Neural Network Approaches to Conservative-to-Primitive Inversion in Relativistic Hydrodynamics

Kaydo Alders

Supervisors: Philipp Mösta, Swapnil Shankar

June 23, 2023

University of Amsterdam
Anton Pannekoek Institute for Astronomy

Acknowledgement

First of all... many thanks to

- Philipp Möstra and Swapnil Shankar for answering all my questions!
- The MMAAMS group for listening to all my troubles!
- Antonia Rowlinson for organizing the bachelor talks!
- My family, relatives, friends and all others for watching online!
- And whoever I may have forgotten to list (sorry)!

Contents

Contents

Contents

Contents

Contents

Contents

Introduction—The Importance of GRMHD Simulations

- The role of General Relativistic Magneto-Hydrodynamics (GRMHD) simulations is to study astrophysical phenomena like core-collapse supernovae and binary neutron star mergers.
- These simulations enable us to study gravitational waves, ejected materials, and remnants of compact-object mergers.
- Complex and demanding calculations require speed and accuracy for meaningful results.

Introduction—The Importance of GRMHD Simulations

- The role of General Relativistic Magneto-Hydrodynamics (GRMHD) simulations is to study astrophysical phenomena like core-collapse supernovae and binary neutron star mergers.
- These simulations enable us to study gravitational waves, ejected materials, and remnants of compact-object mergers.
- Complex and demanding calculations require speed and accuracy for meaningful results.

Introduction—The Importance of GRMHD Simulations

- The role of General Relativistic Magneto-Hydrodynamics (GRMHD) simulations is to study astrophysical phenomena like core-collapse supernovae and binary neutron star mergers.
- These simulations enable us to study gravitational waves, ejected materials, and remnants of compact-object mergers.
- Complex and demanding calculations require speed and accuracy for meaningful results.

GRaM-X—Enhancing Simulations Efficiency

- GRaM-X, a state-of-the-art code built on the Cactus computational framework, incorporates enhancements like GPU utilization to boost simulations' computational efficiency.
- Part of the Einstein Toolkit, GRaM-X facilitates numerical simulations and analysis of astrophysical phenomena in the context of general relativity.

GRaM-X—Enhancing Simulations Efficiency

- GRaM-X, a state-of-the-art code built on the Cactus computational framework, incorporates enhancements like GPU utilization to boost simulations' computational efficiency.
- Part of the Einstein Toolkit, GRaM-X facilitates numerical simulations and analysis of astrophysical phenomena in the context of general relativity.

The Project Focus—Conserved-to-Primitive Inversion

- Grid-based magnetohydrodynamics simulation involves a set of primitive and conserved variables.
- Our stellar object is modelled as a fluid.
- The conversion from conserved variables is computationally expensive.
- Our project focuses on enhancing the performance of this inversion (con2prim).

The Project Focus—Conserved-to-Primitive Inversion

- Grid-based magnetohydrodynamics simulation involves a set of primitive and conserved variables.
- Our stellar object is modelled as a fluid.
- The conversion from conserved variables is computationally expensive.
- Our project focuses on enhancing the performance of this inversion (con2prim).

The Project Focus—Conserved-to-Primitive Inversion

- Grid-based magnetohydrodynamics simulation involves a set of primitive and conserved variables.
- Our stellar object is modelled as a fluid.
- The conversion from conserved variables is computationally expensive.
- Our project focuses on enhancing the performance of this inversion (con2prim).

The Project Focus—Conserved-to-Primitive Inversion

- Grid-based magnetohydrodynamics simulation involves a set of primitive and conserved variables.
- Our stellar object is modelled as a fluid.
- The conversion from conserved variables is computationally expensive.
- Our project focuses on enhancing the performance of this inversion (con2prim).

Overview

Research Objective

- Implement a supervised artificial neural network ANN to replace con2prim root-finding algorithms.
- Use Python and the PyTorch framework for initial implementation.
- Split the project into two parts: special relativistic case (SRHD) and general relativistic case (GRMHD).

Research Objective

- Implement a supervised artificial neural network ANN to replace con2prim root-finding algorithms.
- Use Python and the PyTorch framework for initial implementation.
- Split the project into two parts: special relativistic case (SRHD) and general relativistic case (GRMHD).

Research Objective

- Implement a supervised artificial neural network ANN to replace con2prim root-finding algorithms.
- Use Python and the PyTorch framework for initial implementation.
- Split the project into two parts: special relativistic case (SRHD) and general relativistic case (GRMHD).

Research Objective—Part I (Special Relativistic Case)

- Replicate ANN from Dieselhorst et al.'s work with the same hyperparameters.
- Aim for the same order of accuracy as in Dieselhorst et al.'s work.
- Experiment with different hyperparameters, and other settings.

Research Objective—Part I (Special Relativistic Case)

- Replicate ANN from Dieselhorst et al.'s work with the same hyperparameters.
- Aim for the same order of accuracy as in Dieselhorst et al.'s work.
- Experiment with different hyperparameters, and other settings.

Research Objective—Part I (Special Relativistic Case)

- Replicate ANN from Dieselhorst et al.'s work with the same hyperparameters.
- Aim for the same order of accuracy as in Dieselhorst et al.'s work.
- Experiment with different hyperparameters, and other settings.

Research Objective—Part II (General GRMHD Case)

- Implement ANNs for GRMHD, using insights gained from SRHD.
- Demonstrate that the ANN can reduce computational time by at least an order of magnitude.
- Show the potential to integrate the ANN into the GRaM-X framework by porting to C++.

Research Objective—Part II (General GRMHD Case)

- Implement ANNs for GRMHD, using insights gained from SRHD.
- Demonstrate that the ANN can reduce computational time by at least an order of magnitude.
- Show the potential to integrate the ANN into the GRaM-X framework by porting to C++.

Research Objective—Part II (General GRMHD Case)

- Implement ANNs for GRMHD, using insights gained from SRHD.
- Demonstrate that the ANN can reduce computational time by at least an order of magnitude.
- Show the potential to integrate the ANN into the GRaM-X framework by porting to C++.

Overview

Overview

SRHD—Conservative and primitive variables

- Considering first SRHD.
- Explicitly, conservative variables $\mathbf{U}$ are given by

$$\mathbf{U} = (D, S^1, S^2, S^3, \tau) \tag{1}$$

- Here, the conservative variables are: rest-mass density D, three components of momentum S^i, and energy density τ.
- Primitive variables are i.a. used to calculated the fluxes, and are given by

$$\mathbf{P} = (\rho, v^i, \epsilon, p) \tag{2}$$

- These include the proper rest-mass density ρ, 3-velocity v^i, specific internal energy ϵ, and pressure p of the fluid.

SRHD—Conservative and primitive variables

- Considering first SRHD.
- Explicitly, conservative variables $\mathbf{U}$ are given by

$$\mathbf{U} = (D, S^1, S^2, S^3, \tau) \tag{1}$$

- Here, the conservative variables are: rest-mass density D, three components of momentum S^i, and energy density τ.
- Primitive variables are i.a. used to calculated the fluxes, and are given by

$$\mathbf{P} = (\rho, v^i, \epsilon, p) \tag{2}$$

- These include the proper rest-mass density ρ, 3-velocity v^i, specific internal energy ϵ, and pressure p of the fluid.

SRHD—Conservative and primitive variables

- Considering first SRHD.
- Explicitly, conservative variables **U** are given by

$$\mathbf{U} = (D, S^1, S^2, S^3, \tau) \tag{1}$$

- Here, the conservative variables are: rest-mass density D, three components of momentum S^i, and energy density τ.
- Primitive variables are i.a. used to calculated the fluxes, and are given by

$$\mathbf{P} = (\rho, v^i, \epsilon, p) \tag{2}$$

- These include the proper rest-mass density ρ, 3-velocity v^i, specific internal energy ϵ, and pressure p of the fluid.

SRHD—Conservative and primitive variables

- Considering first SRHD.
- Explicitly, conservative variables $\mathbf{U}$ are given by

$$\mathbf{U} = (D, S^1, S^2, S^3, \tau) \qquad (1)$$

- Here, the conservative variables are: rest-mass density D, three components of momentum S^i, and energy density τ.
- Primitive variables are i.a. used to calculated the fluxes, and are given by

$$\mathbf{P} = (\rho, v^i, \epsilon, p) \qquad (2)$$

- These include the proper rest-mass density ρ, 3-velocity v^i, specific internal energy ϵ, and pressure p of the fluid.

SRHD—Conservative and primitive variables

- Considering first SRHD.
- Explicitly, conservative variables **U** are given by

$$\mathbf{U} = (D, S^1, S^2, S^3, \tau) \tag{1}$$

- Here, the conservative variables are: rest-mass density D, three components of momentum S^i, and energy density τ.
- Primitive variables are i.a. used to calculated the fluxes, and are given by

$$\mathbf{P} = \left(\rho, v^i, \epsilon, p\right) \tag{2}$$

- These include the proper rest-mass density ρ, 3-velocity v^i, specific internal energy ϵ, and pressure p of the fluid.

SRHD—Evolving a stellar object through time

- This evolution is achieved through the equation

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^i}{\partial x^i} = \mathbf{S} \tag{3}$$

- This equation is a conservation law relating changes in conserved quantities to their fluxes $\mathbf{F^i}$ and sources $\mathbf{S}$.
- Such a formulation is foundational in physics, much like the Divergence Theorem in Calculus.

SRHD—Evolving a stellar object through time

- This evolution is achieved through the equation

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^i}{\partial x^i} = \mathbf{S} \tag{3}$$

- This equation is a conservation law relating changes in conserved quantities to their fluxes $\mathbf{F^i}$ and sources $\mathbf{S}$.
- Such a formulation is foundational in physics, much like the Divergence Theorem in Calculus.

SRHD—Evolving a stellar object through time

- This evolution is achieved through the equation

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^i}{\partial x^i} = \mathbf{S} \qquad (3)$$

- This equation is a conservation law relating changes in conserved quantities to their fluxes $\mathbf{F^i}$ and sources $\mathbf{S}$.
- Such a formulation is foundational in physics, much like the Divergence Theorem in Calculus.

SRHD—Evolving a stellar object through time

- This evolution is achieved through the equation

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^i}{\partial x^i} = \mathbf{S} \tag{3}$$

- This equation is a conservation law relating changes in conserved quantities to their fluxes $\mathbf{F^i}$ and sources $\mathbf{S}$.
- Such a formulation is foundational in physics, much like the Divergence Theorem in Calculus.

Numerical methods—(Stellar) Evolution in a simulation

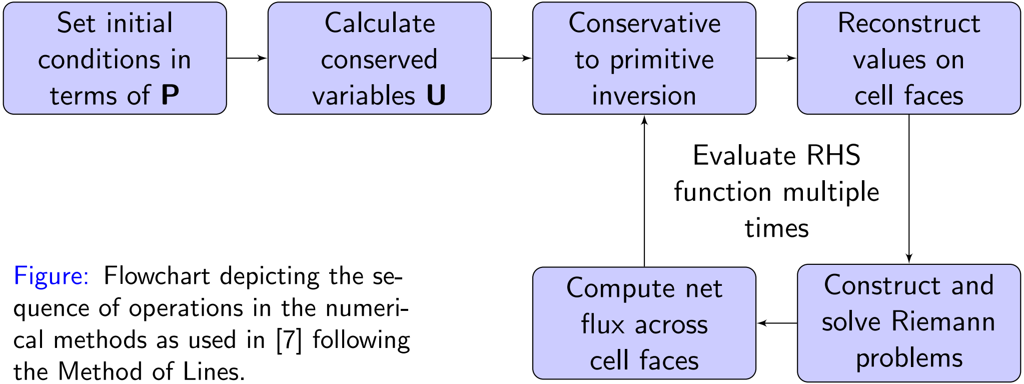


Figure: Flowchart depicting the sequence of operations in the numerical methods as used in [7] following the Method of Lines.

Numerical methods—Cell reconstruction

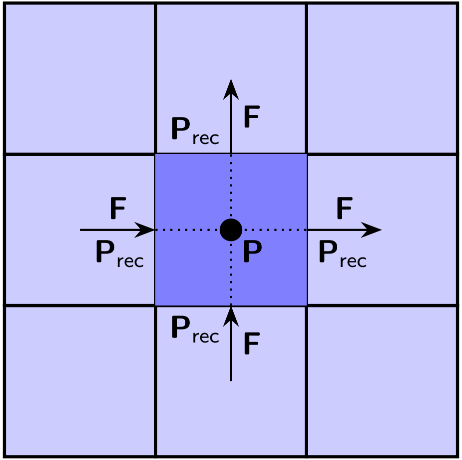


Figure: Illustration of a two-dimensional representation of a grid cell in spacetime.

Relation between cons and prims in SRHD

- The conservative variables are related to the primitive variables through these equations:

$$D = \rho W \ , \tag{4}$$

$$S^i = \rho h W^2 v^i \quad i = 1, 2, 3 \ , \tag{5}$$

$$\tau = \rho h W^2 - p - D \ , \tag{6}$$

- h - the specific enthalpy $h = 1 + \epsilon + p/\rho$
- W - the Lorentz factor given by: $W = \frac{1}{\sqrt{1 - v^i v_i}}$.
- EoS assumed to be of the form: $p = p(\rho, \epsilon) \propto \rho\epsilon$.
- Inversion $\mathbf{P}(\mathbf{U})$ is non-analytic and requires numerical approximation!
- We use the equations for D, S^i, and τ directly as input to the NNs.

Relation between cons and prims in SRHD

- The conservative variables are related to the primitive variables through these equations:

$$D = \rho W \ , \tag{4}$$

$$S^i = \rho h W^2 v^i \quad i = 1, 2, 3 \ , \tag{5}$$

$$\tau = \rho h W^2 - p - D \ , \tag{6}$$

- h - the specific enthalpy $h = 1 + \epsilon + p/\rho$
- W - the Lorentz factor given by: $W = \frac{1}{\sqrt{1 - v^i v_i}}$.
- EoS assumed to be of the form: $p = p(\rho, \epsilon) \propto \rho\epsilon$.
- Inversion $\mathbf{P}(\mathbf{U})$ is non-analytic and requires numerical approximation!
- We use the equations for D, S^i, and τ directly as input to the NNs.

Relation between cons and prims in SRHD

- The conservative variables are related to the primitive variables through these equations:

$$D = \rho W \ , \tag{4}$$

$$S^i = \rho h W^2 v^i \quad i = 1, 2, 3 \ , \tag{5}$$

$$\tau = \rho h W^2 - p - D \ , \tag{6}$$

- h - the specific enthalpy $h = 1 + \epsilon + p/\rho$
- W - the Lorentz factor given by: $W = \frac{1}{\sqrt{1 - v^i v_i}}$.
- EoS assumed to be of the form: $p = p(\rho, \epsilon) \propto \rho\epsilon$.
- Inversion $\mathbf{P}(\mathbf{U})$ is non-analytic and requires numerical approximation!
- We use the equations for D, S^i, and τ directly as input to the NNs.

Relation between cons and prims in SRHD

- The conservative variables are related to the primitive variables through these equations:

$$D = \rho W \ , \tag{4}$$

$$S^i = \rho h W^2 v^i \quad i = 1, 2, 3 \ , \tag{5}$$

$$\tau = \rho h W^2 - p - D \ , \tag{6}$$

- h - the specific enthalpy $h = 1 + \epsilon + p/\rho$
- W - the Lorentz factor given by: $W = \frac{1}{\sqrt{1 - v^i v_i}}$.
- EoS assumed to be of the form: $p = p(\rho, \epsilon) \propto \rho\epsilon$.
- Inversion **P(U)** is non-analytic and requires numerical approximation!
- We use the equations for D, S^i, and τ directly as input to the NNs.

Relation between cons and prims in SRHD

- The conservative variables are related to the primitive variables through these equations:

$$D = \rho W \ , \tag{4}$$

$$S^i = \rho h W^2 v^i \quad i = 1, 2, 3 \ , \tag{5}$$

$$\tau = \rho h W^2 - p - D \ , \tag{6}$$

- h - the specific enthalpy $h = 1 + \epsilon + p/\rho$
- W - the Lorentz factor given by: $W = \frac{1}{\sqrt{1 - v^i v_i}}$.
- EoS assumed to be of the form: $p = p(\rho, \epsilon) \propto \rho\epsilon$.
- Inversion $\mathbf{P}(\mathbf{U})$ is non-analytic and requires numerical approximation!
- We use the equations for D, S^i, and τ directly as input to the NNs.

Relation between cons and prims in SRHD

- The conservative variables are related to the primitive variables through these equations:

$$D = \rho W \ , \tag{4}$$

$$S^i = \rho h W^2 v^i \quad i = 1, 2, 3 \ , \tag{5}$$

$$\tau = \rho h W^2 - p - D \ , \tag{6}$$

- h - the specific enthalpy $h = 1 + \epsilon + p/\rho$
- W - the Lorentz factor given by: $W = \frac{1}{\sqrt{1 - v^i v_i}}$.
- EoS assumed to be of the form: $p = p(\rho, \epsilon) \propto \rho\epsilon$.
- Inversion $\mathbf{P}(\mathbf{U})$ is non-analytic and requires numerical approximation!
- We use the equations for D, S^i, and τ directly as input to the NNs.

- The Valencia formulation continues to hold in GRMHD:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^i}{\partial x^i} = \mathbf{S} \ , \tag{3}$$

- However, the underlying physics now includes the influence of gravity and the magnetic field of the stellar object.

General Relativistic Magnetohydrodynamics (GRMHD)

- The Valencia formulation continues to hold in GRMHD:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^i}{\partial x^i} = \mathbf{S} \ , \tag{3}$$

- However, the underlying physics now includes the influence of gravity and the magnetic field of the stellar object.

GRMHD—Conservative variables in GRMHD

- The vector of conserved variables is redefined:

$$\mathbf{U} = \left(D, S_j, \tau, \mathscr{B}^k\right) \tag{7}$$

- Now includes the conserved magnetic field:

$$\mathscr{B}^k = \sqrt{\gamma} B^k \ , \tag{8}$$

- And the determinant of the three-metric γ_{ij}.

GRMHD—Conservative variables in GRMHD

- The vector of conserved variables is redefined:

$$\mathbf{U} = \left(D, S_j, \tau, \mathscr{B}^k\right) \tag{7}$$

- Now includes the conserved magnetic field:

$$\mathscr{B}^k = \sqrt{\gamma} B^k \ , \tag{8}$$

- And the determinant of the three-metric γ_{ij}.

GRMHD—Conservative variables in GRMHD

- The vector of conserved variables is redefined:

$$\mathbf{U} = \left(D, S_j, \tau, \mathscr{B}^k\right) \tag{7}$$

- Now includes the conserved magnetic field:

$$\mathscr{B}^k = \sqrt{\gamma} B^k \ , \tag{8}$$

- And the determinant of the three-metric γ_{ij}.

GRMHD Conservative-to-primitive Inversion Equations

- Compared to SRHD, modified and additional equations.
- The full set of relations in GRMHD is:

$$D = \sqrt{\gamma}\rho W \ , \tag{9}$$

$$S_j = \sqrt{\gamma}\left(\rho h^* W^2 v_j - \alpha b^0 b_j\right) \ , \tag{10}$$

$$\tau = \sqrt{\gamma}\left(\rho h^* W^2 - p^* - (\alpha b^0)^2\right) - D \ , \tag{11}$$

$$\mathscr{B}^k = \sqrt{\gamma} B^k \ , \quad [8]$$

- Used directly as the input to the NNs for GRMHD.

GRMHD Conservative-to-primitive Inversion Equations

- Compared to SRHD, modified and additional equations.
- The full set of relations in GRMHD is:

$$D = \sqrt{\gamma}\rho W \ , \tag{9}$$

$$S_j = \sqrt{\gamma}\left(\rho h^* W^2 v_j - \alpha b^0 b_j\right) \ , \tag{10}$$

$$\tau = \sqrt{\gamma}\left(\rho h^* W^2 - p^* - (\alpha b^0)^2\right) - D \ , \tag{11}$$

$$\mathscr{B}^k = \sqrt{\gamma} B^k \ , \quad [8]$$

- Used directly as the input to the NNs for GRMHD.

GRMHD Conservative-to-primitive Inversion Equations

- Compared to SRHD, modified and additional equations.
- The full set of relations in GRMHD is:

$$
\begin{aligned}
D &= \sqrt{\gamma}\rho W \ , && (9)\\
S_j &= \sqrt{\gamma}\left(\rho h^* W^2 v_j - \alpha b^0 b_j\right) \ , && (10)\\
\tau &= \sqrt{\gamma}\left(\rho h^* W^2 - p^* - (\alpha b^0)^2\right) - D \ , && (11)\\
\mathscr{B}^k &= \sqrt{\gamma} B^k \ , && [8]
\end{aligned}
$$

- Used directly as the input to the NNs for GRMHD.

Overview

Artificial Neural Networks in Context

- Cornerstone of Machine Learning and Artificial Intelligence (AI) [3].
- Biological neural networks
- Machine learning, a subset of AI, enables computers to learn tasks without being explicitly programmed, parsing data and making decisions based on what they have learned [5].
- An ANN is fundamentally a function mapping from inputs to outputs. It aims to learn this mapping to generalize well to unseen data [1].

Artificial Neural Networks in Context

- Cornerstone of Machine Learning and Artificial Intelligence (AI) [3].
- Biological neural networks
- Machine learning, a subset of AI, enables computers to learn tasks without being explicitly programmed, parsing data and making decisions based on what they have learned [5].
- An ANN is fundamentally a function mapping from inputs to outputs. It aims to learn this mapping to generalize well to unseen data [1].

Artificial Neural Networks in Context

- Cornerstone of Machine Learning and Artificial Intelligence (AI) [3].
- Biological neural networks
- Machine learning, a subset of AI, enables computers to learn tasks without being explicitly programmed, parsing data and making decisions based on what they have learned [5].
- An ANN is fundamentally a function mapping from inputs to outputs. It aims to learn this mapping to generalize well to unseen data [1].

Artificial Neural Networks in Context

- Cornerstone of Machine Learning and Artificial Intelligence (AI) [3].
- Biological neural networks
- Machine learning, a subset of AI, enables computers to learn tasks without being explicitly programmed, parsing data and making decisions based on what they have learned [5].
- An ANN is fundamentally a function mapping from inputs to outputs. It aims to learn this mapping to generalize well to unseen data [1].

Deep Learning and ANN Structure

- Deep Learning (DL) involves constructing and training neural networks with multiple "depth" layers, allowing the network to learn more abstract features [3].
- An ANN architecture consists of interconnected layers of artificial neurons or nodes.

Deep Learning and ANN Structure

- Deep Learning (DL) involves constructing and training neural networks with multiple "depth" layers, allowing the network to learn more abstract features [3].
- An ANN architecture consists of interconnected layers of artificial neurons or nodes.

Network architecure SRHD models

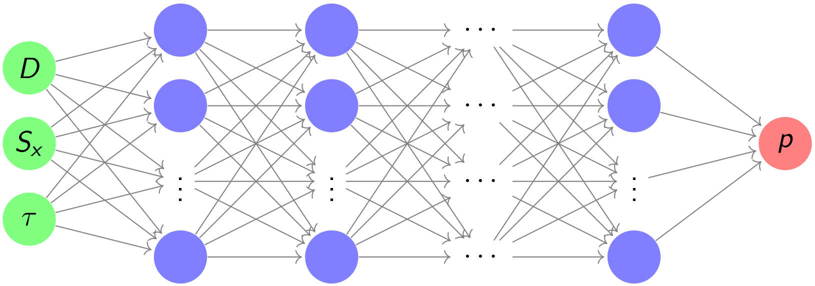


Figure: Illustration of the neural network architecture for the SRHD models, with input layer (green), hidden layers (blue) and the output layer (red)

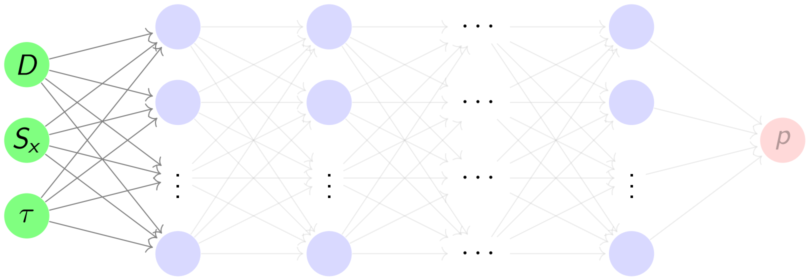


Figure: Illustration of the neural network architecture for the SRHD models, with input layer (green), hidden layers (blue) and the output layer (red)

Network Architecure SRHD models

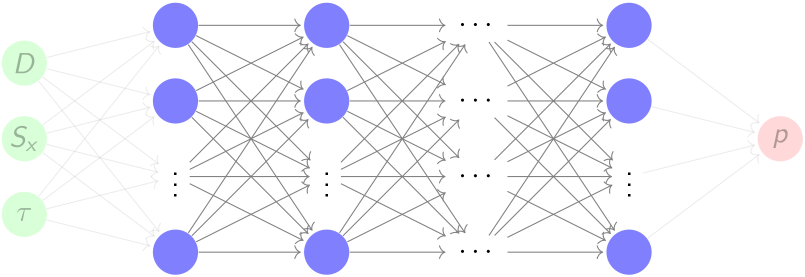


Figure: Illustration of the neural network architecture for the SRHD models, with input layer (green), hidden layers (blue) and the output layer (red)

Network Architecure SRHD models

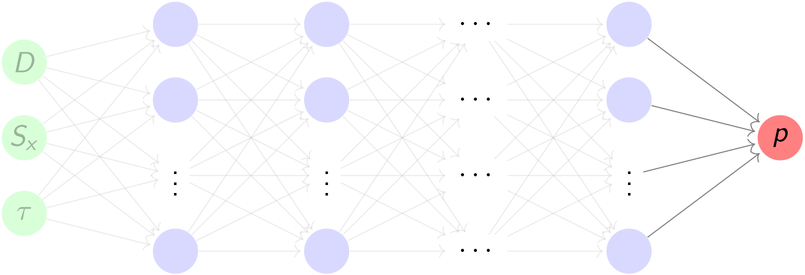


Figure: Illustration of the neural network architecture for the SRHD models, with input layer (green), hidden layers (blue) and the output layer (red)

Network Architecure GRMHD models

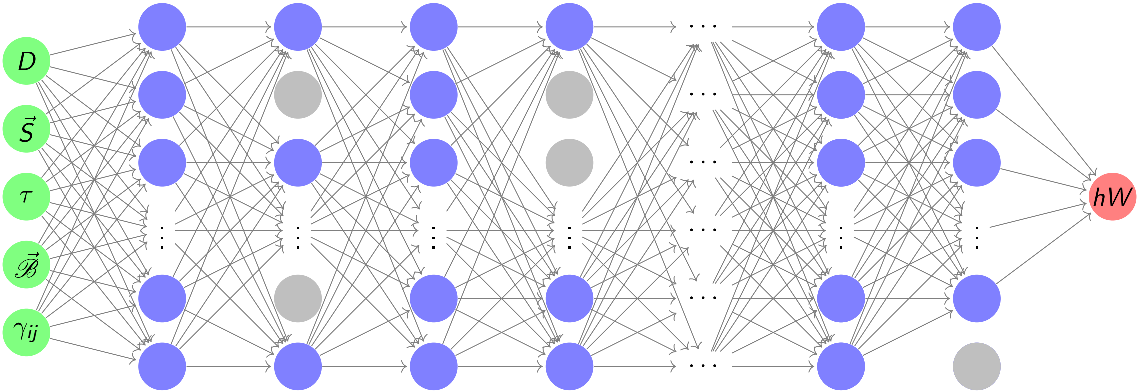


Figure: Illustration of the neural network architecture for the GRMHD models, with input layer (green), hidden layers (blue) and the output layer (red) and layers with dropped out neurons (gray)

Overview

Knobs analogy

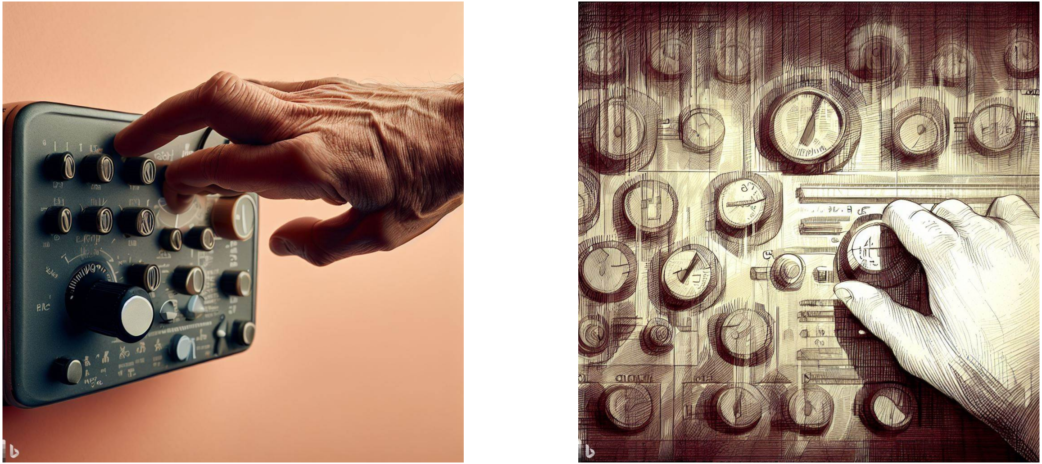


Figure: The hyperparameters are the knobs that we turn (left) while the parameters are the knobs that the machine turns (right). Image prompts credit: Alice Alders.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

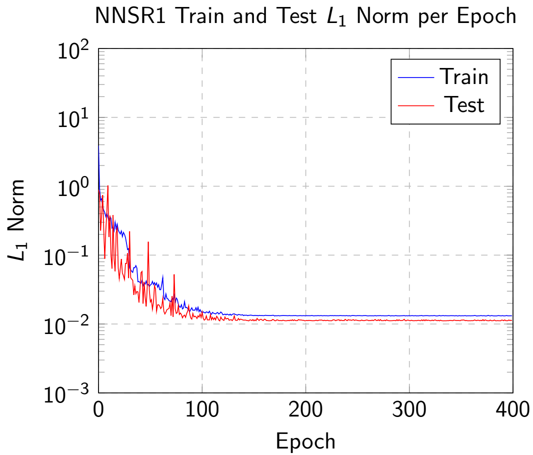
Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

Hyperparameter Comparison of SRHD Neural Network Models

Table: Hyperparameters for SRHD neural network models: NNC2PS, NNSR1, NNSR3, and NNSR4.

Hyperparameter	NNC2PS	NNSR1	NNSR3	NNSR4
Number of hidden layers	2	2	3	5
Number of hidden units	600, 200	600, 200	555, 458, 115	617, 858, 720, 989, 613
Hidden activation	Sigmoid	Sigmoid	ReLU	ReLU
Output activation	ReLU	ReLU	ReLU	ReLU
Loss function	MSE	MSE	Huber	MSE
Optimizer	Adam	Adam	RMSprop	Adagrad
Learning Rate	6×10^{-3}	6×10^{-3}	1.23×10^{-4}	1.69×10^{-3}
Batch Size	32	32	49	16
LR Scheduler	Red. Plat.	Red. Plat.	Red. Plat.	Cos. Ann.

NNSR1 Train and Test Norms per Epoch



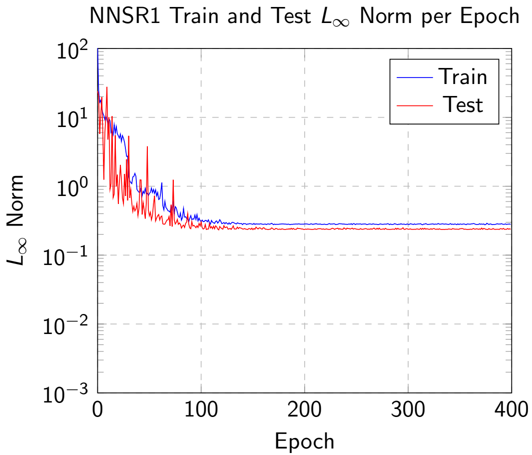
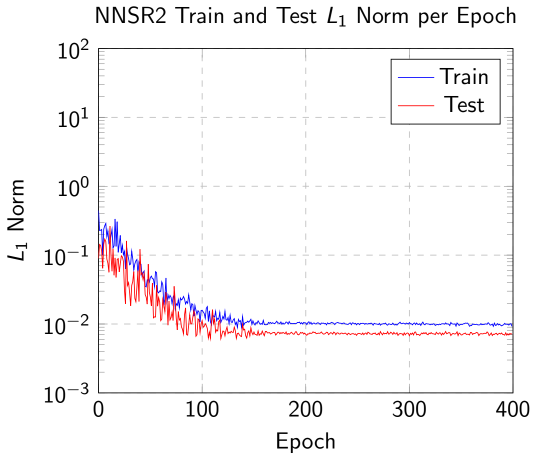


Figure: L_1 and L_∞ norm error plots per epoch for the NNSR1 model during training (blue) and testing (red).

NNSR2 Train and Test Norms per Epoch



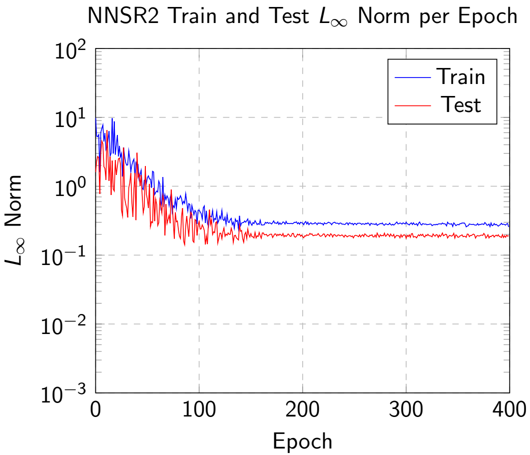


Figure: L_1 and L_∞ norm error plots per epoch for the NNSR2 model during training (blue) and testing (red).

NNSR3 Train and Test Norms per Epoch

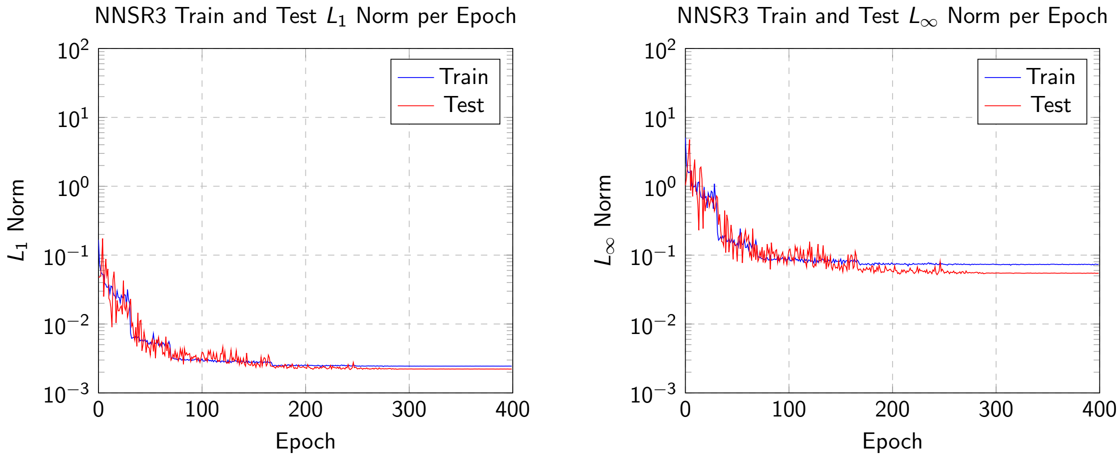


Figure: L_1 and L_∞ norm error plots per epoch for the NNSR3 model during training (blue) and testing (red).

NNSR3 Train and Test Norms per Epoch

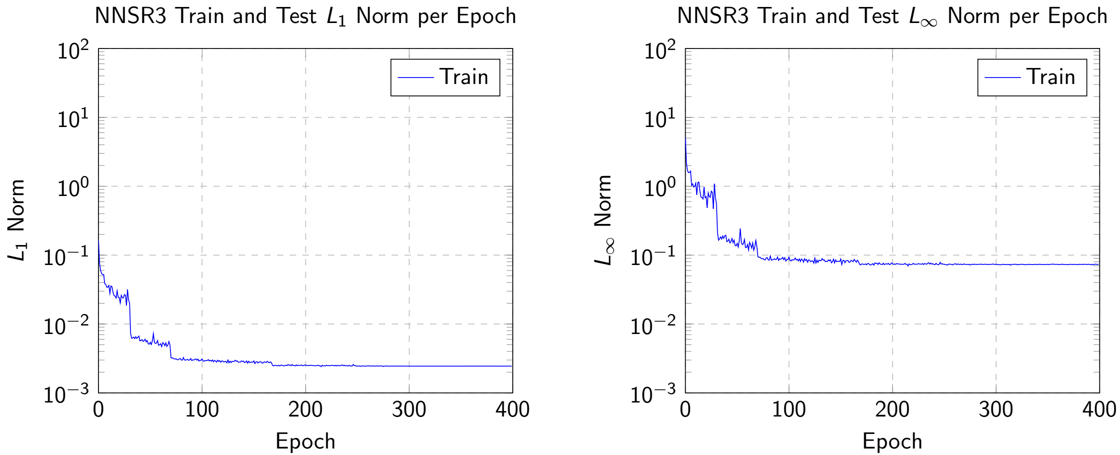
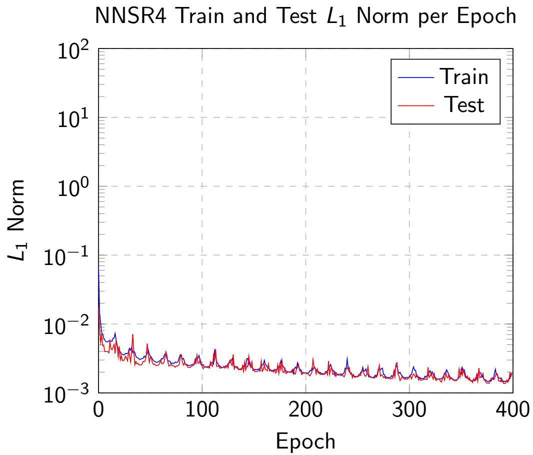


Figure: L_1 and L_∞ norm error plots per epoch for the NNSR3 model during training (blue) and testing (red).

NNSR4 Train and Test Norms per Epoch



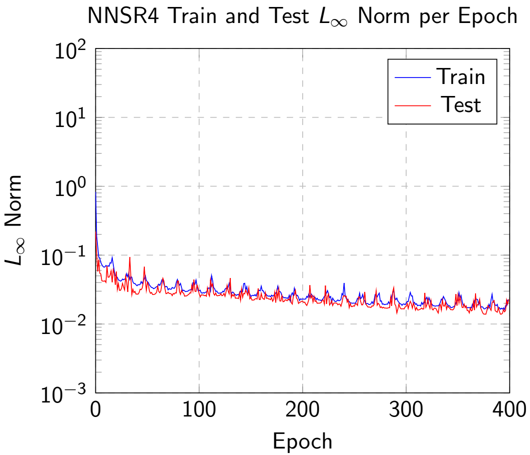


Figure: L_1 and L_∞ norm error plots per epoch for the NNSR4 model during training (blue) and testing (red).

NNSR4 Train and Test Norms per Epoch

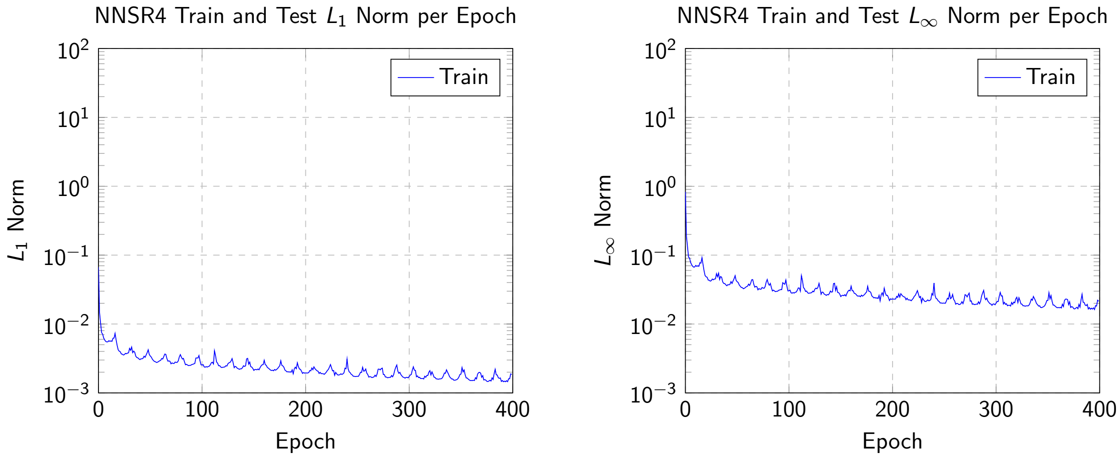


Figure: L_1 and L_∞ norm error plots per epoch for the NNSR4 model during training (blue) and testing (red).

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

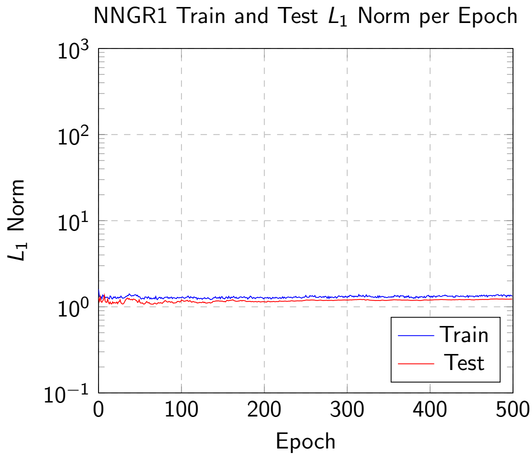
	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

Hyperparameters for GRMHD models

Table: Comparison of hyperparameters for NNGR1 and NNGR2 models

	NNGR1	NNGR2
Hidden layers	5	5
Hidden units	216, 2666, 1459, 485, 103	900, 113, 1440, 478, 3328
Hidden activation	PReLU	PReLU
Output activation	Linear	Linear
Loss function	MSE	MSE
Optimizer	Adagrad	Adagrad
Learn. rate	1.37×10^{-4}	1.11×10^{-4}
Batch size	512	512
LR scheduler	StepLR	ReduceLROnPlateau
Dropout rate	$\sim 29\%$	$\sim 47\%$

NNGR1 Train and Test Norms per Epoch



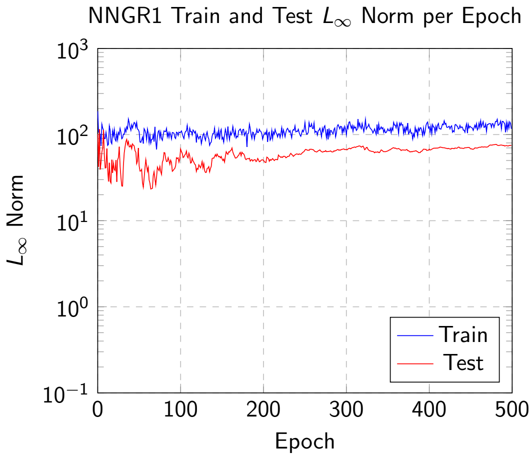


Figure: L_1 and L_∞ norm error plots per epoch for the NNGR1 model during training (blue) and testing (red).

NNGR2 Train and Test Norms per Epoch

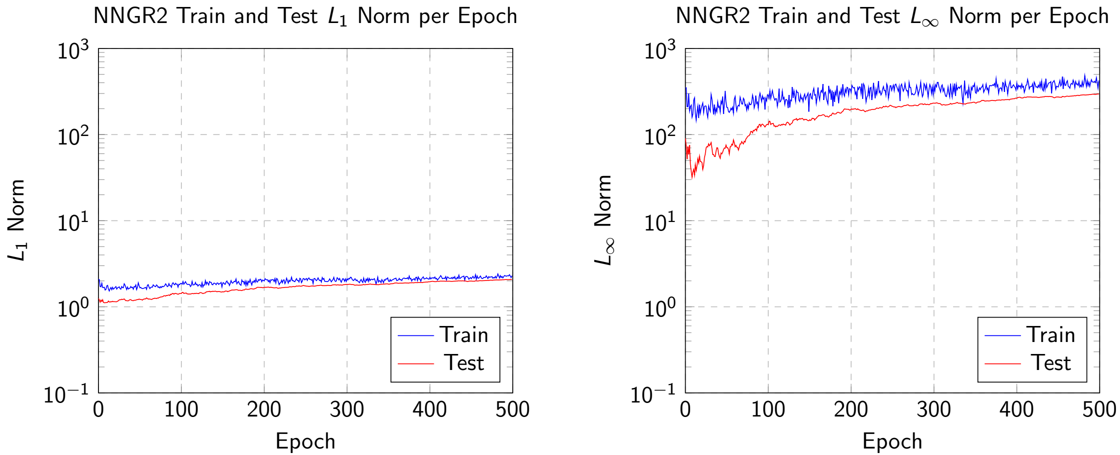


Figure: L_1 and L_∞ norm error plots per epoch for the NNGR2 model during training (blue) and testing (red).

Training settings and evaluation times

	NNSR1	NNSR2	NNSR3	NNSR4	NNGR1	NNGR2
Total samples	90k	90k	90k	90k	100k	100k
Train	80k	80k	80k	80k	80k	80k
Validation	–	–	–	–	15k	15k
Test	10k	10k	10k	10k	15k	15k
Trials	–	–	250	250	500	672
Epochs	400	400	400	400	500	500
Parameters	100k	300k	300k	2500k	5000k	2500k
Average time (μs)	88.3	72.7	361	245	550	581
Best time (μs)	18.4	20.5	24.6	64.5	404	368

Table: Summary of training settings and evaluation times per model.

Training settings and evaluation times

	NNSR1	NNSR2	NNSR3	NNSR4	NNGR1	NNGR2
Total samples	90k	90k	90k	90k	100k	100k
Train	80k	80k	80k	80k	80k	80k
Validation	–	–	–	–	15k	15k
Test	10k	10k	10k	10k	15k	15k
Trials	–	–	250	250	500	672
Epochs	400	400	400	400	500	500
Parameters	100k	300k	300k	2500k	5000k	2500k
Average time (μs)	88.3	72.7	361	245	550	581
Best time (μs)	18.4	20.5	24.6	64.5	404	368

Table: Summary of training settings and evaluation times per model.

Training settings and evaluation times

	NNSR1	NNSR2	NNSR3	NNSR4	NNGR1	NNGR2
Total samples	90k	90k	90k	90k	100k	100k
Train	80k	80k	80k	80k	80k	80k
Validation	–	–	–	–	15k	15k
Test	10k	10k	10k	10k	15k	15k
Trials	–	–	250	250	500	672
Epochs	400	400	400	400	500	500
Parameters	100k	300k	300k	2500k	5000k	2500k
Average time (μs)	88.3	72.7	361	245	550	581
Best time (μs)	18.4	20.5	24.6	64.5	404	368

Table: Summary of training settings and evaluation times per model.

Training settings and evaluation times

	NNSR1	NNSR2	NNSR3	NNSR4	NNGR1	NNGR2
Total samples	90k	90k	90k	90k	100k	100k
Train	80k	80k	80k	80k	80k	80k
Validation	–	–	–	–	15k	15k
Test	10k	10k	10k	10k	15k	15k
Trials	–	–	250	250	500	672
Epochs	400	400	400	400	500	500
Parameters	100k	300k	300k	2500k	5000k	2500k
Average time (μs)	88.3	72.7	361	245	550	581
Best time (μs)	18.4	20.5	24.6	64.5	404	368

Table: Summary of training settings and evaluation times per model.

Training settings and evaluation times

	NNSR1	NNSR2	NNSR3	NNSR4	NNGR1	NNGR2
Total samples	90k	90k	90k	90k	100k	100k
Train	80k	80k	80k	80k	80k	80k
Validation	–	–	–	–	15k	15k
Test	10k	10k	10k	10k	15k	15k
Trials	–	–	250	250	500	672
Epochs	400	400	400	400	500	500
Parameters	100k	300k	300k	2500k	5000k	2500k
Average time (μs)	88.3	72.7	361	245	550	581
Best time (μs)	18.4	20.5	24.6	64.5	404	368

Table: Summary of training settings and evaluation times per model.

Training settings and evaluation times

	NNSR1	NNSR2	NNSR3	NNSR4	NNGR1	NNGR2
Total samples	90k	90k	90k	90k	100k	100k
Train	80k	80k	80k	80k	80k	80k
Validation	–	–	–	–	15k	15k
Test	10k	10k	10k	10k	15k	15k
Trials	–	–	250	250	500	672
Epochs	400	400	400	400	500	500
Parameters	100k	300k	300k	2500k	5000k	2500k
Average time (μs)	88.3	72.7	361	245	550	581
Best time (μs)	18.4	20.5	24.6	64.5	404	368

Table: Summary of training settings and evaluation times per model.

Evaluation times compared to root-finders

	Average fraction NNSR1	Best fraction NNSR1	Average fraction NNGR1	Best fraction NNGR1	Average fraction NNGR2	Best fraction NNGR2
$n_{\text{cells}} = 32$	5.63	27.0	0.904	1.23	0.856	1.35
$n_{\text{cells}} = 64$	23.3	112	3.73	5.08	3.53	5.58
$n_{\text{cells}} = 128$	99.7	478	16.0	21.8	15.2	23.9
$n_{\text{cells}} = 256$	410	1960	65.9	89.7	62.3	98.4

Table: The evaluation times are represented as **fractions** of root-finder times over respective model times. Higher values denote faster model performance. Root-finder values by de Graaf [4].

Evaluation times compared to root-finders

	Average fraction NNSR1	Best fraction NNSR1	Average fraction NNGR1	Best fraction NNGR1	Average fraction NNGR2	Best fraction NNGR2
$n_{\text{cells}} = 32$	5.63	27.0	0.904	1.23	0.856	1.35
$n_{\text{cells}} = 64$	23.3	112	3.73	5.08	3.53	5.58
$n_{\text{cells}} = 128$	99.7	478	16.0	21.8	15.2	23.9
$n_{\text{cells}} = 256$	410	1960	65.9	89.7	62.3	98.4

Table: The evaluation times are represented as **fractions** of root-finder times over respective model times. **Higher values denote faster model performance.** Root-finder values by de Graaf [4].

Evaluation times compared to root-finders

	Average fraction NNSR1	Best fraction NNSR1	Average fraction NNGR1	Best fraction NNGR1	Average fraction NNGR2	Best fraction NNGR2
$n_{\text{cells}} = 32$	5.63	27.0	0.904	1.23	0.856	1.35
$n_{\text{cells}} = 64$	23.3	112	3.73	5.08	3.53	5.58
$n_{\text{cells}} = 128$	99.7	478	16.0	21.8	15.2	23.9
$n_{\text{cells}} = 256$	410	1960	65.9	89.7	62.3	98.4

Table: The evaluation times are represented as **fractions** of root-finder times over respective model times. **Higher values denote faster model performance.** Root-finder values by de Graaf [4].

Evaluation times compared to root-finders

	Average fraction NNSR1	Best fraction NNSR1	Average fraction NNGR1	Best fraction NNGR1	Average fraction NNGR2	Best fraction NNGR2
$n_{\text{cells}} = 32$	5.63	27.0	0.904 :-(	1.23	0.856 :-(	1.35
$n_{\text{cells}} = 64$	23.3	112	3.73	5.08	3.53	5.58
$n_{\text{cells}} = 128$	99.7	478	16.0	21.8	15.2	23.9
$n_{\text{cells}} = 256$	410	1960	65.9 :-)	89.7	62.3	98.4 :-)

Table: The evaluation times are represented as **fractions** of root-finder times over respective model times. **Higher values denote faster model performance.** Root-finder values by de Graaf [4].

Evaluation times compared to root-finders

	Average fraction NNSR1	Best fraction NNSR1	Average fraction NNGR1	Best fraction NNGR1	Average fraction NNGR2	Best fraction NNGR2
$n_{\text{cells}} = 32$	5.63	27.0	0.904	1.23	0.856	1.35
$n_{\text{cells}} = 64$	23.3	112	3.73	5.08	3.53	5.58
$n_{\text{cells}} = 128$	99.7	478	16.0	21.8	15.2	23.9
$n_{\text{cells}} = 256$	410	1960 :-O	65.9	89.7	62.3	98.4

Table: The evaluation times are represented as **fractions** of root-finder times over respective model times. **Higher values denote faster model performance.** Root-finder values by de Graaf [4].

Overview

Overview

Conclusion - Accomplishments Part I

- Explored con2prim inversion in SRHD using a neural network.
- Extended this investigation to the more complex GRMHD case.
- Achieved computational time reduction of con2prim inversion in GRMHD by at least an order of magnitude (given successful parallelization on the GPU), benchmarked on the Nvidia RTX A6000 GPU.

Conclusion - Accomplishments Part I

- Explored con2prim inversion in SRHD using a neural network.
- Extended this investigation to the more complex GRMHD case.
- Achieved computational time reduction of con2prim inversion in GRMHD by at least an order of magnitude (given successful parallelization on the GPU), benchmarked on the Nvidia RTX A6000 GPU.

Conclusion - Accomplishments Part I

- Explored con2prim inversion in SRHD using a neural network.
- Extended this investigation to the more complex GRMHD case.
- Achieved computational time reduction of con2prim inversion in GRMHD by at least an order of magnitude (given successful parallelization on the GPU), benchmarked on the Nvidia RTX A6000 GPU.

Conclusion - Accomplishments Part II

- Despite initial challenges with the SRHD models, improved network tuning led to achieving acceptable error metrics.
- For GRMHD models, despite higher errors, demonstrated a reduction in computation time, achieving up to 100 times faster evaluation time [1] than the current GRaM-X implementation.
- Provided potential for complete integration of the GRMHD models into the GRaM-X framework.

[1] Again, given NNs work well with GPU parallelization on GRaM-X

Conclusion - Accomplishments Part II

- Despite initial challenges with the SRHD models, improved network tuning led to achieving acceptable error metrics.
- For GRMHD models, despite higher errors, demonstrated a reduction in computation time, achieving up to 100 times faster evaluation time [1] than the current GRaM-X implementation.
- Provided potential for complete integration of the GRMHD models into the GRaM-X framework.

[1] Again, given NNs work well with GPU parallelization on GRaM-X

Conclusion - Accomplishments Part II

- Despite initial challenges with the SRHD models, improved network tuning led to achieving acceptable error metrics.
- For GRMHD models, despite higher errors, demonstrated a reduction in computation time, achieving up to 100 times faster evaluation time [1] than the current GRaM-X implementation.
- Provided potential for complete integration of the GRMHD models into the GRaM-X framework.

[1] Again, given NNs work well with GPU parallelization on GRaM-X

Conclusion - Future Work and Final Thoughts

- Future work: Integrate GRMHD NNs into GRaM-X by providing tabulated EoS and multiple outputs, quantify relationship between hyperparameters and NN performance.
- Despite challenges, the project underlines the potential of machine learning in relativistic hydrodynamics.
- Optimistic about future advancements and applications of these techniques in GRMHD.

Conclusion - Future Work and Final Thoughts

- Future work: Integrate GRMHD NNs into GRaM-X by providing tabulated EoS and multiple outputs, quantify relationship between hyperparameters and NN performance.
- Despite challenges, the project underlines the potential of machine learning in relativistic hydrodynamics.
- Optimistic about future advancements and applications of these techniques in GRMHD.

Conclusion - Future Work and Final Thoughts

- Future work: Integrate GRMHD NNs into GRaM-X by providing tabulated EoS and multiple outputs, quantify relationship between hyperparameters and NN performance.
- Despite challenges, the project underlines the potential of machine learning in relativistic hydrodynamics.
- Optimistic about future advancements and applications of these techniques in GRMHD.

References I

[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[2] Tobias Dieselhorst et al. "Machine Learning for Conservative-to-Primitive in Relativistic Hydrodynamics". In: (2021). URL: https://doi.org/10.3390/sym13112157.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[4] Yannick de Graaf. "GPU Optimization of GRMHD code GRHydroX". MA thesis. University of Amsterdam, 2023.

[5] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[6] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[7] Swapnil Shankar et al. "GRaM-X: A new GPU-accelerated dynamical spacetime GRMHD code for Exascale computing with the Einstein Toolkit". In: *arXiv preprint arXiv:2210.17509* (2022).

Appendix

SRHD—Flux vectors

- The flux vectors $\mathbf{F}^i$ describe how the conserved quantities change across a fluid surface

$$\mathbf{F}^i = \left(Dv^i, S^1 v^i + p\delta^{1i}, S^2 v^i + p\delta^{2i}, S^3 v^i + p\delta^{3i}, S^i - Dv^i\right) \;, \tag{12}$$

Computational times of the old algorithm

Table: Comparative runtime of GraM-X simulation evolution steps at different grid resolutions. Data adapted from [4]

Evolution step	$n_{\text{cells}} = 32$ (us)	$n_{\text{cells}} = 64$ (us)	$n_{\text{cells}} = 128$ (us)	$n_{\text{cells}} = 256$ (us)
Con2prim Interior	498	2052	8802	36225
Fluxes	1560	6975	54303	434775
Source	171	741	5115	40164
Update	81	300	2133	16956
Tmunu	237	1137	8232	65409
GraM-X Iteration	36699	157401	1008399	10702599

Computational times of the old algorithm

Table: Comparative runtime of GraM-X simulation evolution steps at different grid resolutions. Data adapted from [4]

Evolution step	$n_{\text{cells}} = 32$ (us)	$n_{\text{cells}} = 64$ (us)	$n_{\text{cells}} = 128$ (us)	$n_{\text{cells}} = 256$ (us)
Con2prim Interior	498	2052	8802	36225
Fluxes	1560	6975	54303	434775
Source	171	741	5115	40164
Update	81	300	2133	16956
Tmunu	237	1137	8232	65409
GraM-X Iteration	36699	157401	1008399	10702599

Computational times of the old algorithm

Table: Comparative runtime of GraM-X simulation evolution steps at different grid resolutions. Data adapted from [4]

Evolution step	$n_{\text{cells}} = 32$ (us)	$n_{\text{cells}} = 64$ (us)	$n_{\text{cells}} = 128$ (us)	$n_{\text{cells}} = 256$ (us)
Con2prim Interior	498	2052	8802	36225
Fluxes	1560	6975	54303	434775
Source	171	741	5115	40164
Update	81	300	2133	16956
Tmunu	237	1137	8232	65409
GraM-X Iteration	36699	157401	1008399	10702599

Computational times of the old algorithm

Table: Comparative runtime of GraM-X simulation evolution steps at different grid resolutions. Data adapted from [4]

Evolution step	$n_{\text{cells}} = 32$ (us)	$n_{\text{cells}} = 64$ (us)	$n_{\text{cells}} = 128$ (us)	$n_{\text{cells}} = 256$ (us)
Con2prim Interior	498	2052	8802	36225
Fluxes	1560	6975	54303	434775
Source	171	741	5115	40164
Update	81	300	2133	16956
Tmunu	237	1137	8232	65409
GraM-X Iteration	36699	157401	1008399	10702599

Obtaining primitive variables from pressure

In the special relativistic case, all the other primitive variables can be constructed once we know the pressure [2]:

$$\begin{aligned}
\rho(p) &= \frac{D}{W(p)} \, , \\
\epsilon(p) &= \frac{\tau + D\left[1 - W(p)\right] + p\left[1 - W^2(p)\right]}{DW(p)} \, , \\
W(p) &= \frac{1}{\sqrt{1 - v^2(p)}} \, , \\
v^i(p) &= \frac{S^i}{\tau + D + p} \, .
\end{aligned} \tag{13}$$

Additional GRMHD definitions

- New variables for GRMHD:

$$b^\mu = u_\nu {}^*F^{\mu\nu} \ , \tag{14}$$

$$p^* := p + \frac{b^2}{2} \ , \tag{15}$$

$$h^* := 1 + \epsilon + \frac{(p + b^2)}{\rho} \ . \tag{16}$$

- Field tensor and dual tensor:

$$F^{\mu\nu} = \begin{pmatrix} 0 & -E_x & -E_y & -E_z \\ E_x & 0 & -B_z & B_y \\ E_y & B_z & 0 & -B_x \\ E_z & -B_y & B_x & 0 \end{pmatrix} \ , \tag{17}$$

$${}^*F^{\mu\nu} = \frac{1}{2}\varepsilon^{\mu\nu\rho\sigma} F_{\rho\sigma} \tag{18}$$

Additional GRMHD definitions

- New variables for GRMHD:

$$b^\mu = u_\nu {}^*F^{\mu\nu} \ , \tag{14}$$

$$p^* := p + \frac{b^2}{2} \ , \tag{15}$$

$$h^* := 1 + \epsilon + \frac{(p + b^2)}{\rho} \ . \tag{16}$$

- Field tensor and dual tensor:

$$F^{\mu\nu} = \begin{pmatrix} 0 & -E_x & -E_y & -E_z \\ E_x & 0 & -B_z & B_y \\ E_y & B_z & 0 & -B_x \\ E_z & -B_y & B_x & 0 \end{pmatrix} \ , \tag{17}$$

$$ {}^*F^{\mu\nu} = \frac{1}{2}\varepsilon^{\mu\nu\rho\sigma}F_{\rho\sigma} \tag{18}$$

Weights, Biases, and Activation Functions

- Each node in an ANN performs a weighted sum of their inputs, followed by the application of an activation function.
- Weights and biases form the adjustable parameters of the network, tuned during the learning process.
- Activation functions, such as the sigmoid or Rectified Linear Unit (ReLU) functions, introduce non-linearity into the model, enabling it to learn and represent more complex relationships in the data.

Weights, Biases, and Activation Functions

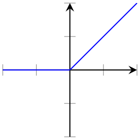
- Each node in an ANN performs a weighted sum of their inputs, followed by the application of an activation function.
- Weights and biases form the adjustable parameters of the network, tuned during the learning process.
- Activation functions, such as the sigmoid or Rectified Linear Unit (ReLU) functions, introduce non-linearity into the model, enabling it to learn and represent more complex relationships in the data.

Weights, Biases, and Activation Functions

- Each node in an ANN performs a weighted sum of their inputs, followed by the application of an activation function.
- Weights and biases form the adjustable parameters of the network, tuned during the learning process.
- Activation functions, such as the sigmoid or Rectified Linear Unit (ReLU) functions, introduce non-linearity into the model, enabling it to learn and represent more complex relationships in the data.

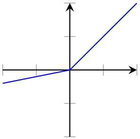
Activation functions

ReLU



$\max(0, x)$

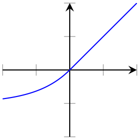
LeakyReLU / PReLU



$\max(ax, x)$

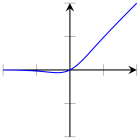
$\max(0, x) + a \cdot \min(0, x)$

ELU



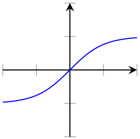
$\max(e^x - 1, x)$

GELU



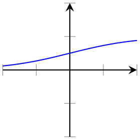
$bx(1 + \tanh(cx))$

Tanh



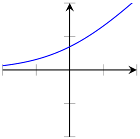
$\tanh(x)$

Sigmoid



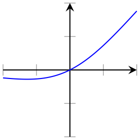
$1/(1 + e^{-x})$

SoftPlus



$\ln(1 + e^x)$

Swish



$x/(1 + e^{-x})$

Artificial Neural Networks—Forward and Backward Propagation

- Training ANNs involves forward propagation and backpropagation [6].
- During forward propagation, input data is passed through the network, and an output prediction is generated.
- Backpropagation involves propagating the network's error backwards and updating the weights according to the calculated gradients.

Artificial Neural Networks—Forward and Backward Propagation

- Training ANNs involves forward propagation and backpropagation [6].
- During forward propagation, input data is passed through the network, and an output prediction is generated.
- Backpropagation involves propagating the network's error backwards and updating the weights according to the calculated gradients.

Artificial Neural Networks—Forward and Backward Propagation

- Training ANNs involves forward propagation and backpropagation [6].
- During forward propagation, input data is passed through the network, and an output prediction is generated.
- Backpropagation involves propagating the network's error backwards and updating the weights according to the calculated gradients.

Artificial Neural Networks—Summary

- ANNs, as part of ML and AI, are powerful tools for modelling complex patterns within data.
- Deep Learning allows for abstraction of features through layered networks.
- Understanding ANNs as functions and the role of weights, biases, and activation functions are key to comprehend their learning capabilities.
- Forward and backward propagation are key processes in training ANNs.

Flowchart Part 1

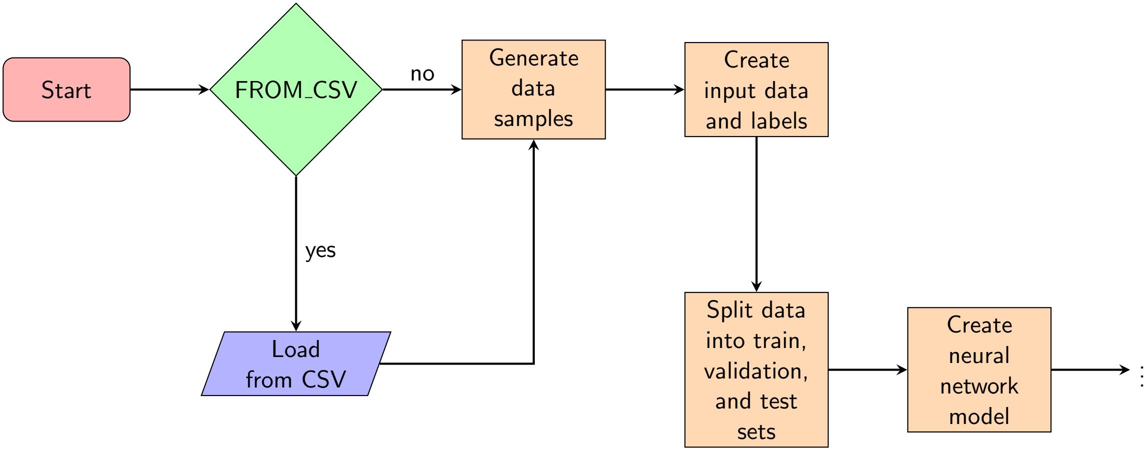


Figure: Part 1 of the flowchart illustrating program flow for con2prim.

Flowchart Part 2

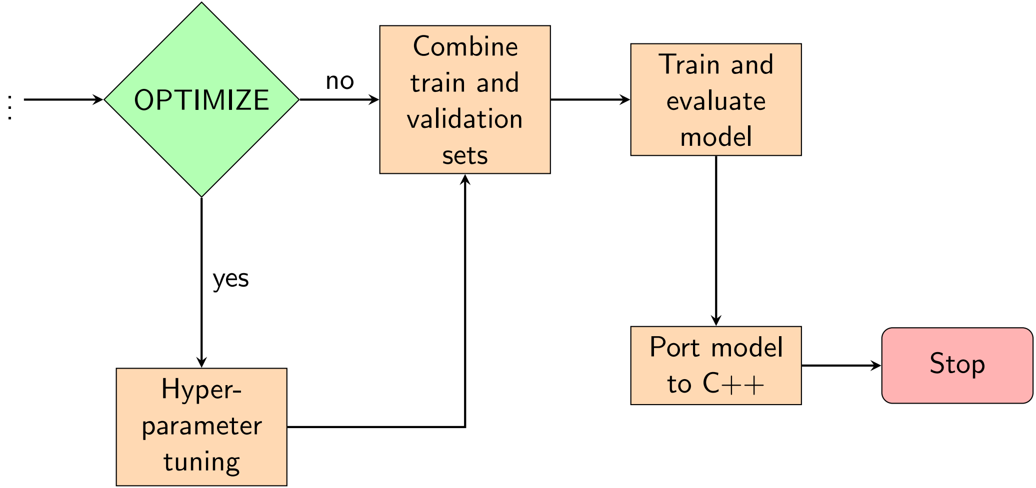


Figure: Part 2 of the flowchart illustrating program flow for con2prim.

Selected Hyperparameter search spaces

Selected hyperparameters	Search space for NNSR3	Search space for NNGR
n_{layers}	$1 \leq n_{\text{layers}} \leq 3$	$1 \leq n_{\text{layers}} \leq 5$
n_{units}	$16 \leq n_{\text{units}} \leq 256$	$16 \leq n_{\text{units}} \leq 4096$
Hidden activation function	ReLU, LeakyReLU, ELU, Tanh, Sigmoid	ReLU, LeakyReLU, ELU, PReLU, Swish, GELU, SoftPlus
Output activation function	Linear, ReLU	Linear
Loss function	MSE, MAE, Huber, LogCosh	MSE, MAE, Huber
Optimizer	Adam, SGD, RMSprop, Adagrad	Adam, SGD, RMSprop, Adagrad
Learning rate (η)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)
Scheduler	None, Cos. Ann., Red. Plat., S. LR, Exp. LR	Cos. Ann., Red. Plat., S. LR
Dropout rate (p_{dropout})	–	$0.0 \leq p_{\text{dropout}} \leq 0.5$

Selected Hyperparameter search spaces

Selected hyperparameters	Search space for NNSR3	Search space for NNGR
n_{layers}	$1 \leq n_{\text{layers}} \leq 3$	$1 \leq n_{\text{layers}} \leq 5$
n_{units}	$16 \leq n_{\text{units}} \leq 256$	$16 \leq n_{\text{units}} \leq 4096$
Hidden activation function	ReLU, LeakyReLU, ELU, Tanh, Sigmoid	ReLU, LeakyReLU, ELU, PReLU, Swish, GELU, SoftPlus
Output activation function	Linear, ReLU	Linear
Loss function	MSE, MAE, Huber, LogCosh	MSE, MAE, Huber
Optimizer	Adam, SGD, RMSprop, Adagrad	Adam, SGD, RMSprop, Adagrad
Learning rate (η)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)
Scheduler	None, Cos. Ann., Red. Plat., S. LR, Exp. LR	Cos. Ann., Red. Plat., S. LR
Dropout rate (p_{dropout})	–	$0.0 \leq p_{\text{dropout}} \leq 0.5$

Selected Hyperparameter search spaces

Selected hyperparameters	Search space for NNSR3	Search space for NNGR
n_{layers}	$1 \leq n_{\text{layers}} \leq 3$	$1 \leq n_{\text{layers}} \leq 5$
n_{units}	$16 \leq n_{\text{units}} \leq 256$	$16 \leq n_{\text{units}} \leq 4096$
Hidden activation function	ReLU, LeakyReLU, ELU, Tanh, Sigmoid	ReLU, LeakyReLU, ELU, PReLU, Swish, GELU, SoftPlus
Output activation function	Linear, ReLU	Linear
Loss function	MSE, MAE, Huber, LogCosh	MSE, MAE, Huber
Optimizer	Adam, SGD, RMSprop, Adagrad	Adam, SGD, RMSprop, Adagrad
Learning rate (η)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)
Scheduler	None, Cos. Ann., Red. Plat., S. LR, Exp. LR	Cos. Ann., Red. Plat., S. LR
Dropout rate (p_{dropout})	–	$0.0 \leq p_{\text{dropout}} \leq 0.5$

Selected Hyperparameter search spaces

Selected hyperparameters	Search space for NNSR3	Search space for NNGR
n_{layers}	$1 \leq n_{\text{layers}} \leq 3$	$1 \leq n_{\text{layers}} \leq 5$
n_{units}	$16 \leq n_{\text{units}} \leq 256$	$16 \leq n_{\text{units}} \leq 4096$
Hidden activation function	ReLU, LeakyReLU, ELU, Tanh, Sigmoid	ReLU, LeakyReLU, ELU, PReLU, Swish, GELU, SoftPlus
Output activation function	Linear, ReLU	Linear
Loss function	MSE, MAE, Huber, LogCosh	MSE, MAE, Huber
Optimizer	Adam, SGD, RMSprop, Adagrad	Adam, SGD, RMSprop, Adagrad
Learning rate (η)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)
Scheduler	None, Cos. Ann., Red. Plat., S. LR, Exp. LR	Cos. Ann., Red. Plat., S. LR
Dropout rate (p_{dropout})	–	$0.0 \leq p_{\text{dropout}} \leq 0.5$

Selected Hyperparameter search spaces

Selected hyperparameters	Search space for NNSR3	Search space for NNGR
n_{layers}	$1 \leq n_{\text{layers}} \leq 3$	$1 \leq n_{\text{layers}} \leq 5$
n_{units}	$16 \leq n_{\text{units}} \leq 256$	$16 \leq n_{\text{units}} \leq 4096$
Hidden activation function	ReLU, LeakyReLU, ELU, Tanh, Sigmoid	ReLU, LeakyReLU, ELU, PReLU, Swish, GELU, SoftPlus
Output activation function	Linear, ReLU	Linear
Loss function	MSE, MAE, Huber, LogCosh	MSE, MAE, Huber
Optimizer	Adam, SGD, RMSprop, Adagrad	Adam, SGD, RMSprop, Adagrad
Learning rate (η)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)
Scheduler	None, Cos. Ann., Red. Plat., S. LR, Exp. LR	Cos. Ann., Red. Plat., S. LR
Dropout rate (p_{dropout})	–	$0.0 \leq p_{\text{dropout}} \leq 0.5$

Selected Hyperparameter search spaces

Selected hyperparameters	Search space for NNSR3	Search space for NNGR
n_{layers}	$1 \leq n_{\text{layers}} \leq 3$	$1 \leq n_{\text{layers}} \leq 5$
n_{units}	$16 \leq n_{\text{units}} \leq 256$	$16 \leq n_{\text{units}} \leq 4096$
Hidden activation function	ReLU, LeakyReLU, ELU, Tanh, Sigmoid	ReLU, LeakyReLU, ELU, PReLU, Swish, GELU, SoftPlus
Output activation function	Linear, ReLU	Linear
Loss function	MSE, MAE, Huber, LogCosh	MSE, MAE, Huber
Optimizer	Adam, SGD, RMSprop, Adagrad	Adam, SGD, RMSprop, Adagrad
Learning rate (η)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)
Scheduler	None, Cos. Ann., Red. Plat., S. LR, Exp. LR	Cos. Ann., Red. Plat., S. LR
Dropout rate (p_{dropout})	–	$0.0 \leq p_{\text{dropout}} \leq 0.5$

Selected Hyperparameter search spaces

Selected hyperparameters	Search space for NNSR3	Search space for NNGR
n_{layers}	$1 \leq n_{\text{layers}} \leq 3$	$1 \leq n_{\text{layers}} \leq 5$
n_{units}	$16 \leq n_{\text{units}} \leq 256$	$16 \leq n_{\text{units}} \leq 4096$
Hidden activation function	ReLU, LeakyReLU, ELU, Tanh, Sigmoid	ReLU, LeakyReLU, ELU, PReLU, Swish, GELU, Soft-Plus
Output activation function	Linear, ReLU	Linear
Loss function	MSE, MAE, Huber, LogCosh	MSE, MAE, Huber
Optimizer	Adam, SGD, RMSprop, Ada-grad	Adam, SGD, RMSprop, Ada-grad
Learning rate (η)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)	$1 \times 10^{-4} \leq \eta \leq 1 \times 10^{-2}$ (log-uniform)
Scheduler	None, Cos. Ann., Red. Plat., S. LR, Exp. LR	Cos. Ann., Red. Plat., S. LR
Dropout rate (p_{dropout})	–	$0.0 \leq p_{\text{dropout}} \leq 0.5$

Training Process

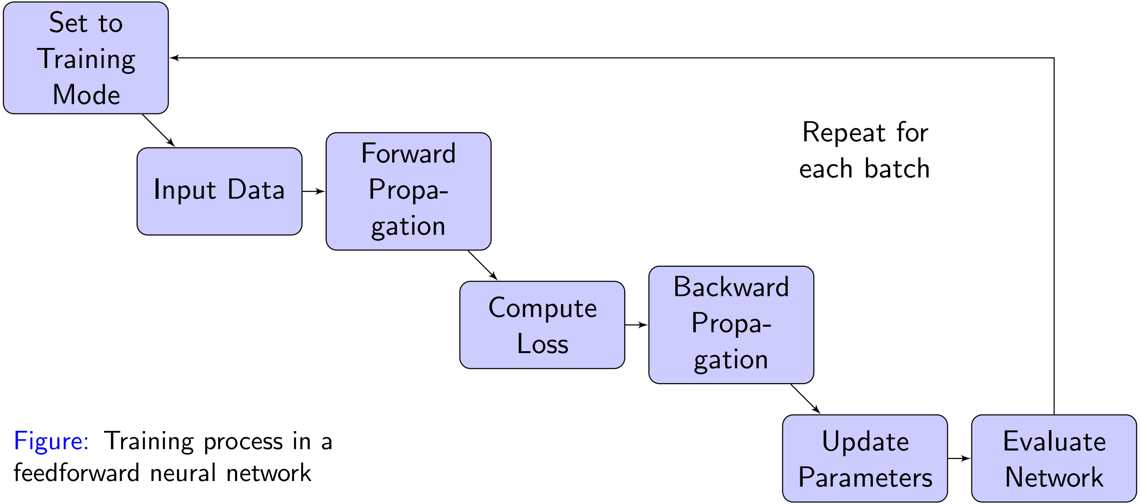


Figure: Training process in a feedforward neural network

Evaluation Process

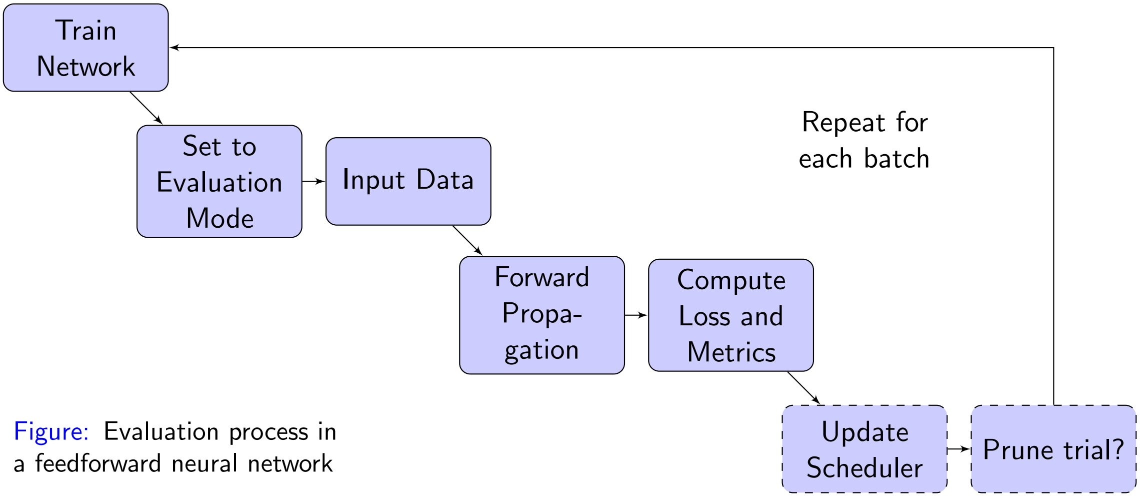
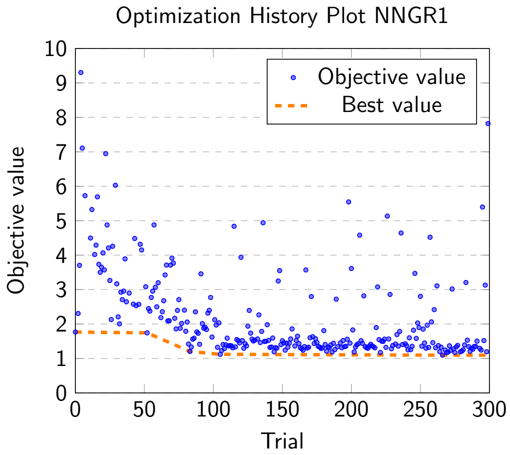


Figure: Evaluation process in a feedforward neural network

Optimization histories for GRMHD models



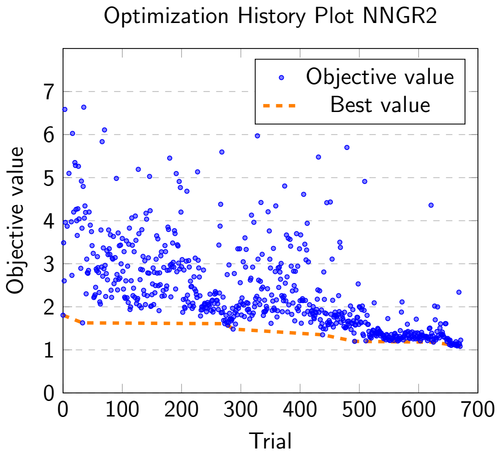
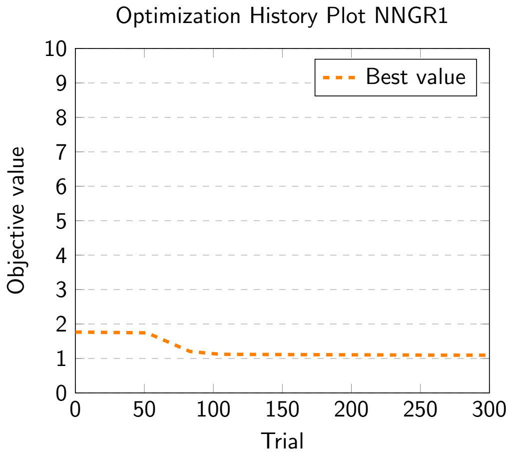


Figure: Optimization history plots for the NNGR1 (left) and NNGR2 (right) models.

Optimization histories for GRMHD models



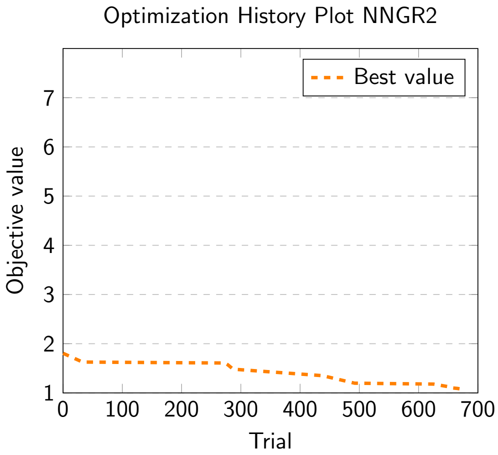


Figure: Optimization history plots for the NNGR1 (left) and NNGR2 (right) models.

Parameter importances NNGR1

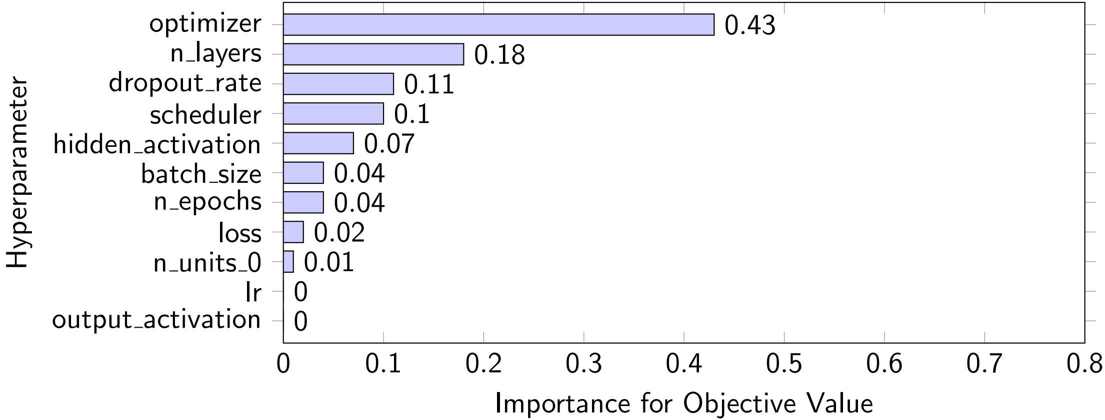


Figure: Relative importance of different hyperparameters for the NNGR1 model, represented by their objective value fractions.

Parameter importances NNGR2

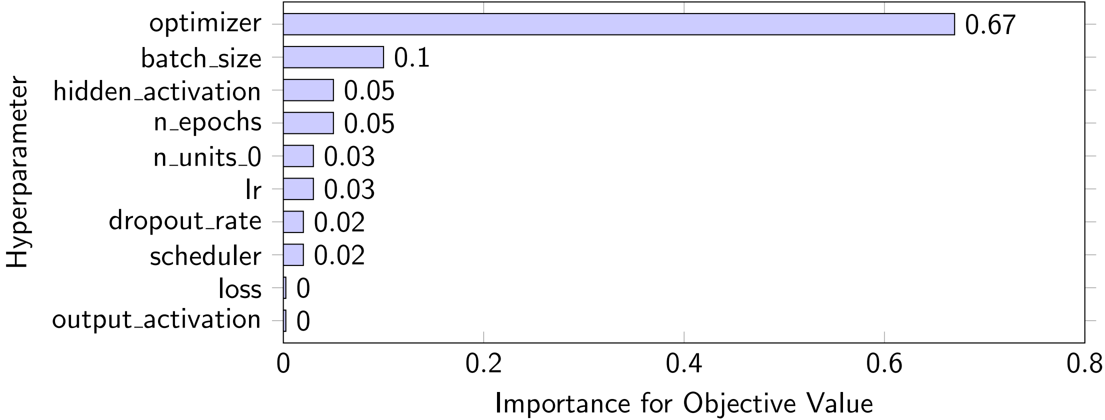


Figure: Relative importance of different hyperparameters for the NNGR2 model, represented by their objective value fractions.