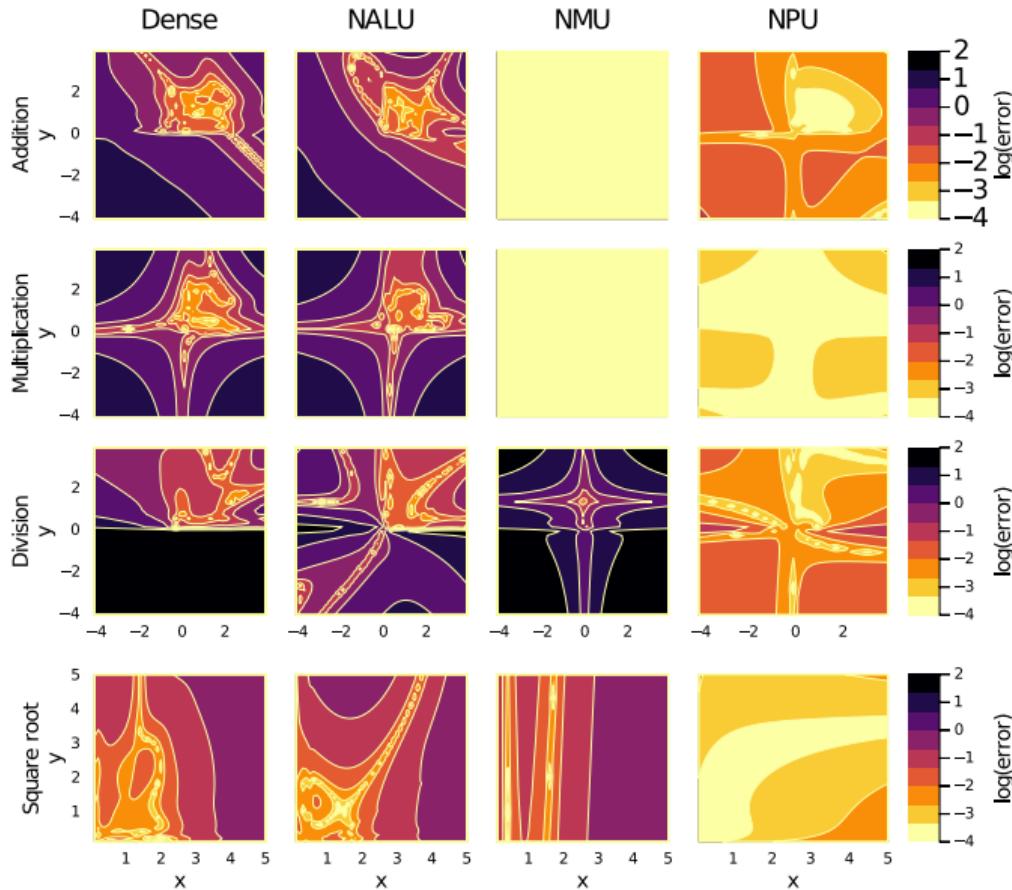
$$r = \hat{g} \odot (|x| + \epsilon) + (1 - \hat{g}),$$

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

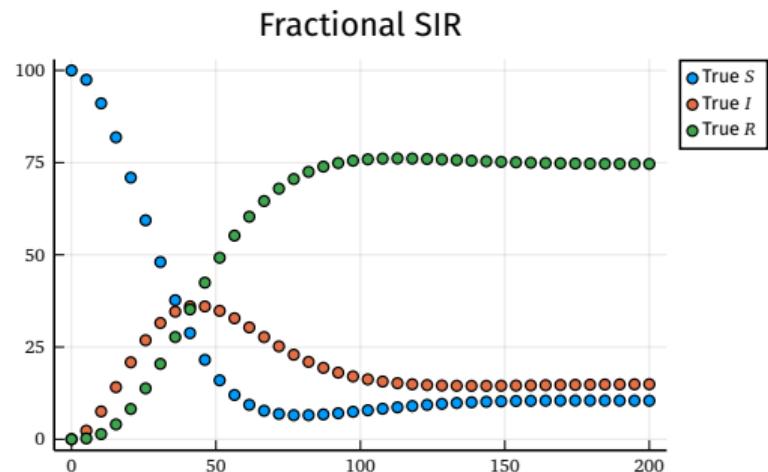
$$\hat{g} = \min(\max(g, 0), 1),$$

Task: Learn f in one model

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



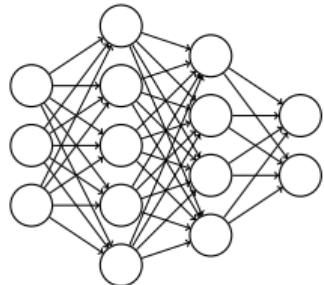
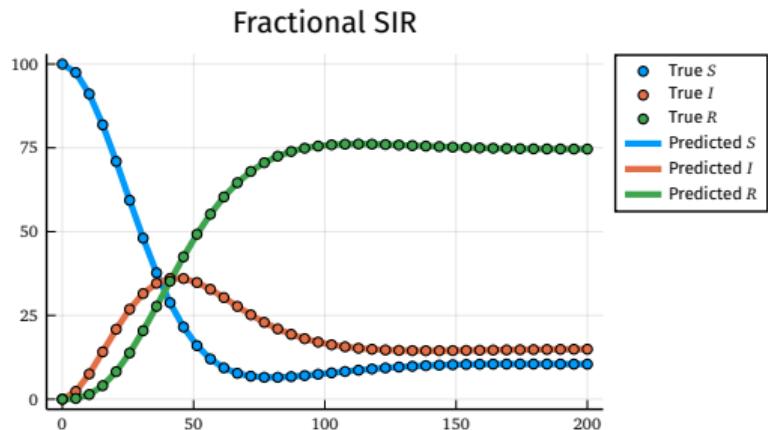
Fractional SIR



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:

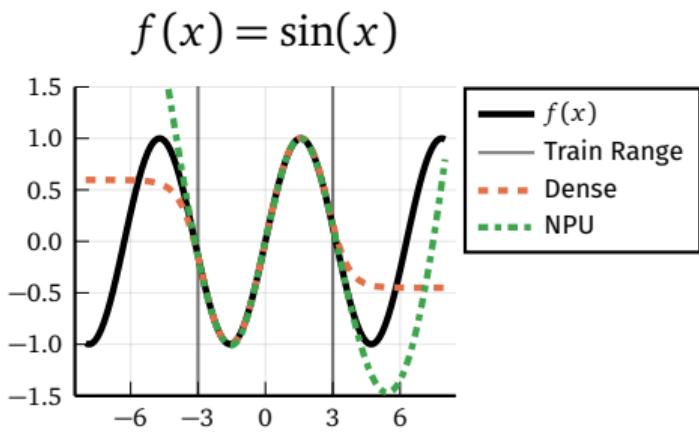
$$\begin{bmatrix} \dot{S} \\ \dot{I} \\ \dot{R} \end{bmatrix} = \text{NeuralODE} \left(\begin{bmatrix} S \\ I \\ R \end{bmatrix}, \theta \right)$$

Fractional SIR

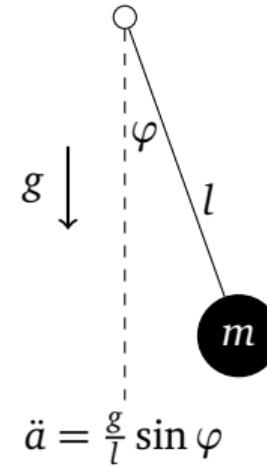
Output	NAU Layer 2						$r = \frac{S^{0.62}}{I^{0.57}}$	Hidden			NPU (real) Layer 1			Input
	-0.12r +0.13I	-0.12	0.0	0.0	-0.0	0.13		1.0	0.62	0.57	0.02	0.0	0.0	
0.11r -0.1R	0.11	0.0	0.0	-0.1	0.02	0.0	1.0	0.0	0.0	-0.0	0.0	-0.0	-0.0	S
0.09I -0.15R ^{0.6}	0.01	0.0	-0.0	0.09	-0.15	0.0	I	-0.0	1.0	0.03	-0.0	1.0	0.03	I
							$R^{0.64}$	-0.0	0.0	0.64	-0.0	0.0	0.64	R
							1.0	-0.0	-0.0	-0.0				

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

Future Work

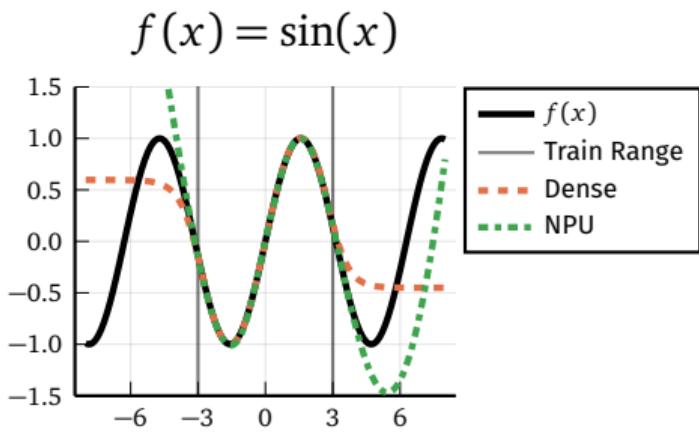


Trigonometric functions

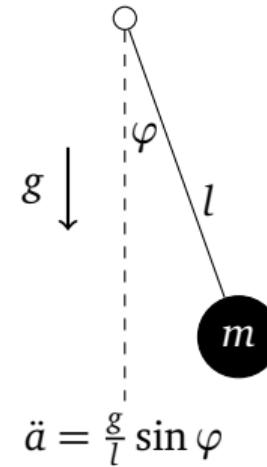


Some serious equation discovery

Future Work



Trigonometric functions



Some serious equation discovery

Do you have an application?



EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education



MINISTRY OF EDUCATION,
YOUTH AND SPORTS

This work has been supported by the OP VVV funded project Research Center for Informatics; reg. No.: CZ.02.1.01/0.0./0.0./16_019/0000765.

Neural Arithmetic

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



Niklas Heim



Tomáš Pevný



Václav Šmíd

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