Accelerating Scientific Applications with Deep Neural Networks

AMD RIPS 2021
Institute for Pure and Applied Mathematics

David Davini, Bhargav Samineni, Ben Thomas,
Amelia Tran, Cherlin Zhu

Industry Mentor: Laurent White
Academic Mentor: Kyung Ha

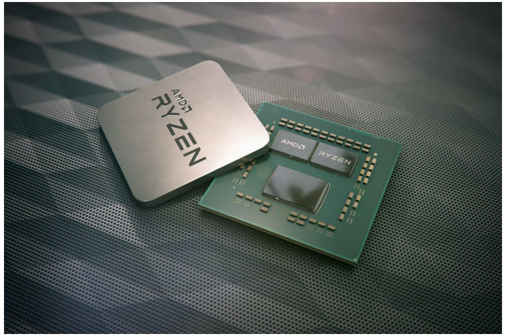
Advanced Micro Devices Inc.

- Based in Santa Clara, CA
- American multinational semiconductor manufacturing company
- Develops computer processors and technologies for business and consumer markets

Image source: Hexus

AMD Research

- AMD main products:
  - Processors and motherboard chipsets
  - Central Processing Units (CPUs)
  - Graphic Processing Units (GPUs)
- Goal: research novel scientific applications where the use of GPUs is emphasized
- Neural Network is a big application of GPUs



Wave Propagation

- Waves are everywhere: sound waves, light waves, ocean waves
- Used in everything from earthquake detection to ultrasound imaging
- Simulation time with traditional methods can take days and weeks!

Is there an alternative?

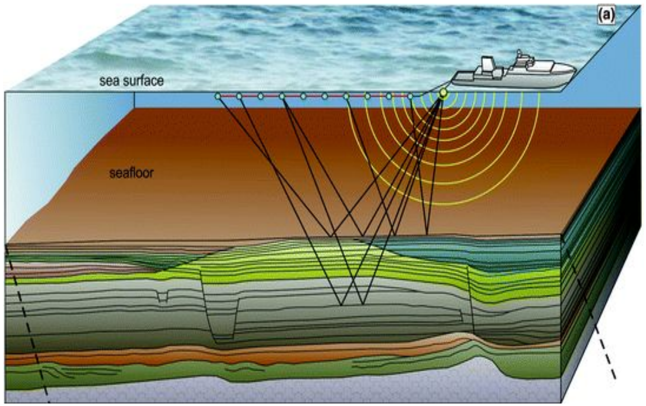
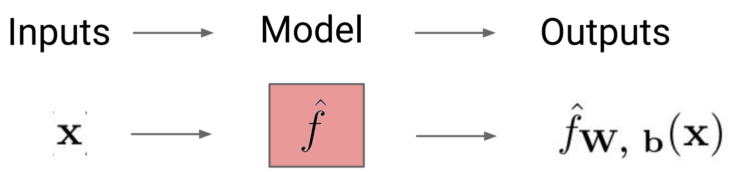


Image source: https://doi.org/10.1007/978-3-319-57852-1_4

Neural Network as an Alternative

- Neural Networks are models approximating an unknown function (f)
- Once trained, they have very fast prediction time
- We want to find $\hat{f}$ to approximate the wave equation

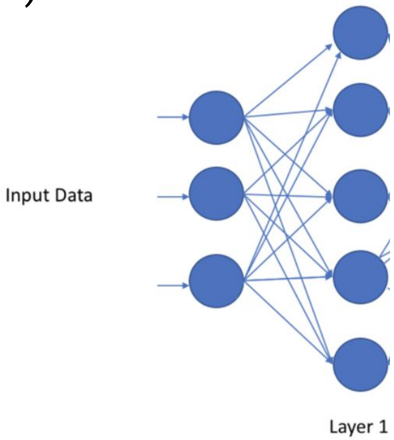


Defining Neural Networks

- NNs approximate unknown mapping
- Parameters: weights (W) and biases (b)
- Transformation includes:
  - Linear transform (W and b)
  - Nonlinear activation (σ)

$$\mathbf{a}^{(i)} = \sigma(\mathbf{W}^{(i)}\mathbf{a}^{(i-1)} + \mathbf{b}^{(i)})$$

$$\mathbf{a}^{(L)} = \hat{f}_{\mathbf{W},\ \mathbf{b}}(\mathbf{x})$$



Defining Neural Networks

- NNs approximate unknown mapping
- Parameters: weights (W) and biases (b)
- Transformation includes:
  - Linear transform (W and b)
  - Nonlinear activation (σ)

$$\mathbf{a}^{(i)} = \sigma(\mathbf{W}^{(i)}\mathbf{a}^{(i-1)} + \mathbf{b}^{(i)})$$

$$\mathbf{a}^{(L)} = \hat{f}_{\mathbf{W},\ \mathbf{b}}(\mathbf{x})$$

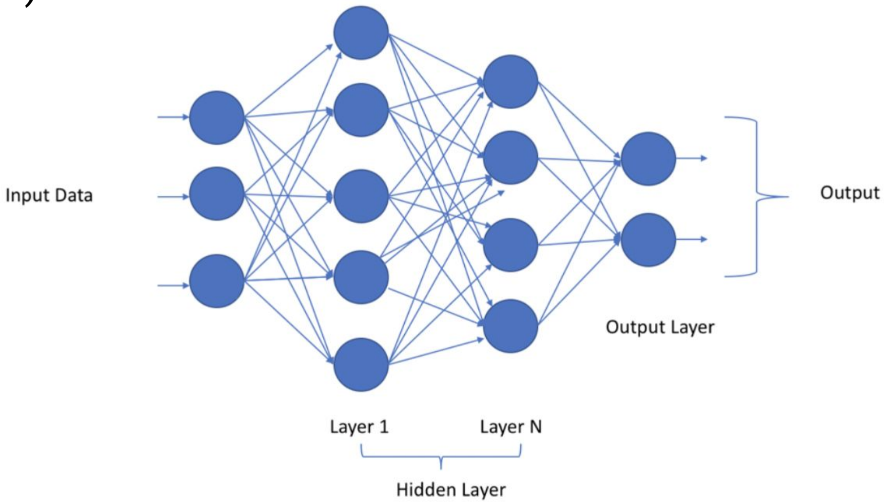


Image Source: https://towardsdatascience.com/DNN

Training Neural Networks

- Training Neural Networks:
  - Collect data
  - Solve an optimization problem by minimizing the loss function
  - Loss is error between model output and true output data

$$\mathbf{W}^*, \mathbf{b}^* = \arg\min_{\mathbf{W},\mathbf{b}} L(f, \hat{f}_{\mathbf{W},\mathbf{b}})$$

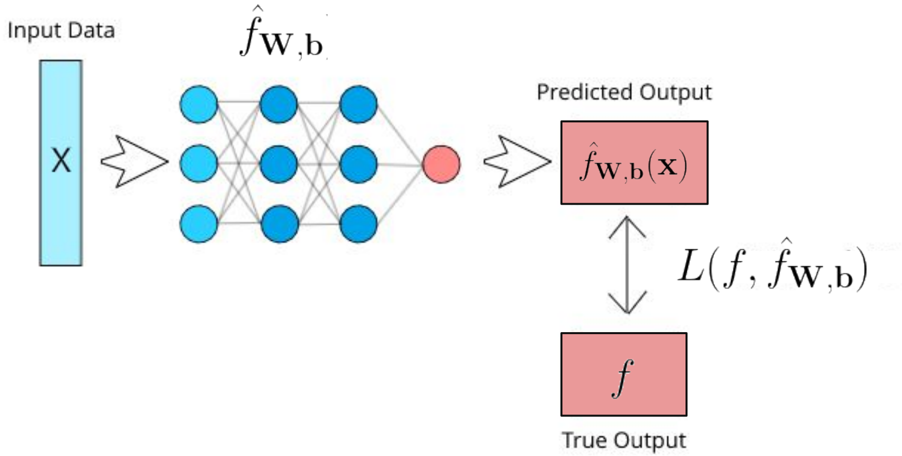
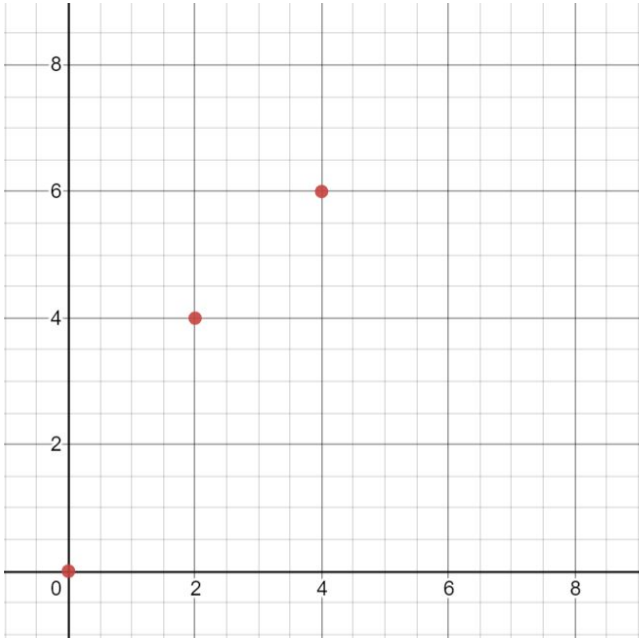


Image Source: https://deeplearningdemystified.com/articleDNN

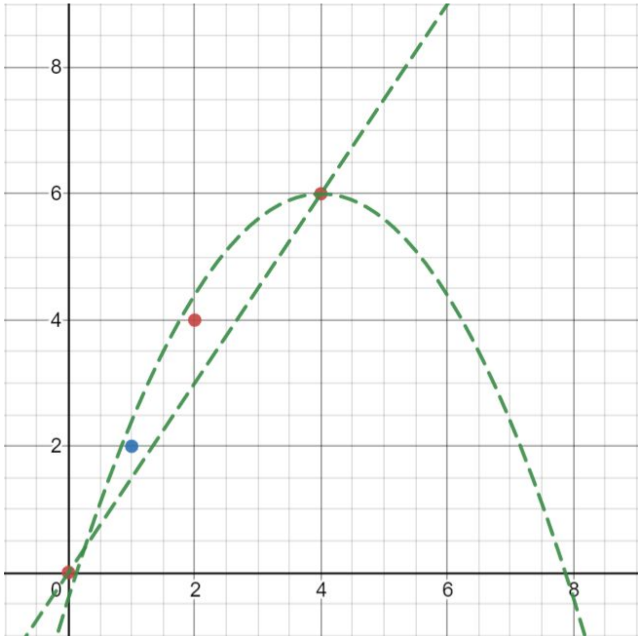
Interpolation and Extrapolation

Training Data

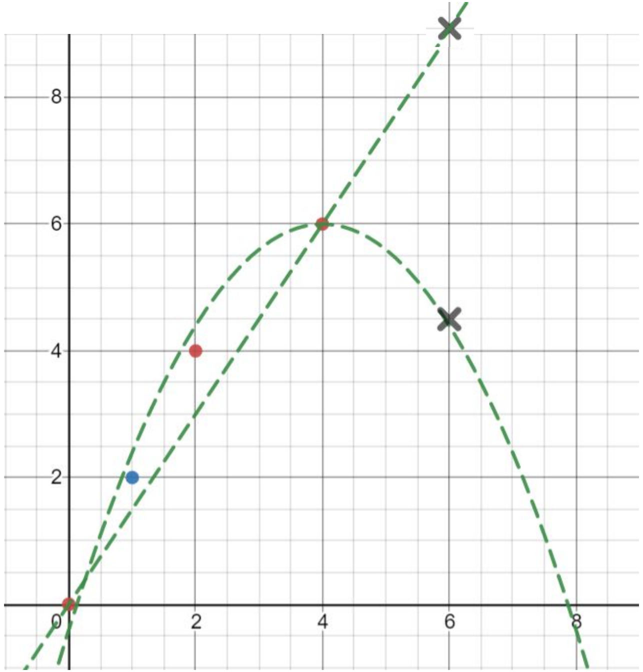


Interpolation and Extrapolation

Interpolation



Extrapolation



Neural Network models can’t extrapolate...

- Neural Network models:
  - **approximate** unknown mapping
  - trained with data

- Example:
  - Target function: y = 0
  - NN good at **interpolation** (grey)
  - NN bad at **extrapolation** (white)

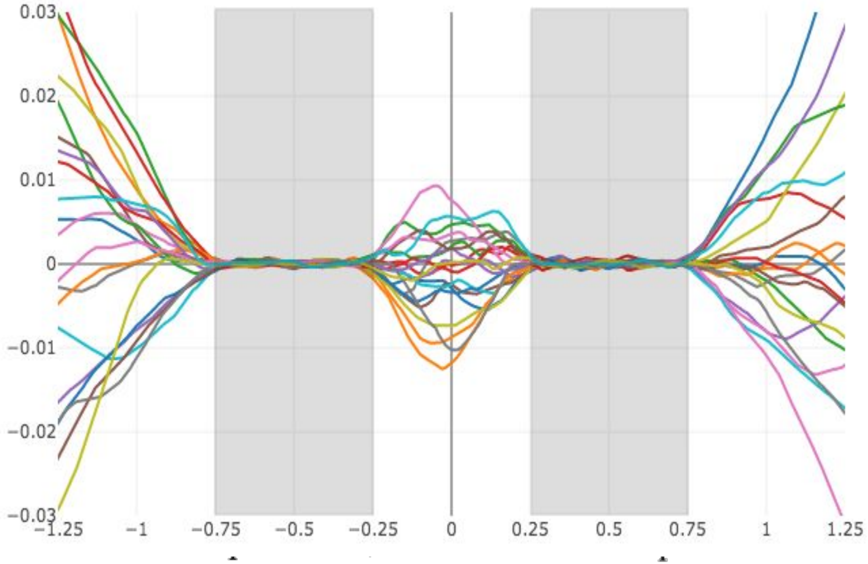


Image Source: QXplore: Q-learning Exploration by Maximizing Temporal Difference Error

Physics-Informed Neural Networks (PINNs)

- Recent research:
  - Physics Informed Neural Networks
  - Train NNs with differential equations describing physical systems
  - PINNs **successfully extrapolate** and **ensure physically consistent output**

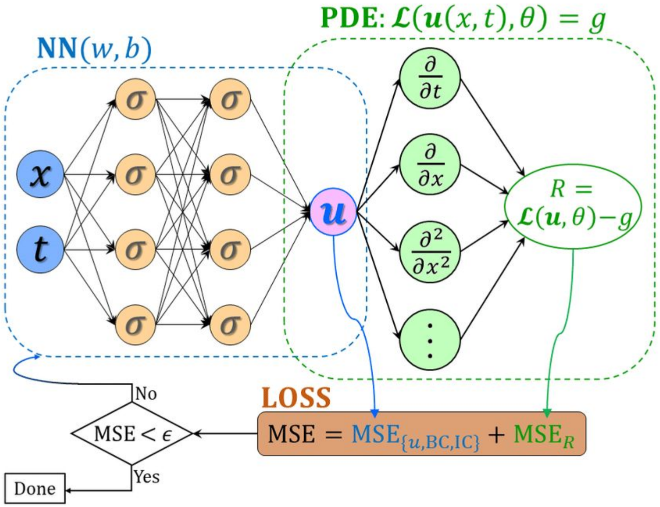


Image Source: https://www.researchgate.net/PINN

Project Description

Implement physics-informed neural network algorithm to accurately extrapolate the wave equation

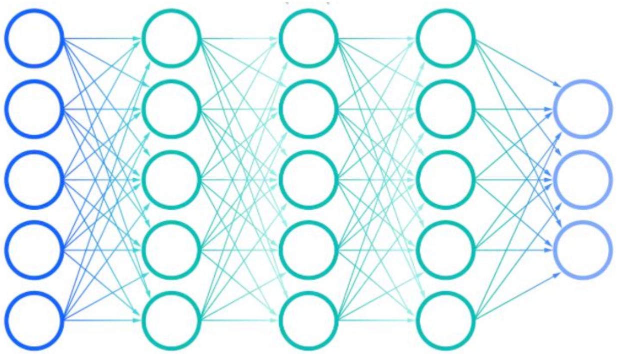
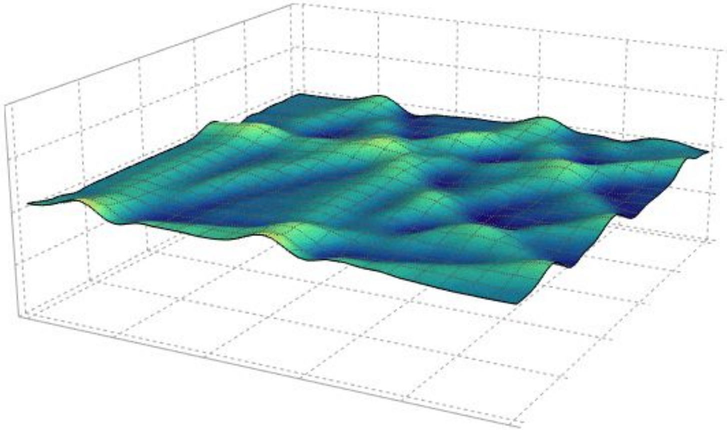


Image source: https://www.ibm.com/cloud/learn/neural-networks

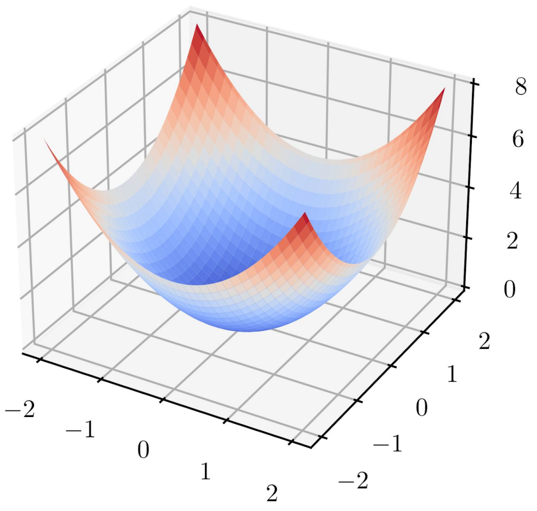


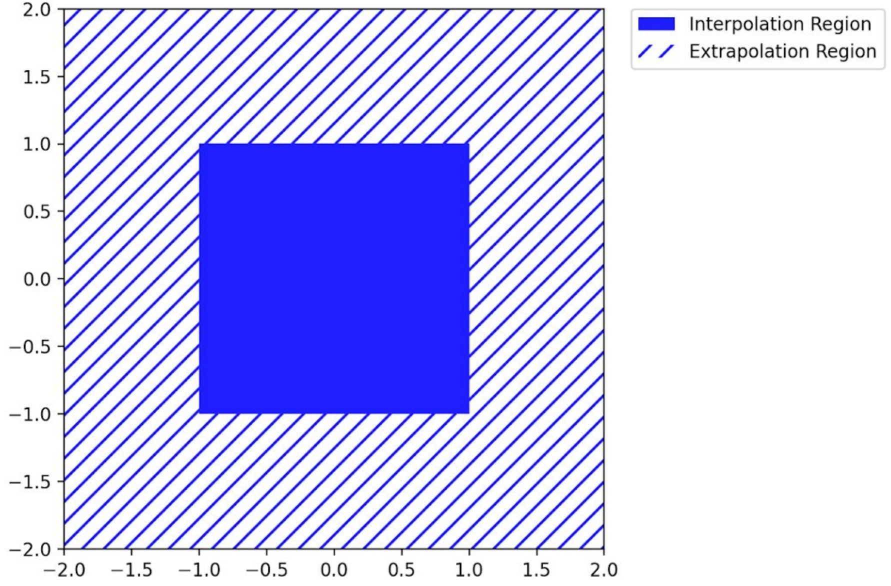
Overview

- Introduction
- Neural Networks - Amelia
- Paraboloid Extrapolation - Bhargav
- Wave Equation Extrapolation - Ben
- Weight Analysis - David
- Conclusion - Cherlin

Target Function: Paraboloid (Toy Problem)

$$f(x, y) = x^2 + y^2$$





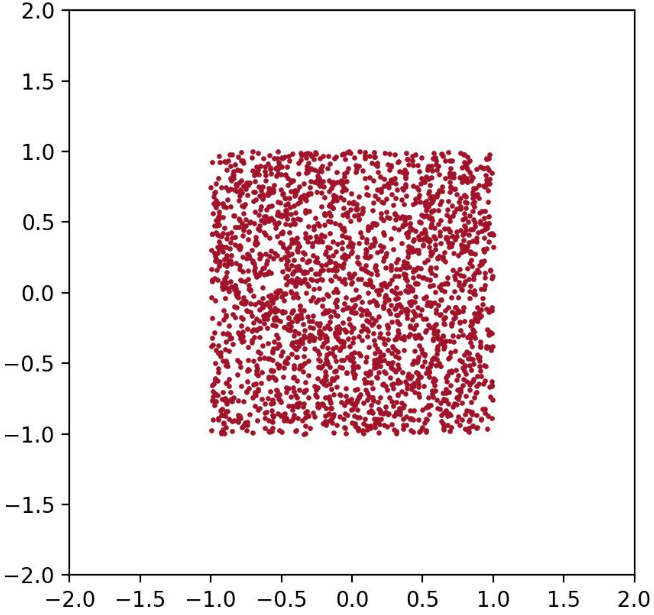
Main Goal: Improve accuracy of network in extrapolation region

Data Sampling for Baseline Model

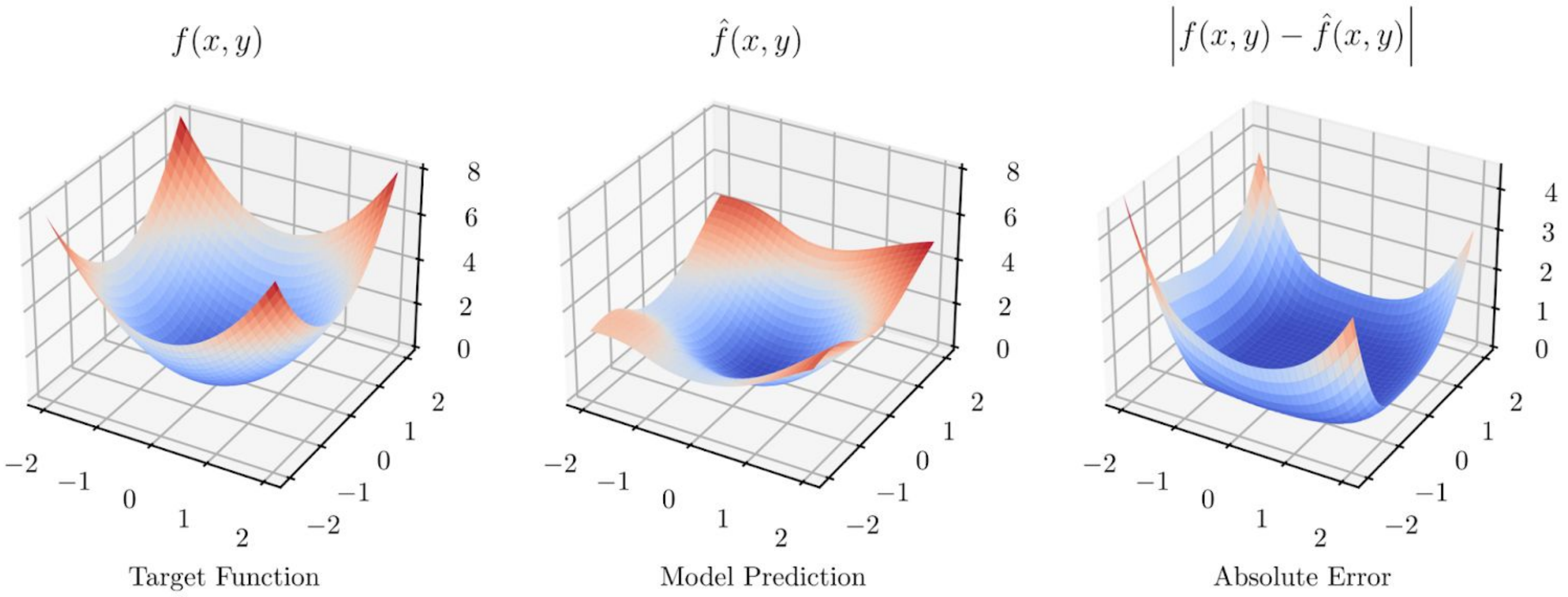
Uniform random sampling of labeled points from the interpolation region

Loss Function:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left(f\left(\mathbf{x}^{\{i\}}\right) - \hat{f}\left(\mathbf{x}^{\{i\}}\right)\right)^2$$



Baseline Model



Interpolation Region Error Avg*
1.79E-3

Extrapolation Region Error Avg*
8.46E-1

*RMSE averaged across 10 trials

Physical Constraints

$$f(x, y) = ax^2 + by^2$$

Third Order Partials

$$\begin{aligned} & f_{xxx},\ f_{xxy},\ f_{xyy}, \\ & f_{xyx},\ f_{yxy}, \qquad = \qquad 0 \\ & f_{yxx},\ f_{yyx},\ f_{yyy} \end{aligned}$$

$\Longrightarrow$

Third Order Regularizer

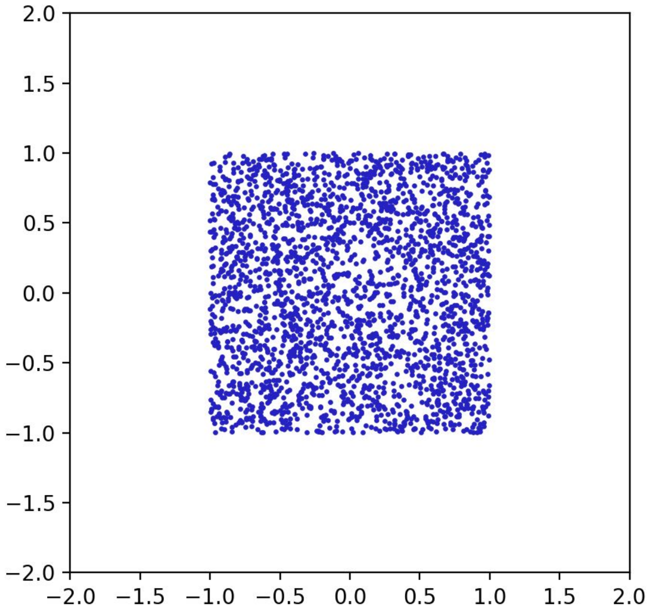
$$\begin{aligned} E_{\text{third}}\left(\hat{f}, \mathbf{x}^{\{i\}}\right) &= \left|\hat{f}_{xxx}\left(\mathbf{x}^{\{i\}}\right)\right| + \left|\hat{f}_{xxy}\left(\mathbf{x}^{\{i\}}\right)\right| + \ldots \\ &\quad + \left|\hat{f}_{yyx}\left(\mathbf{x}^{\{i\}}\right)\right| + \left|\hat{f}_{yyy}\left(\mathbf{x}^{\{i\}}\right)\right| \\ &\approx 0 \end{aligned}$$

Data Sampling for PINN Model

Uniform random sampling of labeled points from the interpolation region

Loss Function Example:

$$\mathcal{L}_{\text{int}} = \frac{1}{N}\sum_{i=1}^{N}\left(\left(f\left(\mathbf{x}^{\{i\}}\right) - \hat{f}\left(\mathbf{x}^{\{i\}}\right)\right)^2 + \lambda E_{\text{third}}\left(\hat{f}, \mathbf{x}^{\{i\}}\right)\right)$$

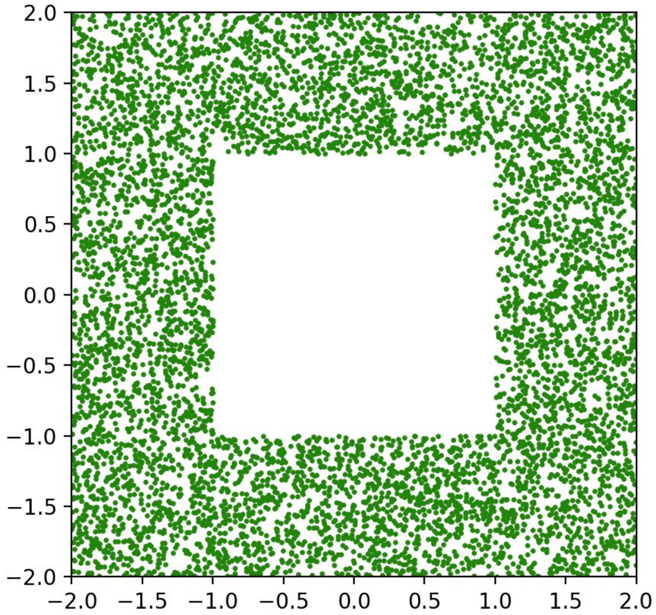


Data Sampling for PINN Model

Uniform random sampling of collocation points from the extrapolation region

Loss Function Example:

$$\mathcal{L}_{\text{ext}} = \frac{\lambda}{N} \sum_{i=1}^{N} E_{\text{third}} \left(\hat{f}, \mathbf{x}^{\{i\}} \right)$$



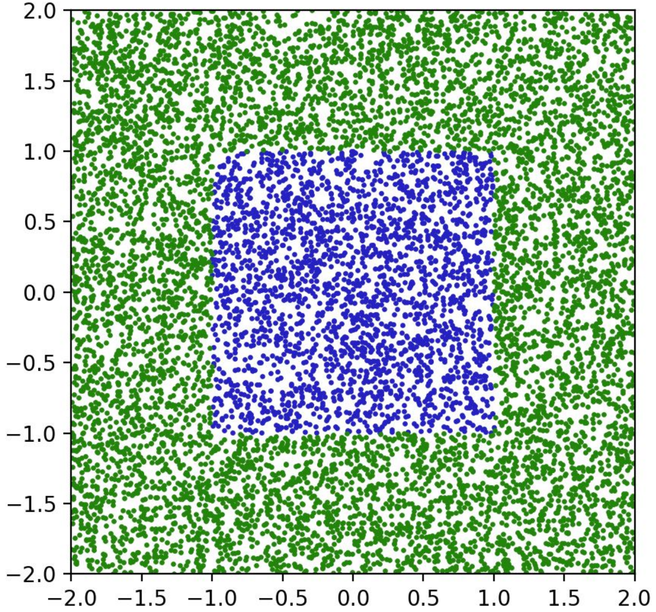
Data Sampling for PINN Model

Uniform random sampling of labeled points from the interpolation region and collocation points from the extrapolation region

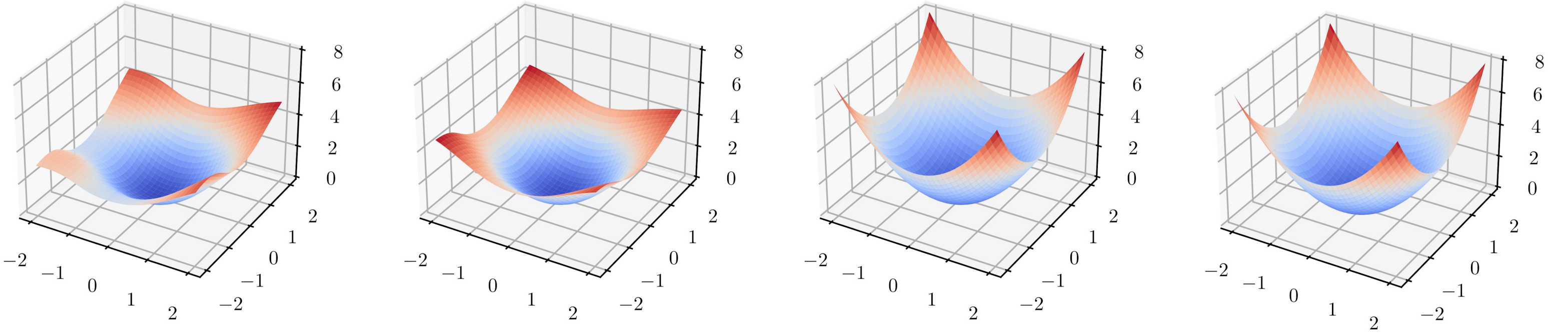
Loss Function Example:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} L\left(\hat{f}, \mathbf{x}^{\{i\}}\right) \text{ where}$$

$$L\left(\hat{f}, \mathbf{x}^{\{i\}}\right) = \begin{cases} \left(f\left(\mathbf{x}^{\{i\}}\right) - \hat{f}\left(\mathbf{x}^{\{i\}}\right)\right)^2 + \lambda E_{\text{third}}\left(\hat{f}, \mathbf{x}^{\{i\}}\right) & \text{if } \mathbf{x}^{\{i\}} \in \Omega_{\text{int}} \\ \lambda E_{\text{third}}\left(\hat{f}, \mathbf{x}^{\{i\}}\right) & \text{if } \mathbf{x}^{\{i\}} \in \Omega_{\text{ext}} \end{cases}$$



Gradient Regularizer Results

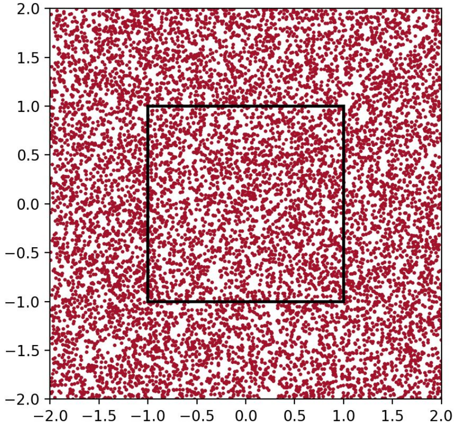


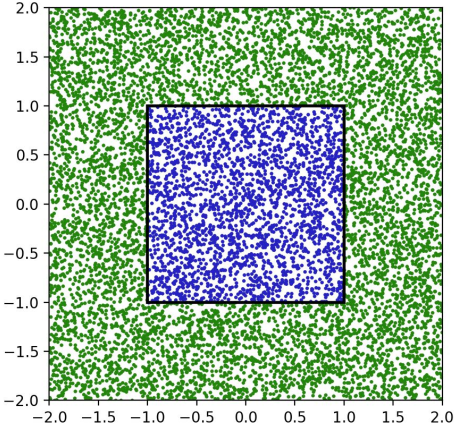
	Baseline Model	1st Order Regularizer	2nd Order Regularizer	3rd Order Regularizer
Interpolation Region Error Avg*	1.79E-3	2.18E-2	**1.39E-3**	5.11E-3
Extrapolation Region Error Avg*	8.46E-1	1.19	**3.00E-3**	1.90E-2

*RMSE averaged across 10 trials

Further Results

- $\mathcal{L} = \text{MSE}$
- $\mathcal{L} = \text{MSE} + \mathbf{E}_{phys}$
- $\mathcal{L} = \mathbf{E}_{phys}$





	Pure Interpolation (10,000 points with no physics information)	2nd Order Regularizer (2500 labeled Int, 7500 collocation Ext)
Interpolation Region Error Avg*	1.79E-3	**1.39E-3**
Extrapolation Region Error Avg*	3.02E-3	**3.00E-3**

*RMSE averaged across 10 trials

Target Function: Wave equation

First Order System of Wave Equation

$$\frac{\partial p}{\partial t} + \kappa \nabla \cdot \mathbf{v} = 0$$

$$\frac{\partial \mathbf{v}}{\partial t} + \frac{1}{\rho} \nabla p = 0$$

Variables

$\mathrm{p} = \text{pressure}$

$\mathbf{v} = \text{velocity}$

$\rho = \text{density}$

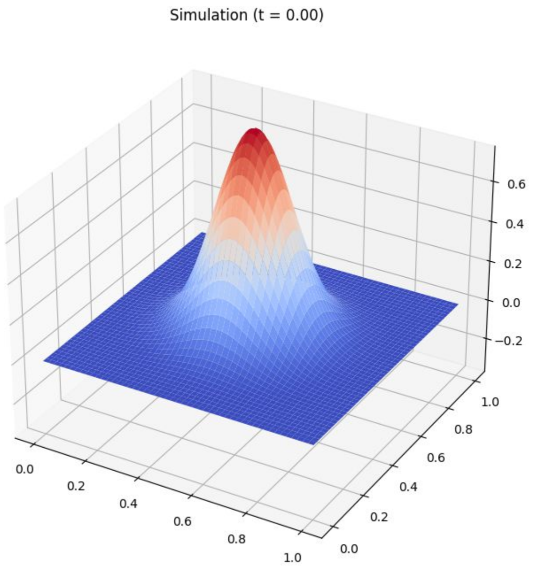
$\kappa = \text{bulk modulus of compressibility}$

Reflective Boundary Conditions

$$\mathbf{v}_n = 0$$

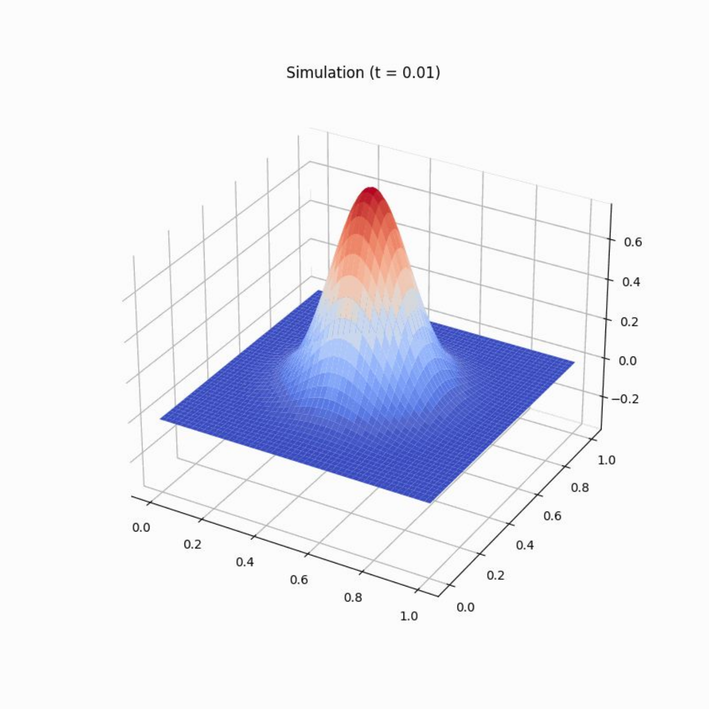
Simulation

- Gaussian Initialization
- Data from simulation used as input for Neural Network
- Interpolation for $t \in [0, 1)$
- Extrapolation for $t \in [1, 2]$



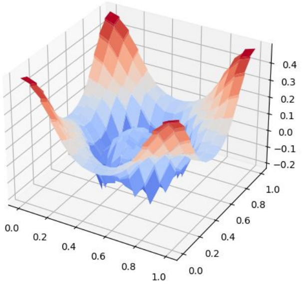
Simulation

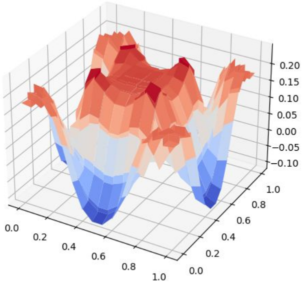
- Gaussian Initialization
- Data from simulation used as input for Neural Network
- Interpolation for $t \in [0, 1)$
- Extrapolation for $t \in [1, 2]$

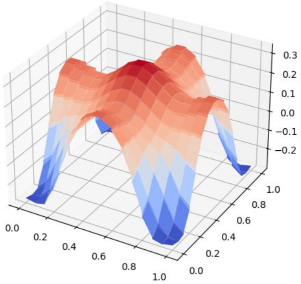


Baseline

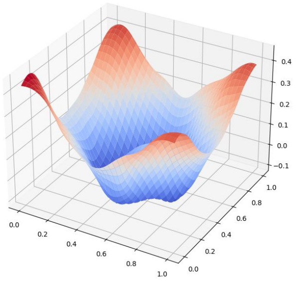
Simulation(p)

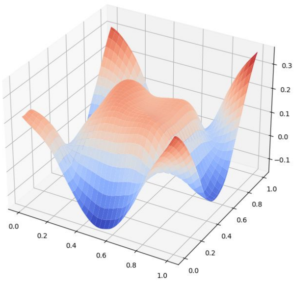


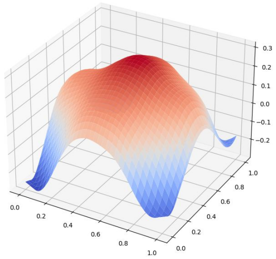




Model($\hat{p}$)





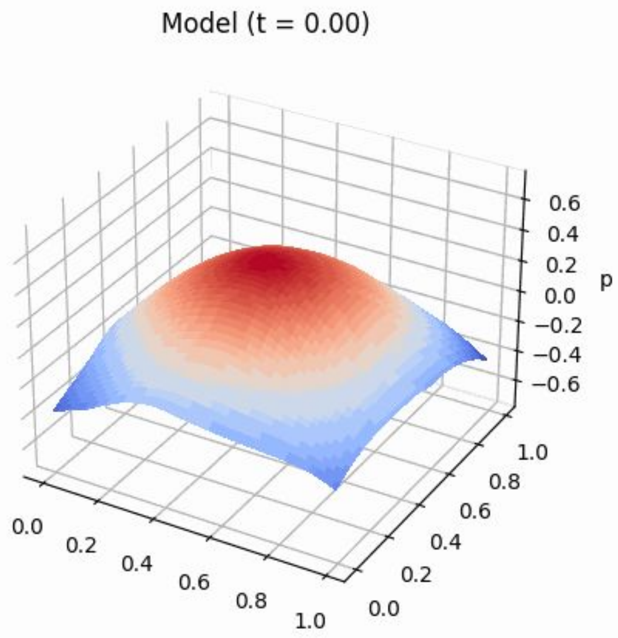


t = 0.60

t = 0.80

t = 1.00

Baseline



$$\mathcal{L} = MSE$$

Neural Networks

$$\hat{f}(x, y, t) = \hat{f}(\mathbf{x}) = (\hat{p}, \hat{u}, \hat{v})$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(\mathbf{x}^{\{i\}}) - \hat{f}(\mathbf{x}^{\{i\}}))^2}$$

Physics-Informed Neural Network

First Order Equations

$$\frac{\partial p}{\partial t} + \kappa \nabla \cdot \mathbf{v} = 0$$

$$\frac{\partial \mathbf{v}}{\partial t} + \frac{1}{\rho} \nabla p = 0$$

First Order Regularizer

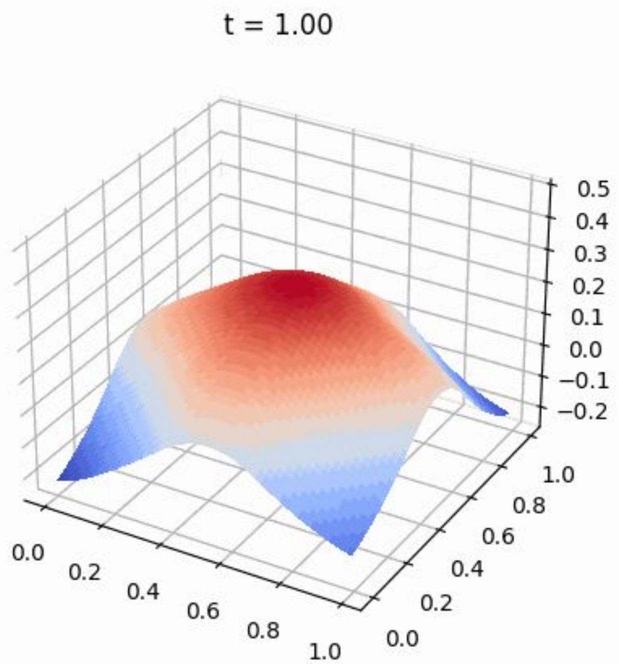
$$E_{\text{first}}(\hat{f}, \mathbf{x}^{\{i\}}) = \left| \hat{p}_t(\mathbf{x}^{\{i\}}) + \kappa \nabla \cdot \langle \hat{u}(\mathbf{x}^{\{i\}}), \hat{v}(\mathbf{x}^{\{i\}}) \rangle \right|$$

$$+ \left\| \langle \hat{u}_t(\mathbf{x}^{\{i\}}), \hat{v}_t(\mathbf{x}^{\{i\}}) \rangle + \tfrac{1}{\rho} \nabla \hat{p}(\mathbf{x}^{\{i\}}) \right\|$$

Second Order Regularizer

$$E_{\text{second}}(\hat{f}, \mathbf{x}^{\{i\}}) = \left| \hat{p}_{tt}(\mathbf{x}^{\{i\}}) - c(\hat{p}_{xx}(\mathbf{x}^{\{i\}}) + \hat{p}_{yy}(\mathbf{x}^{\{i\}})) \right|$$

Second Order



$$\mathcal{L} = MSE + \frac{\lambda_2}{N} \sum_{i=1}^{N} E_{\text{second}}(\hat{f}, \mathbf{x}^{\{i\}})$$

Boundary Conditions

$$\mathbf{v}_n = \mathbf{0}$$

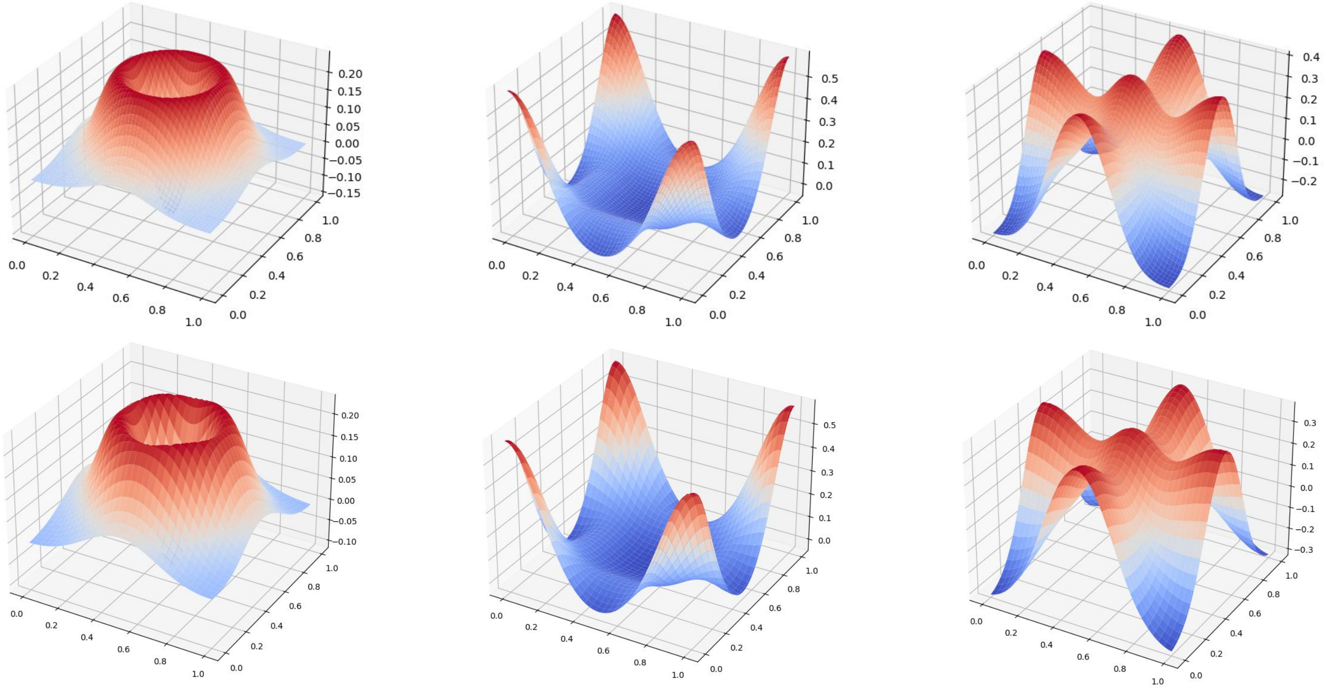
$$E_{\text{bound}} = \begin{cases} |\hat{u}| & \text{if } \; x = 0, x = 1 \\ |\hat{v}| & \text{if } \; y = 0, y = 1 \\ 0 & \text{otherwise} \end{cases}$$

Interpolation Results

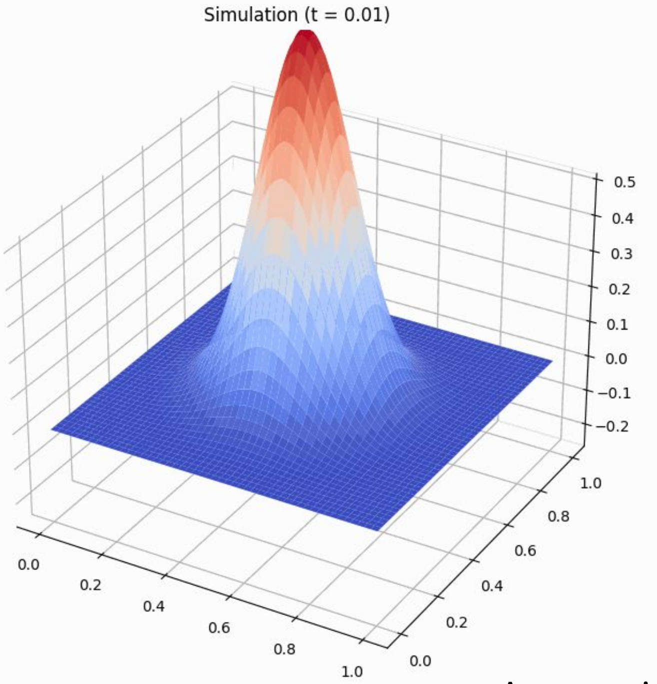
t = 0.2 | t = 0.6 | t = 1.0

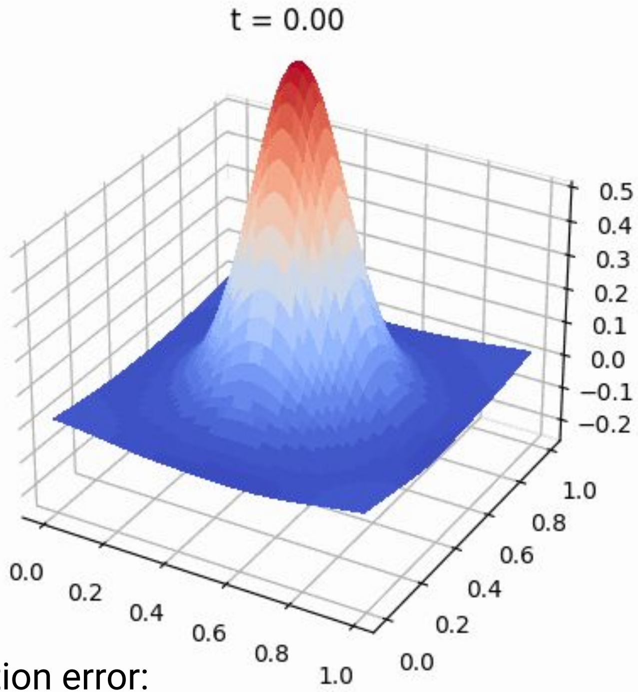
Simulation(p)

Model$(\hat{p})$



Interpolation results





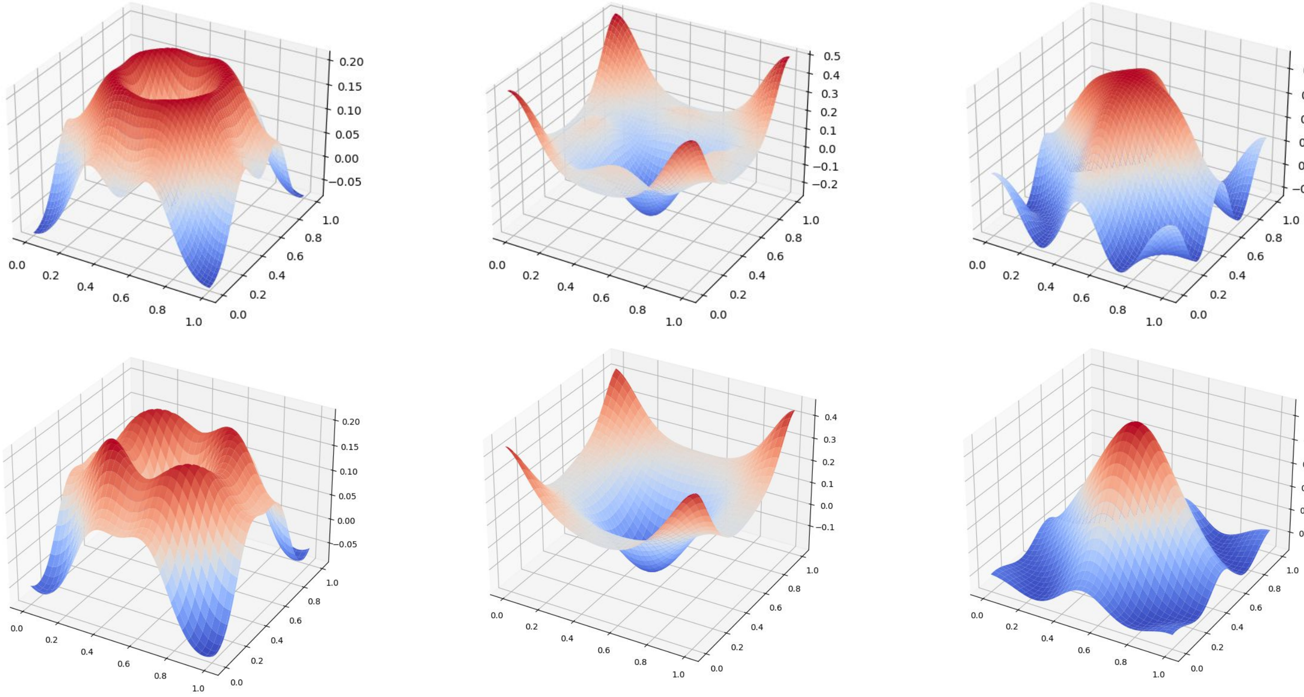
Average interpolation error:
1.04E-2

Extrapolation Results

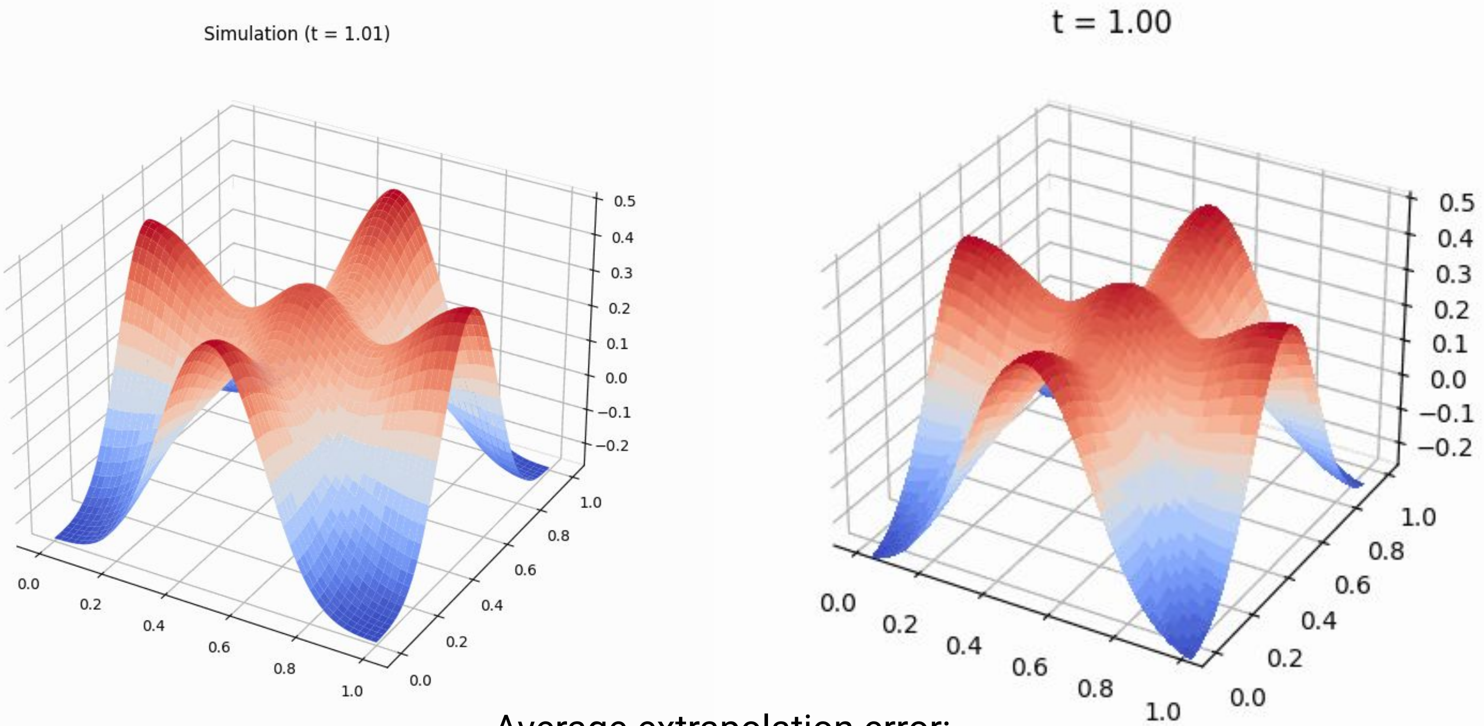
t = 1.2 | t = 1.6 | t = 2.0

$\text{Simulation}(p)$

$\text{Model}(\hat{p})$

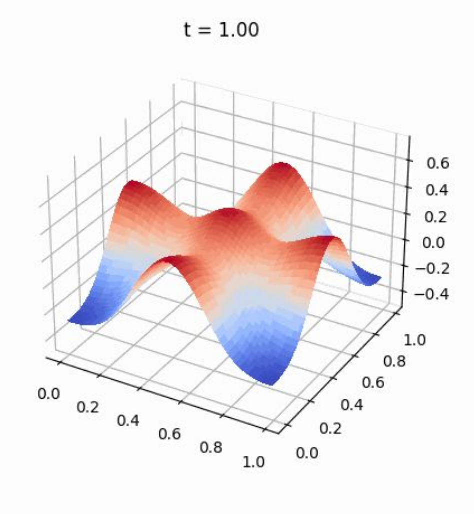


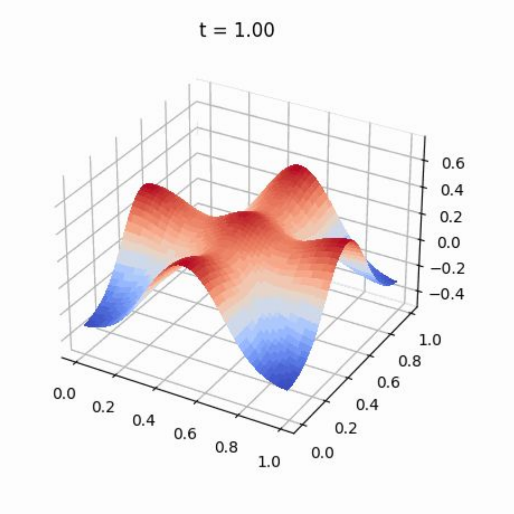
Extrapolation Results



Average extrapolation error:
2.33E-2

Errors for Wave





	Pure Interpolation	Gradient Regularizers
Interpolation Region Error	6.0E-4	1.04E-2
Extrapolation Region Error	1.66E-1	2.33E-2

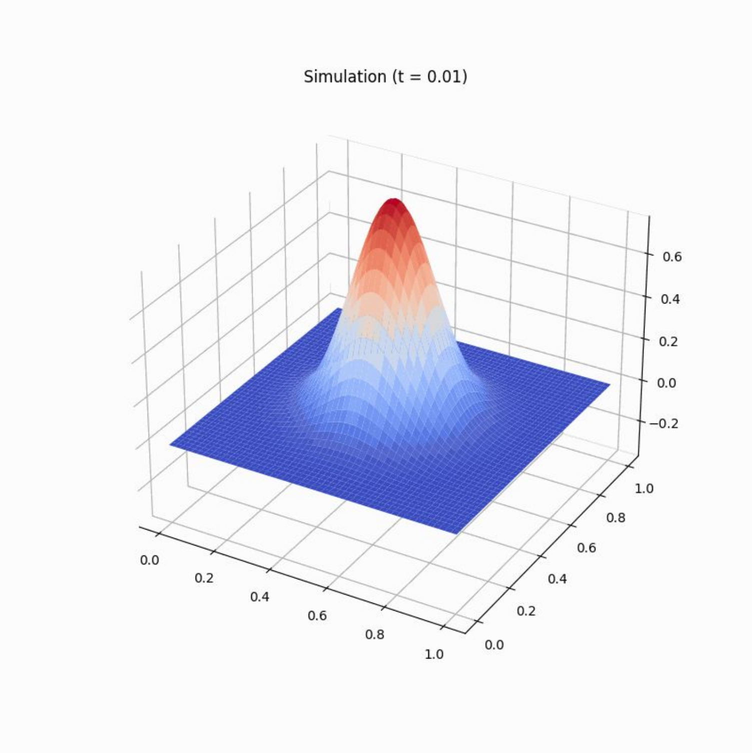
Source Location as an Input

Previous model:

$$\hat{f}(x, y, t) = (\hat{p}, \hat{u}, \hat{v})$$

New model:

$$\hat{f}(0.5, 0.5, x, y, t) = (\hat{p}, \hat{u}, \hat{v})$$



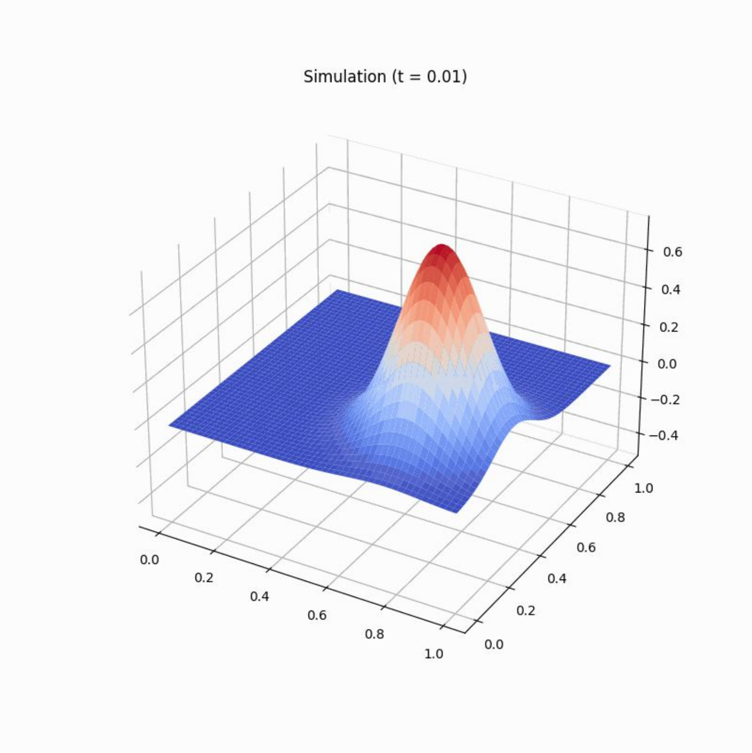
Source Location as an Input

Previous model:

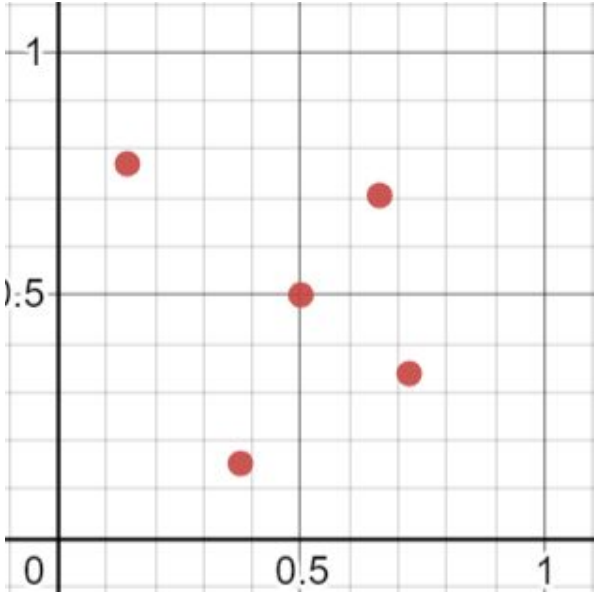
$$\hat{f}(x, y, t) = (\hat{p}, \hat{u}, \hat{v})$$

New model:

$$\hat{f}(0.75, 0.35, x, y, t) = (\hat{p}, \hat{u}, \hat{v})$$



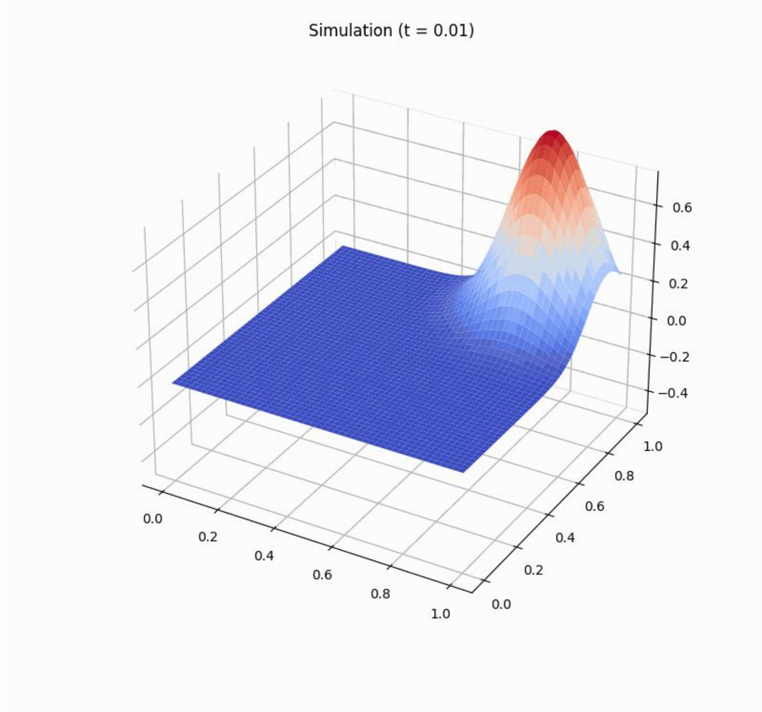
Training Data



Combined 5 simulations, each with different source locations for dataset

A Preliminary Result

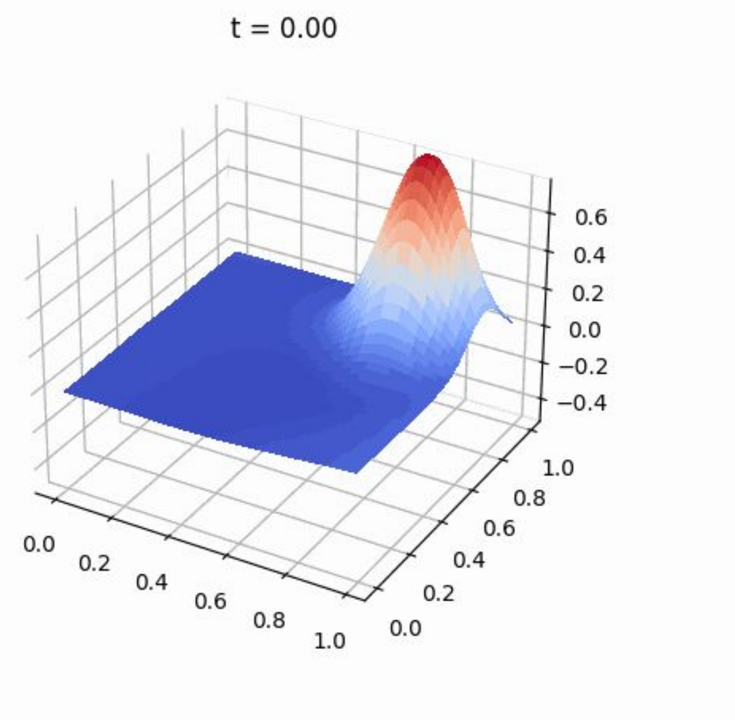
Physics-based Simulation



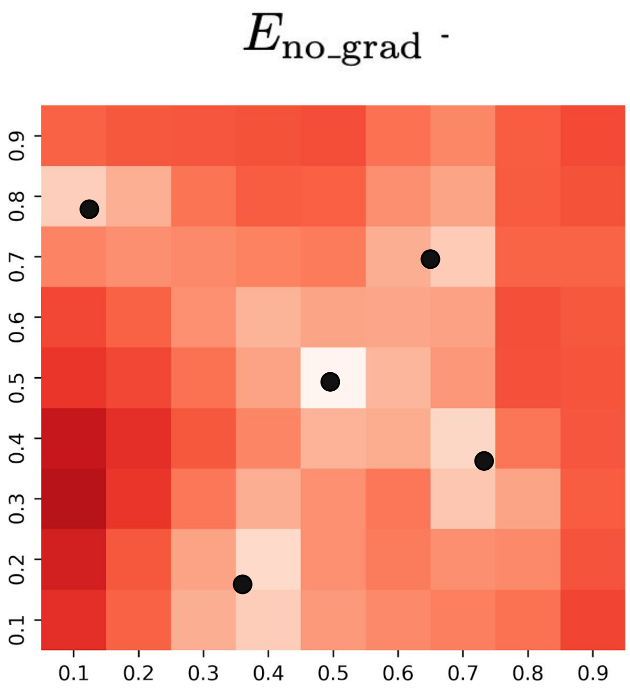
vs.

Source:
(0.8, 0.9)

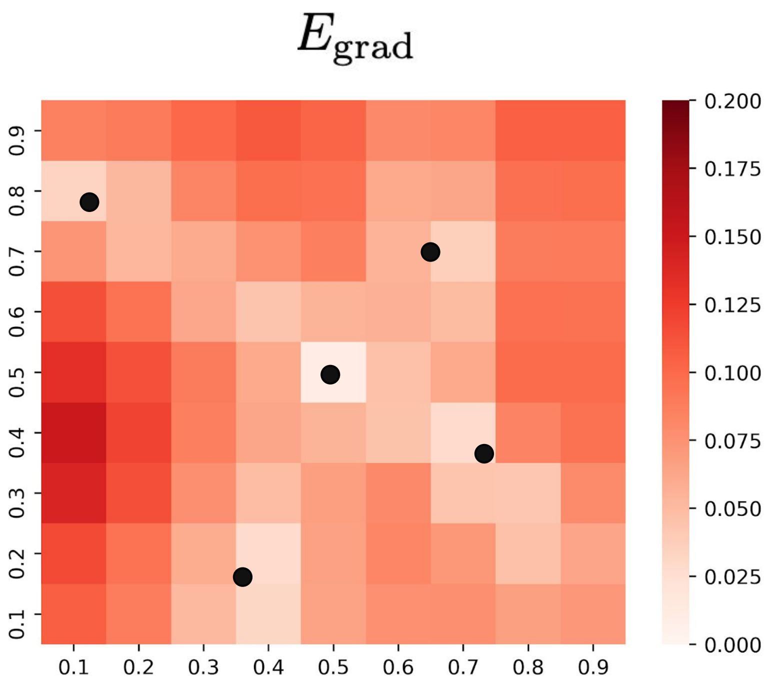
NN Model



Comparing Error



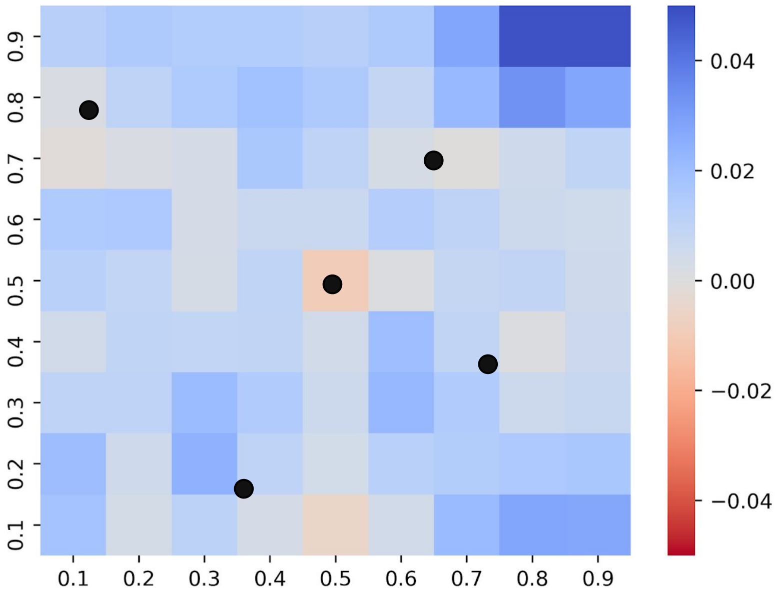
Avg. error $= 0.89$



Avg. error $= 0.77$

Comparing Error

$$E_{\text{no\_grad}} - E_{\text{grad}}$$



- Blue - gradient regularizer lowers error
- Red - gradient regularizer does not lower error

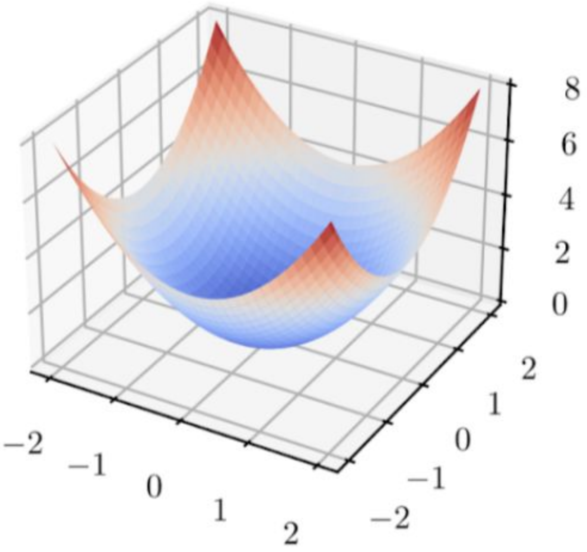
A simple question

- Problem:
  - Even PINNs can't extrapolate beyond extrapolation region
    - Need collocation points

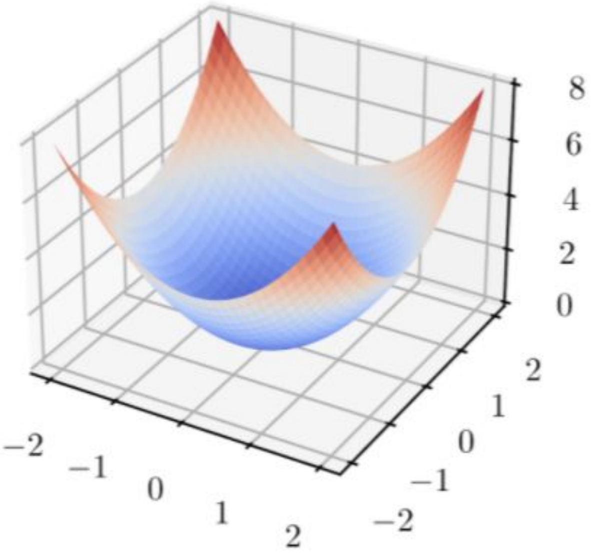
- Question:
  - Is it possible for a neural network to extrapolate *indefinitely*?
  - If not, can we predict which regions a model fails?

Paraboloid Revisited

Paraboloid: $f(x, y) = x^2 + y^2$



PINN: $\hat{f}(x, y)$

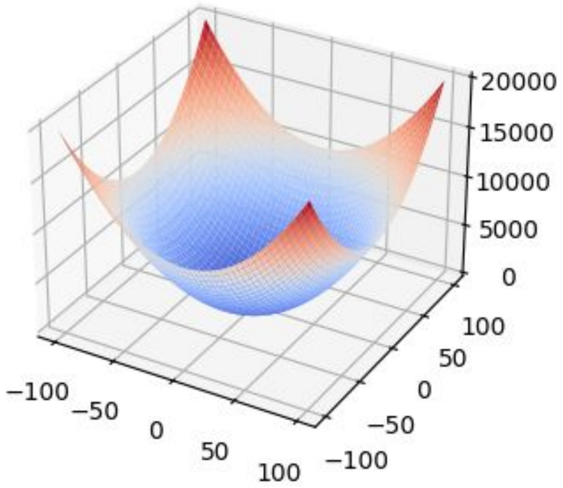


Taking a bird's eye view...

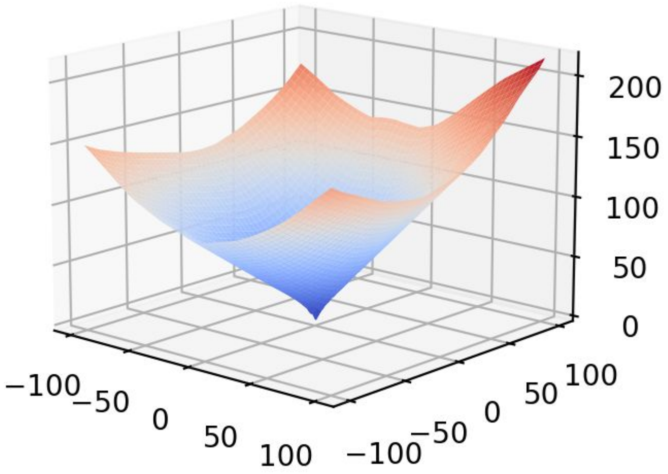
- NNs failure to extrapolate is <u>systematic</u>
- NNs <u>tune</u> to their training region

- Large scale → reduce to <u>trivial</u> function
  - Determined by activation

<u>Paraboloid:</u> $f(x, y) = x^2 + y^2$



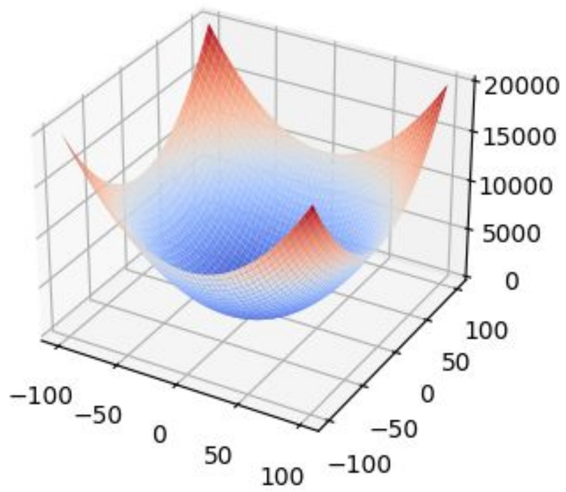
<u>PINN:</u> $\hat{f}(x, y) \approx |x| + |y|$ (using elu)



Taking a bird's eye view...

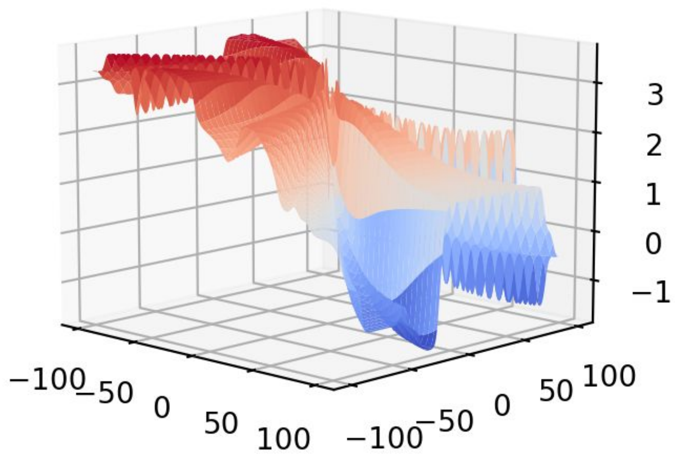
- NNs failure to extrapolate is systematic
- NNs tune to their training region

Paraboloid: $f(x, y) = x^2 + y^2$



- Large scale → reduce to trivial function
  - Determined by activation

PINN: $\hat{f}(x, y) = O(1)$ (using tanh)



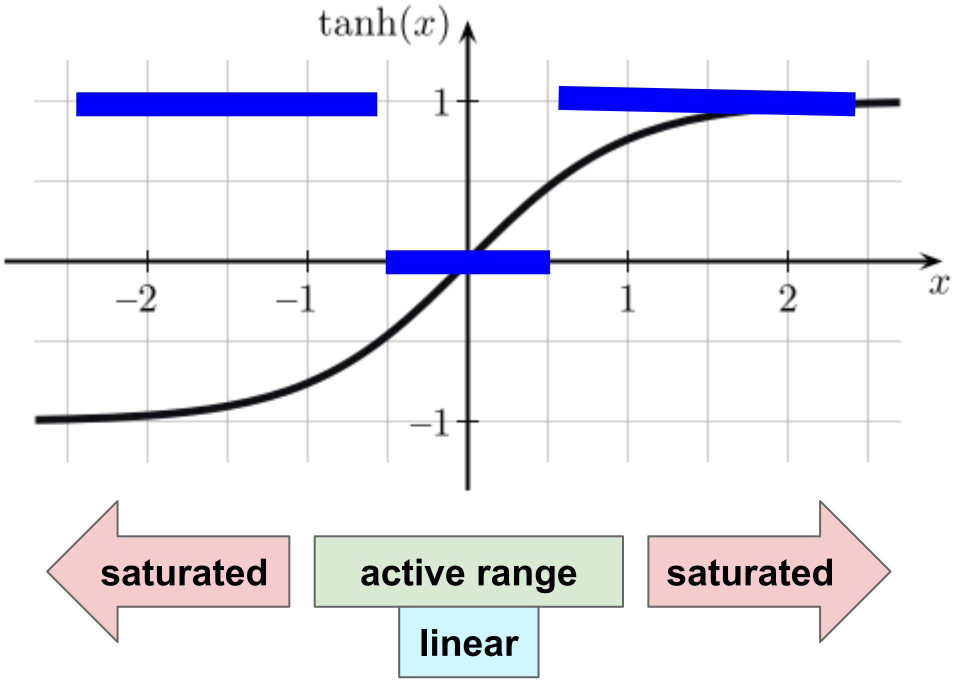
What causes NNs to reduce?

- Neurons <u>“saturate”</u> from large input
  - End behavior
- Neurons <u>tune</u> their “active range” to training region

- **<u>Key Idea:</u>**
  Modeling a complex target requires unsaturated neurons

- **<u>Hypothesis</u>**:
  NN saturated → NN will extrapolate poorly

- We develop a <u>saturation measure</u>



Saturation across domain: Paraboloid

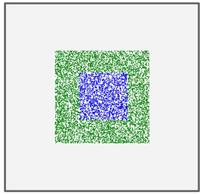
- Saturation of first layer
  - On [-4, 4] square
  - Average of 5 runs

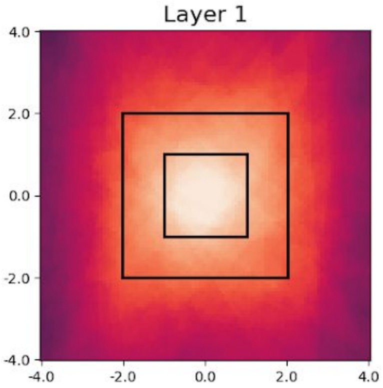
PINN

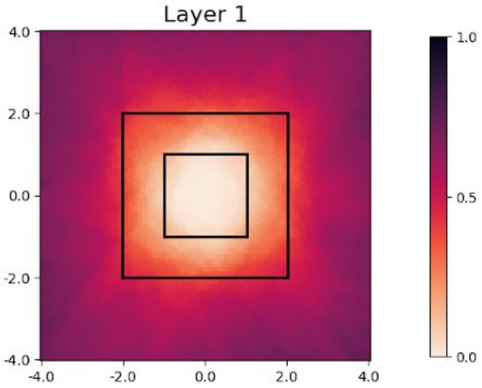
Baseline

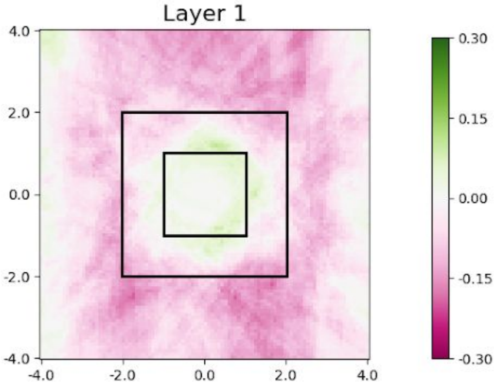
Difference

(training data)



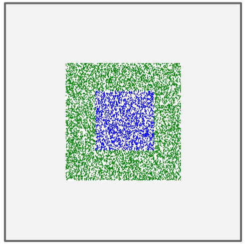






Saturation across domain: Paraboloid

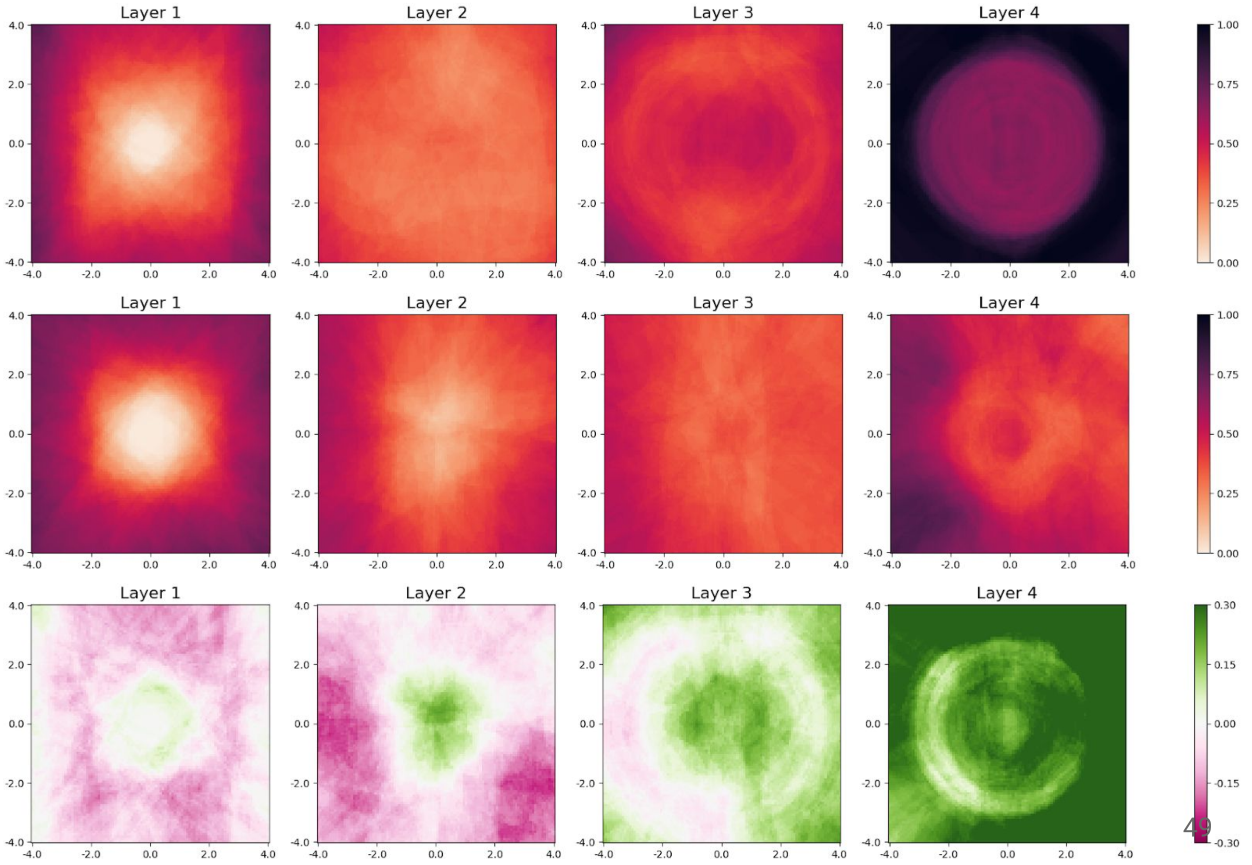
(training data)



PINN

Baseline

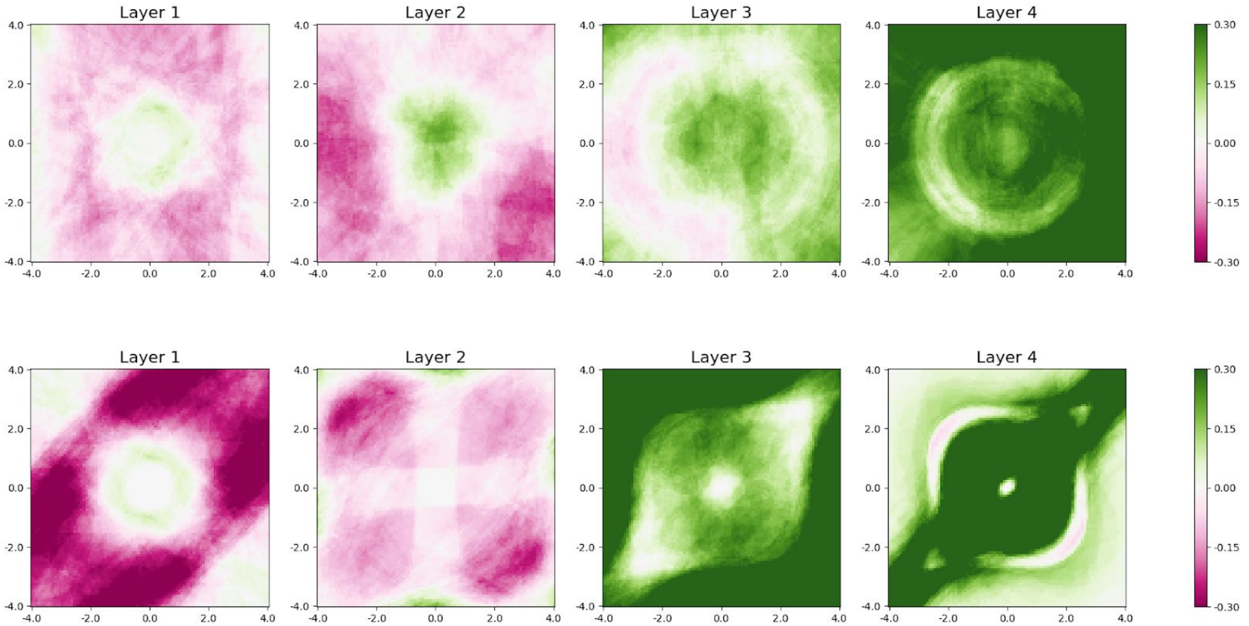
Difference



Saturation: Parabola vs Cubic

Paraboloid Difference

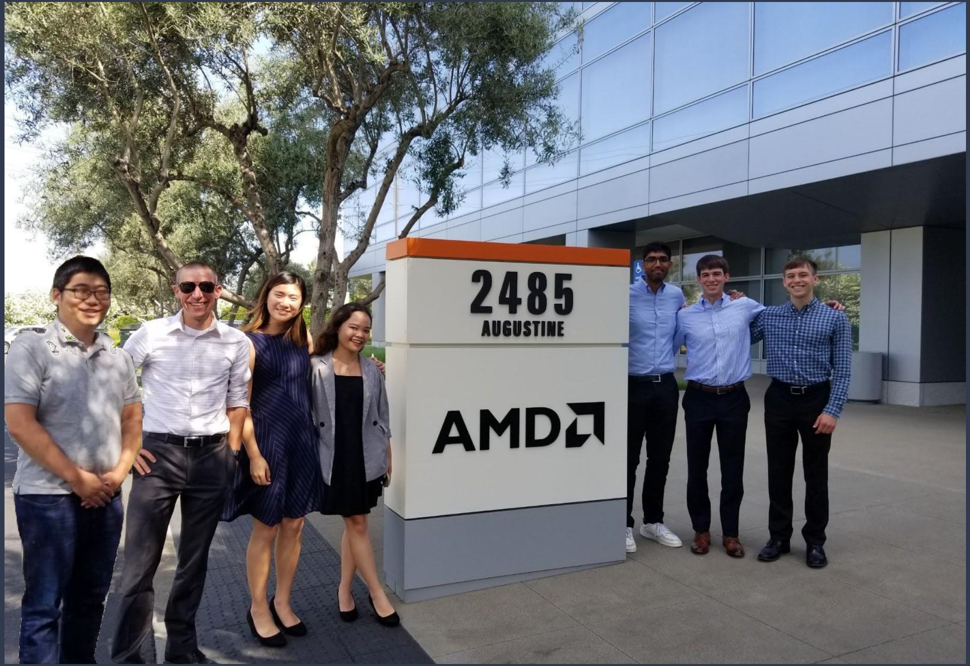
Cubic Difference



Conclusion

- Modeled paraboloid with higher extrapolation accuracy than pure interpolation
- Successfully extrapolated wave equation using PINNs
- Created model that predicts behavior of wave with different source locations
- Developed metric for measuring saturation of neuron weights

Thank you so much to Laurent, Kyung, the whole AMD team, and IPAM!



Questions?

Physical Constraints

$$f(x,y) = c_1x^2 + c_2y^2$$

Second Order Partials

$$f_{xx} = 2c_1$$

$$f_{yy} = 2c_2$$

$$f_{xy} = 0 = f_{yx}$$

$$\Longrightarrow$$

Second Order Regularizer

$$\begin{aligned} E_{\text{second}}\left(\hat{f}, \mathbf{x}\right) &= \text{Var}\left(\hat{f}_{xx}(\mathbf{x})\right) + \text{Var}\left(\hat{f}_{yy}(\mathbf{x})\right) \\ &+ 2\text{Var}\left(\hat{f}_{xy}(\mathbf{x})\right) \\ &\approx 0 \end{aligned}$$

First Order Partials

$$f_x = 2c_1x$$

$$f_y = 2c_2y$$

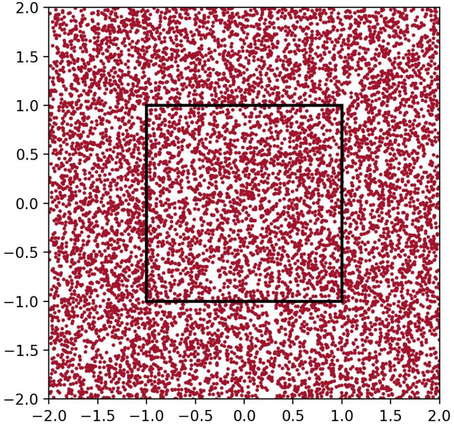
$$\Longrightarrow$$

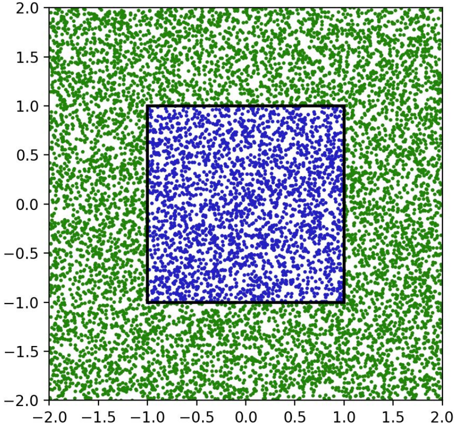
First Order Regularizer

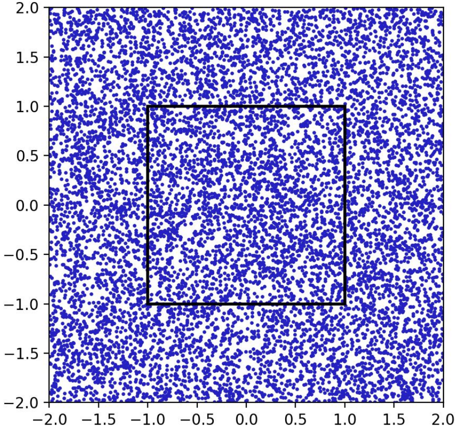
$$\begin{aligned} E_{\text{first}}\left(\hat{f}, \mathbf{x}\right) &= \text{Var}\left(\frac{\hat{f}_x(\mathbf{x})}{x}\right) + \text{Var}\left(\frac{\hat{f}_y(\mathbf{x})}{y}\right) \\ &\approx 0 \end{aligned}$$

Further Results

- $\mathcal{L} = \text{MSE}$ (red)
- $\mathcal{L} = \text{MSE} + \mathbf{E}_{phys}$ (blue)
- $\mathcal{L} = \mathbf{E}_{phys}$ (green)



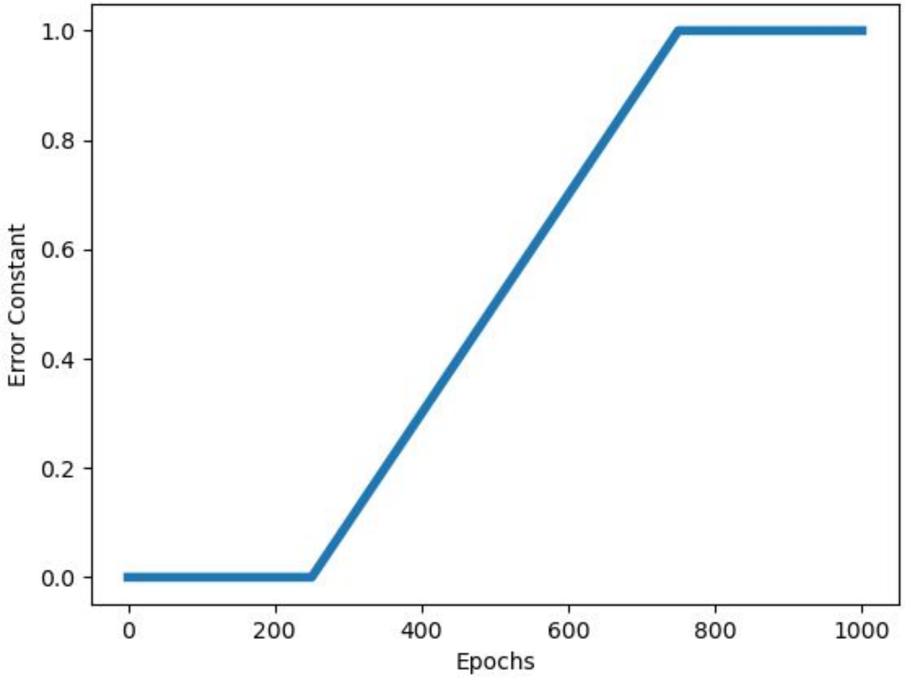




	Pure Interpolation (10,000 points with no physics information)	2nd Order Regularizer (2500 labeled Int, 7500 collocation Ext)	Interpolation + 2nd Order Regularizer (10,000 points Int+Ext)
Interpolation Region Error Avg*	1.79E-3	**1.36E-3**	5.36E-4
Extrapolation Region Error Avg*	3.02E-3	**2.53E-3**	9.40E-4

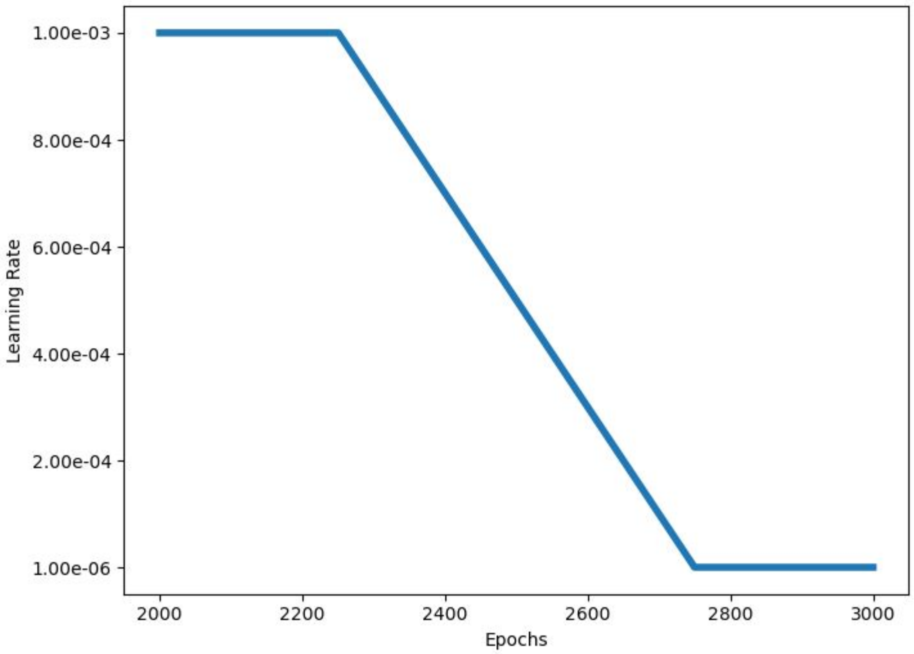
*RMSE averaged across 10 trials

Gradual Loss Change



$$\mathcal{L} = MSE + \lambda_{reg} E_{reg}$$

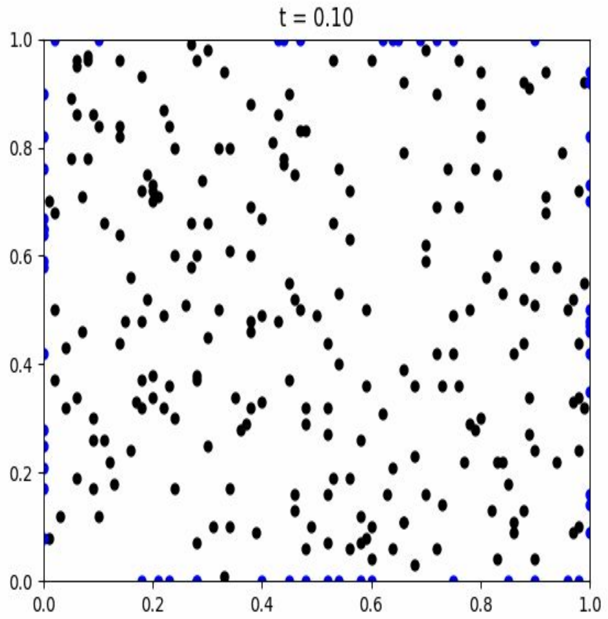
Learning Rate Change



$$w_{k+1} = w_k - \alpha \nabla \hat{f}(w_k)$$

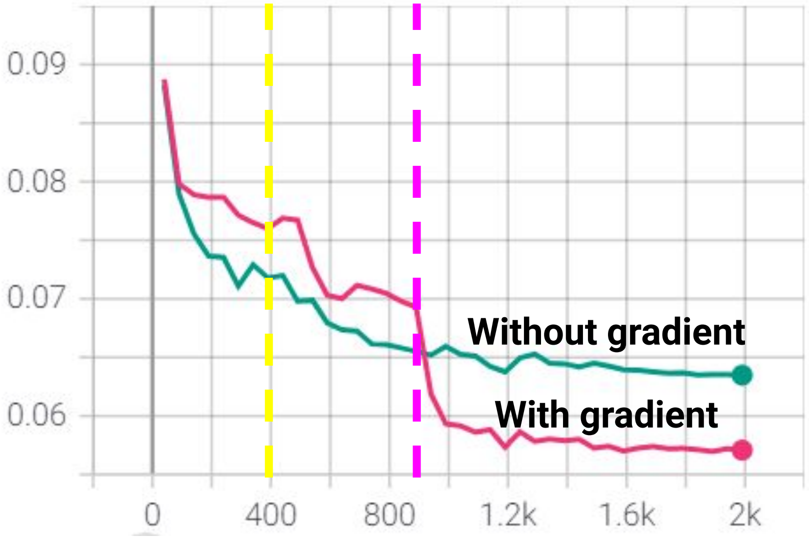
Data Sampling

- Various parameters regarding data preprocessing
  - Interior vs. Boundary
  - Interpolated vs. Extrapolated
  - Randomly vs. Uniformly
- Sample again for test points



Adding More Boundary Points

Error/interpolation error (t <= 1)



Epochs

Gradient regularizers:

- Boundary added at 400
- First order added at 900

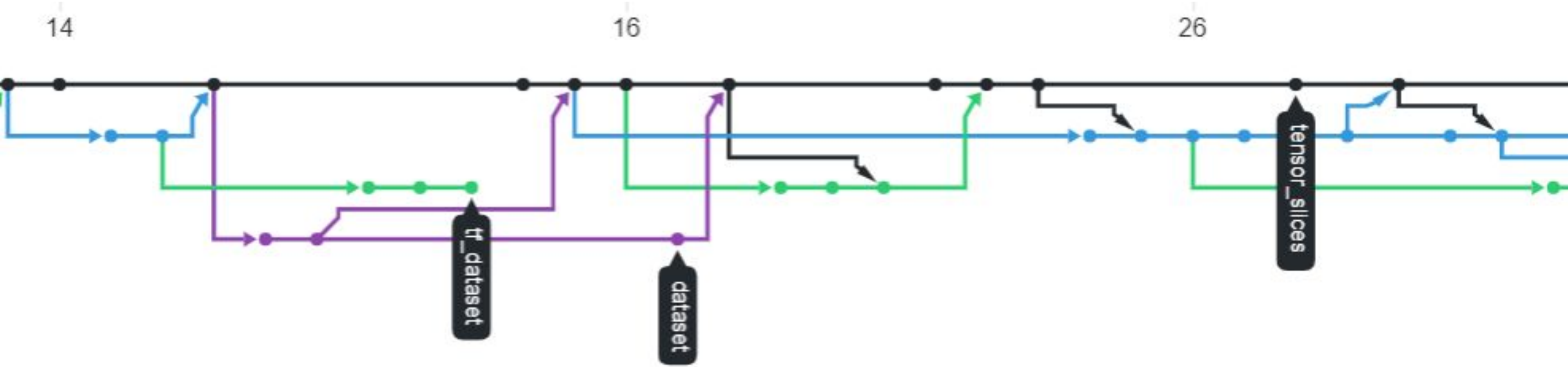
$$\mathcal{L} = \text{MSE} + \lambda_{\text{bound}} E_{\text{bound}} + \lambda_{\text{first}} E_{\text{first}}$$

Github and Config files

- configs
  - default
    - toy.yaml
    - wave.yaml
  - search
    - parabola_gridsearch
    - wave_gridsearch
  - single
    - bound_tests
    - analysis.yaml
    - baseline.yaml
    - bens_best.yaml
    - bhargavs_best.yaml
    - test.yaml
  - changes.yaml

```yaml
gradient_loss:
  - region: boundary_lr     # which data to apply to
    name: velocity_lr       # id of gradient regularizer
    weight: [0.0, 0.03]     # start_weight, end_weight
    schedule: [200, 500]    # start_epoch,  end_epoch
  - region: boundary_ud
    name: velocity_ud
    weight: [0.0, 0.03]
    schedule: [200, 500]
  - region: interior
    name: first_explicit
    weight: [0.0, 0.03]
    schedule: [600, 1000]
  - region: interior
    name: second_explicit
    weight: [0.0, 0.01]
    schedule: [1100, 1400]
```

Code Organization/Collaboration



Coding process:

- Small changes can be made on individual machine
- Large changes are made in a branch
- All merged into main branch

Experiment Management

- Config files
  - *many* levers to adjust

```yaml
##############
#  Training  #
##############
device: gpu
trials: 1
seed: 0
epochs: 300
batch_size: 256

lr: 1.0e-4
lr_scheduler: true
lr_scheduler_type: piecewise_linear
lr_scheduler_params: [1.0e-3, 1.0e-6, 100, 200]

gd_noise: 0.0
from_tensor_slices: true
shuffle: true
```

```yaml
#############
#  Dataset  #
#############
source: synthetic
target: parabola
target_coefficients: [1.0, 1.0]

corners: false
dataset: [2500, 7500, 0, 1.0, 0.0]
noise: 0.0

###########
#  Model  #
###########
activation: swish
dropout_rates: 0.0
layers: [2, 30, 30, 30, 30, 1]
```

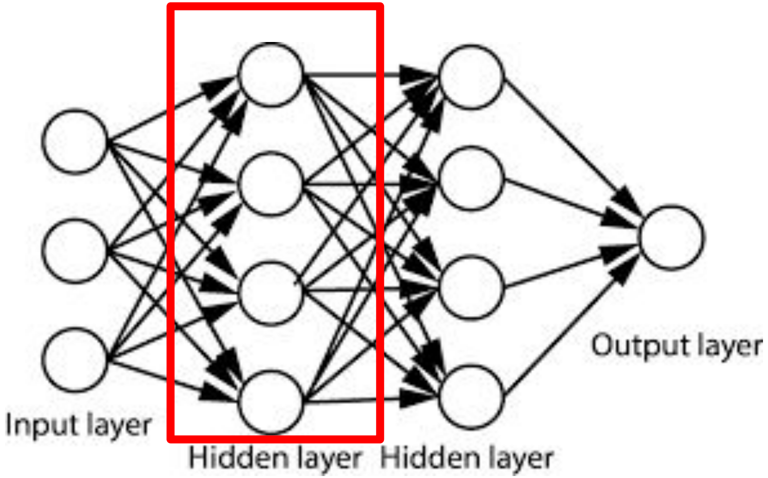
```yaml
##################
#  Regularizers  #
##################
regularizer: none
reg_const: 0.1
gradient_loss:
  - region: all               # which data to apply to
    name: second              # id of gradient regularizer
    weight: 1                 # start_weight, end_weight
grad_reg_const: 1
loss_schedulerizer: false
# loss_schedulerizer_params: [2000, 2400] #[Begin adding

#############
#  Logging  #
#############
debug: false
output_dir: null   # overridden in main.py
output_root: output/toy
plots: [extrapolation, data-distribution, tensorboard]
saves: [model]
tb_error_timestep: 20
tb_loss_timestep: 5
```

Measuring saturation (cont.)

- **<u>Goal:</u>** Quantify saturation of a layer
  l with input **x**

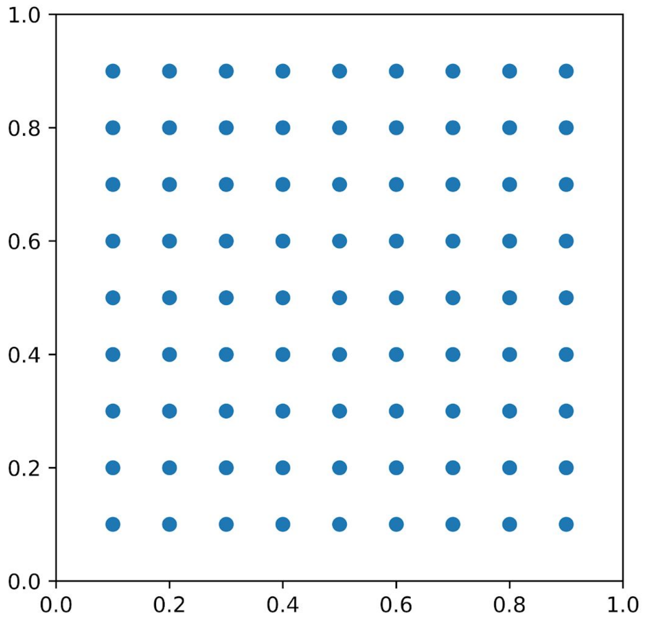
- Saturation of layer:

$$\mu_l = \frac{1}{\#\ \text{nodes}} \sum_{\substack{\text{node } n \\ \text{in layer } l}} \mu_n$$

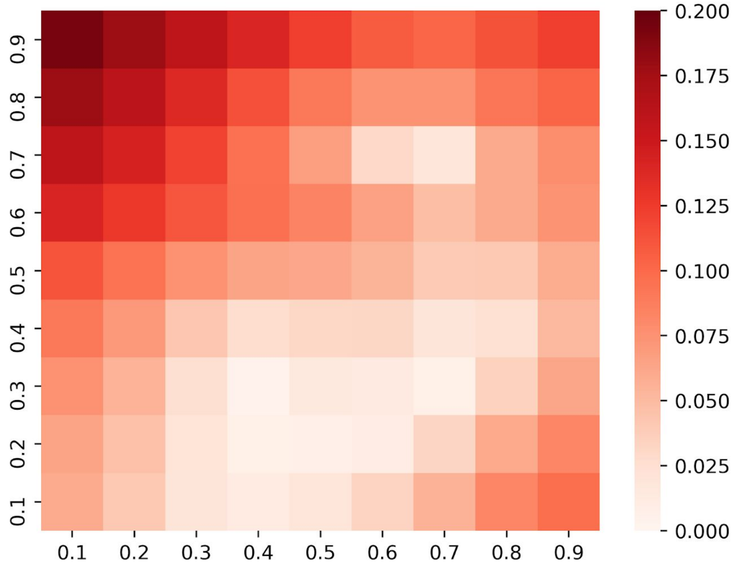


Error Analysis

Test set sources



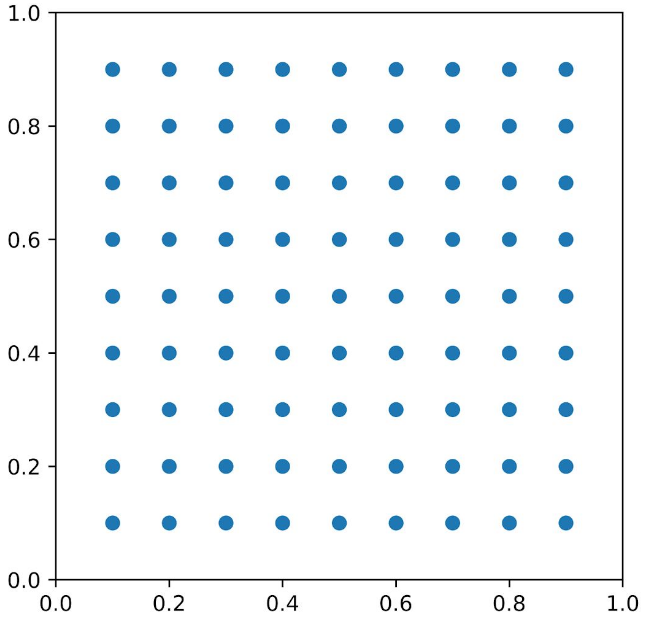
Heatmap of error on Dataset 2 (random)



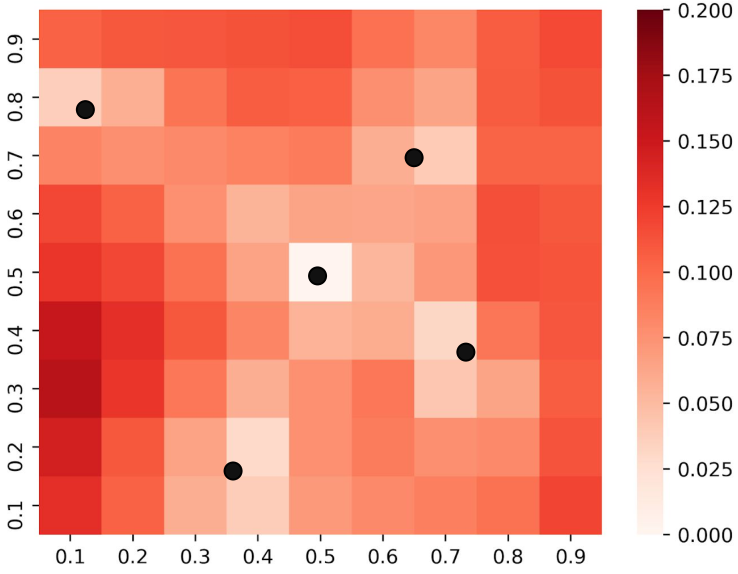
Error Analysis

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(x_s, y_s, \mathbf{x}^i) - \hat{f}(x_s, y_s, \mathbf{x}^i))^2}$$

Test set sources

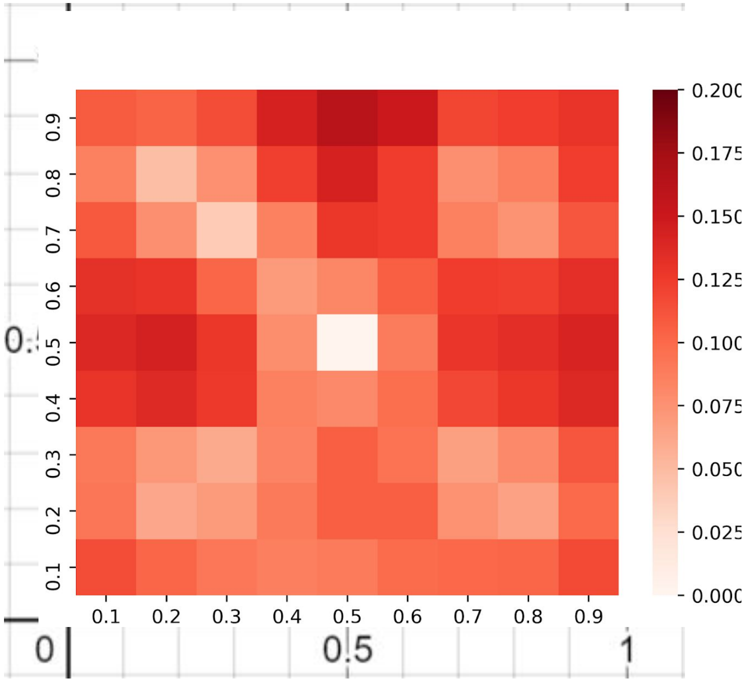


Heatmap of error

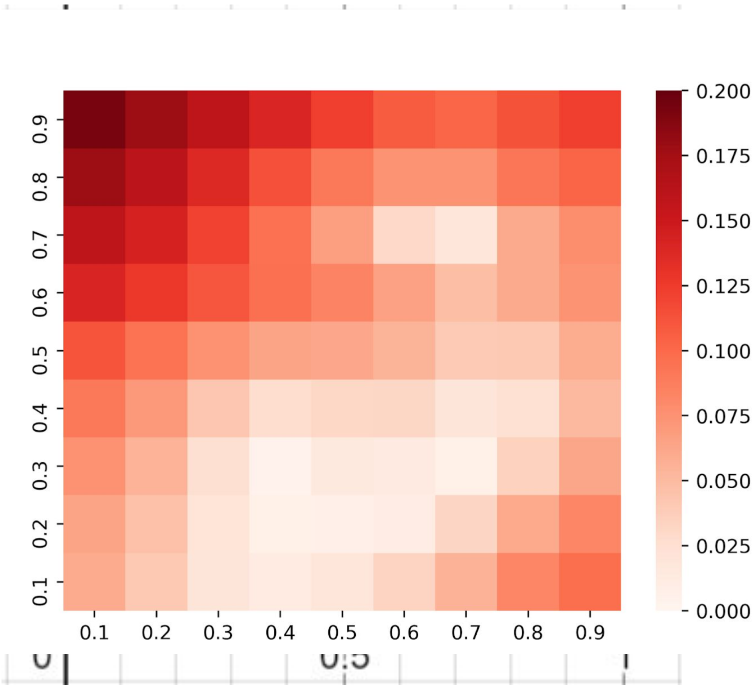


Different Training Data

Uniform source points (Dataset 1)



Random source points (Dataset 2)



Saturation across domain: Paraboloid

Gradient Regularized:

$$\mu_{\mathrm{GR}}(\mathbf{x})$$

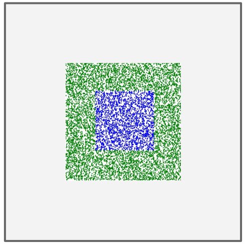
Baseline:

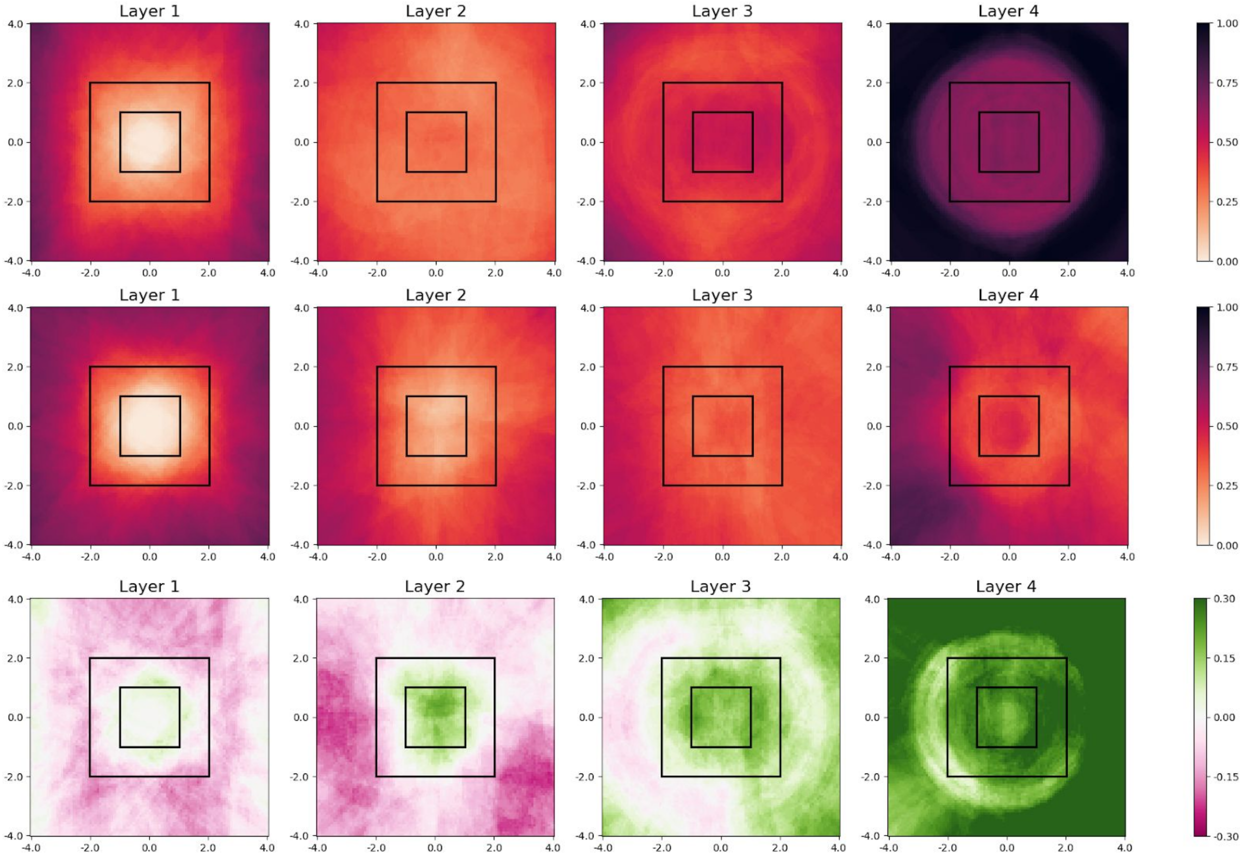
$$\mu_{\mathrm{BL}}(\mathbf{x})$$

Difference:

$$\mu_{\mathrm{GR}}(\mathbf{x}) - \mu_{\mathrm{BL}}(\mathbf{x})$$

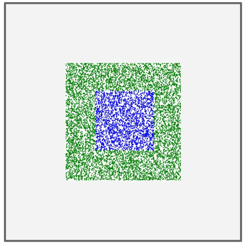
(training data)





Saturation across domain: Cubic

(training data)



<u>Gradient Regularized:</u>

$\mu_{\mathrm{GR}}(\mathbf{x})$

<u>Baseline:</u>

$\mu_{\mathrm{BL}}(\mathbf{x})$

<u>Difference:</u>

$\mu_{\mathrm{GR}}(\mathbf{x}) - \mu_{\mathrm{BL}}(\mathbf{x})$