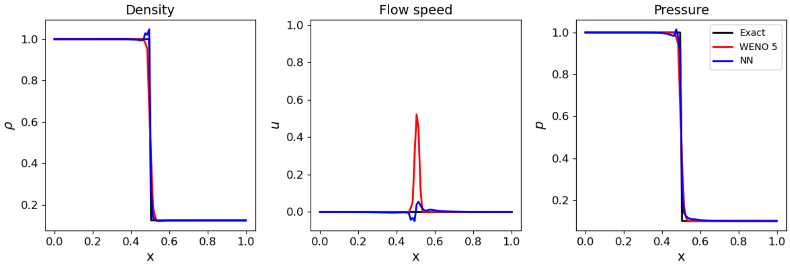
Physics-informed neural networks to solve the compressible Euler equations

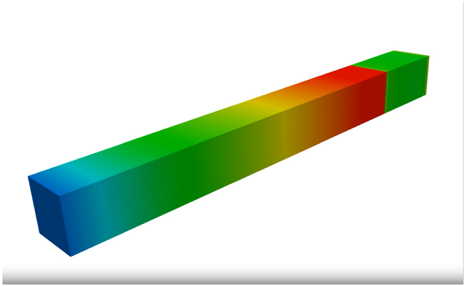
A parametric hyperparameter tuning approach



Dario Rodriguez (AE)

Physics Overview

- Shock tubes create high-pressure, high-temperature conditions for testing materials, producing combustion or as high-pressure gas source in a shock tube-driven wind tunnel
- Discontinuities in flowfield arise and experiments are very transitory in nature (miliseconds) $\rightarrow$ Numerical methods that handle discontinuities (shocks) accurate predictions
- Approach: conservation laws of mass, momentum and energy (inviscid limit) to characterize the flowfield (Euler equations)



https://help.sim-flow.com/validation/sod-shock

Project Goals

Research question: How well a NN can manage irregularities like shocks?

- Approximate solution of the Euler equations using a neural network using PyTorch
- Strategies considered: domain extension, weighting losses, clustered sampling of internal points
- Perform a simple hyperparameter tuning study (trial and error) to set the neural network parameters
- Compare computational cost with analytical solutions and a classical finite volume method (5^{th} order WENO)
- ~~Implement an inverse problem where the physics of a 1-D compressible flow is inferred from density gradients data (Schilieren imaging)~~

PINNs to approximate high-speed unsteady compressible flows

Approximate solution of the Euler equations using a neural network and a set of BC/IC

- 1-D Compressible Euler equations (hyperbolic PDEs in characteristic form)

$$\frac{\partial U}{\partial t} + A\frac{\partial U}{\partial x} = 0$$

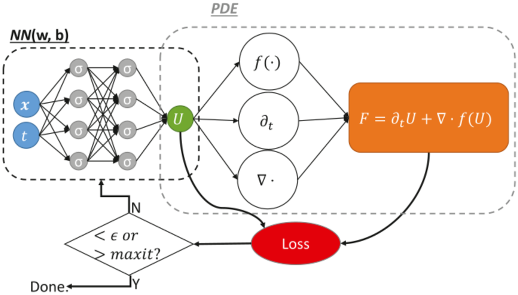
where,

$$U = (\rho, u, p)^T$$

$$A = \begin{bmatrix} u & \rho & 0 \\ 0 & u & \frac{1}{\rho} \\ 0 & \rho a^2 & u \end{bmatrix}$$

$$a = \sqrt{\gamma p/\rho}$$

- *Given* x, t $\rightarrow$ *Predict* ρ, u, p



Mao et al, 2019

Sod shock tube problem (Dirichlet BC):

$$U_L = [1.0, 0.0, 1.0]^T \qquad U_R = [0.125, 0.0, 0.1]^T$$

Loss function and training data

- In general, the standard loss function is defined as follows:

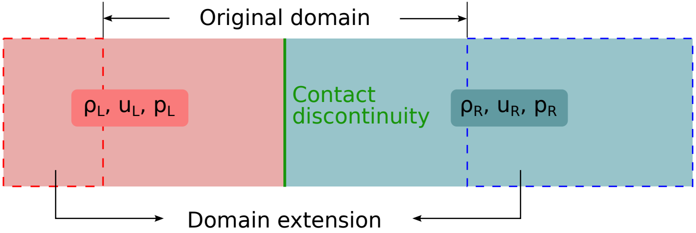
$$L(\theta) = \frac{1}{N_f} \left| \frac{\partial \tilde{U}}{\partial t}(x,t,\theta) + \tilde{A}\frac{\partial \tilde{U}}{\partial x}(x,t,\theta) \right|^2_{\Omega\times(0,T],\nu_1} +$$
$$\frac{1}{N_{IC}} \left| \tilde{U}(x,0,\theta) - U(x,0) \right|^2_{\Omega,\nu_2} + \frac{1}{N_{BC}} \left| \tilde{U}(x,t,\theta) - U(x,t) \right|^2_{\partial\Omega\times(0,T],\nu_3}$$

- Since the boundary conditions are induced by the initial conditions, the BC term is dropped, resulting in:

$$L(\theta) = L_f(\theta) + L_{IC}(\theta)$$

Domain Extension and Loss Weighting

- Domain extends towards the dominant direction of propagation (Papados, 2021)

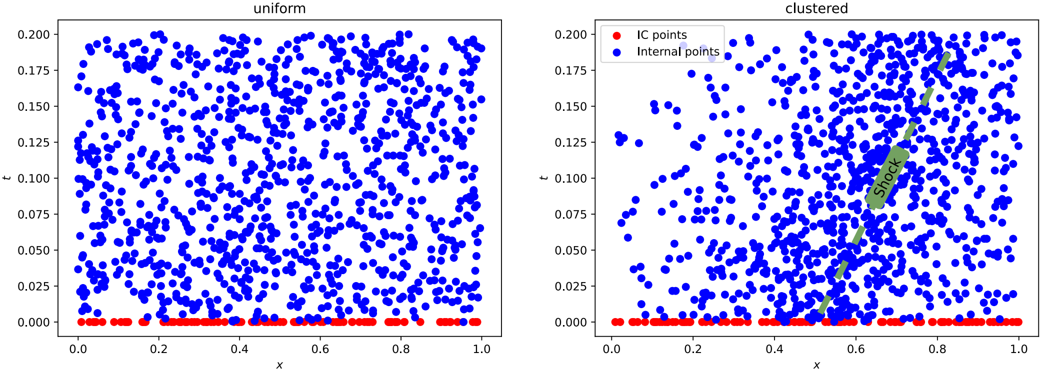


- Loss weighting: $L_f(\theta)$ decreases at a faster rate than $L_{IC}(\theta) \rightarrow$ The NN learns an arbitrary solution to the PDE and tries to adjust it at all interior training points and at all times (x_n, t_n) to fit the IC (Papados, 2021)

$$L(\theta) = w_f L_f(\theta) + w_{IC} L_{IC}(\theta)$$

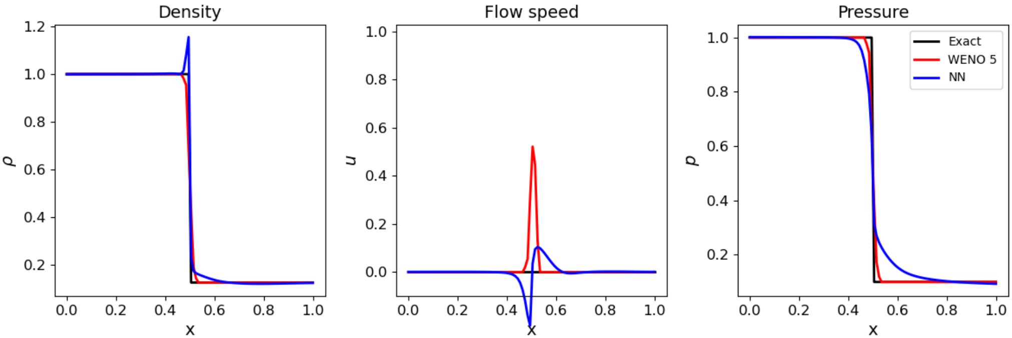
Uniform and Clustered Sampling

- A line equation, together with an exponential distribution, were used to randomly sample points near the shock location at any time

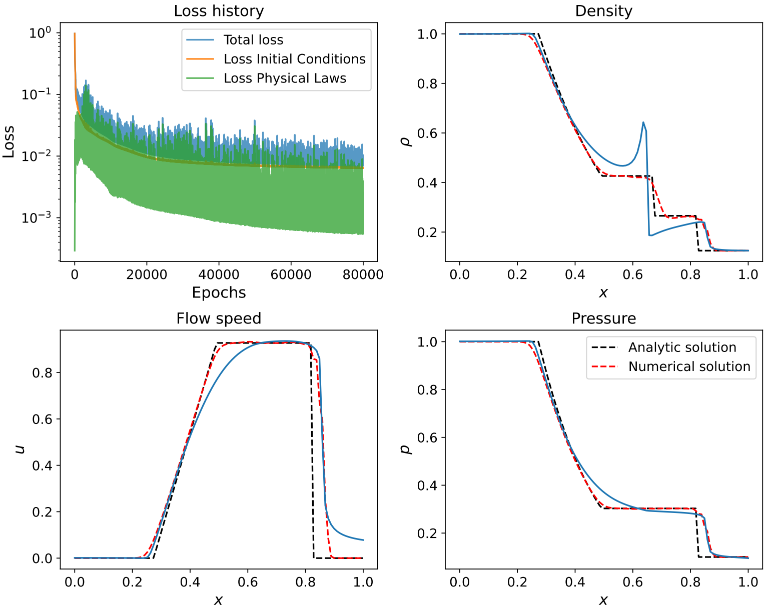


Results and Analysis - Original model

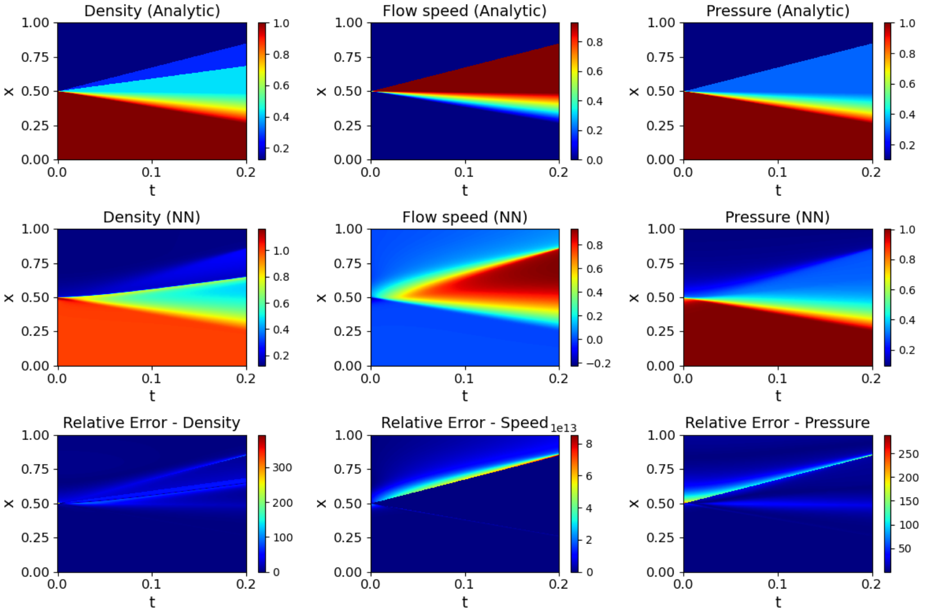
The models were trained by using a neural network (30-neuron width & 7 hidden layers), a learning rate of 0.0005 and the classical Adam optimizer.



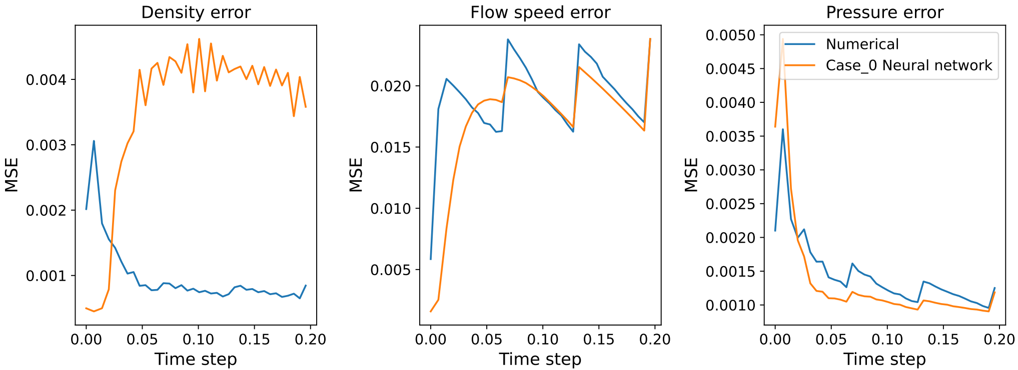
Results and Analysis - Original model



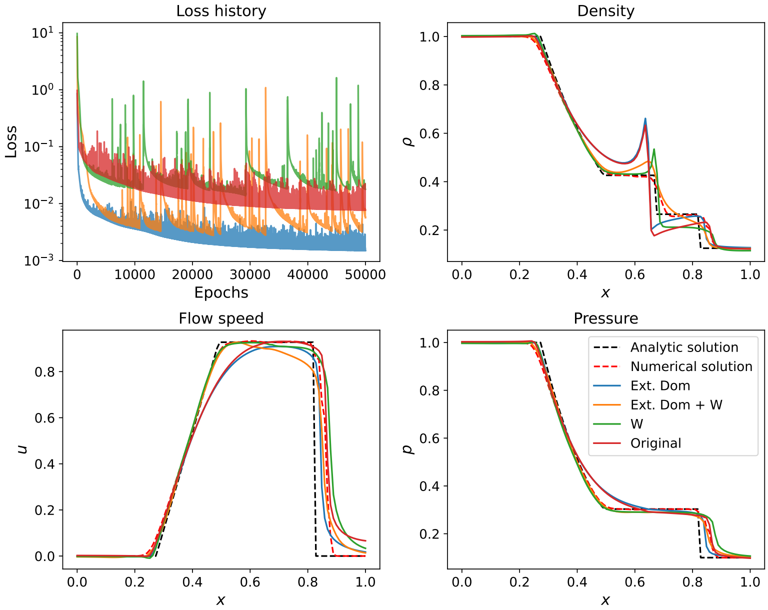
Results and Analysis - Original model



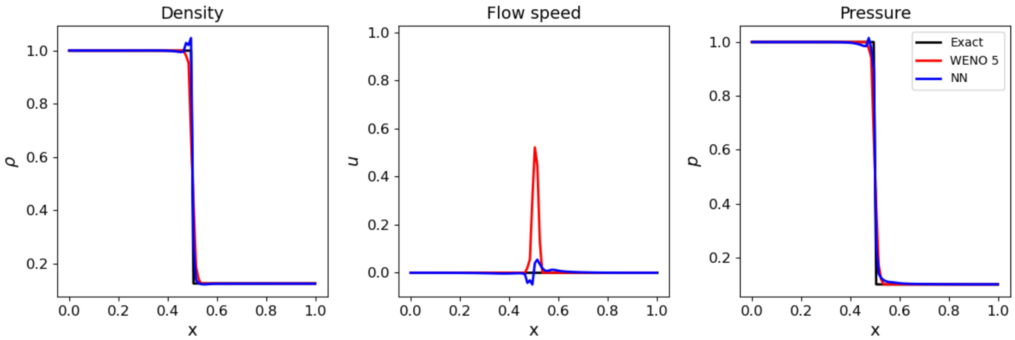
Results and Analysis - Original model



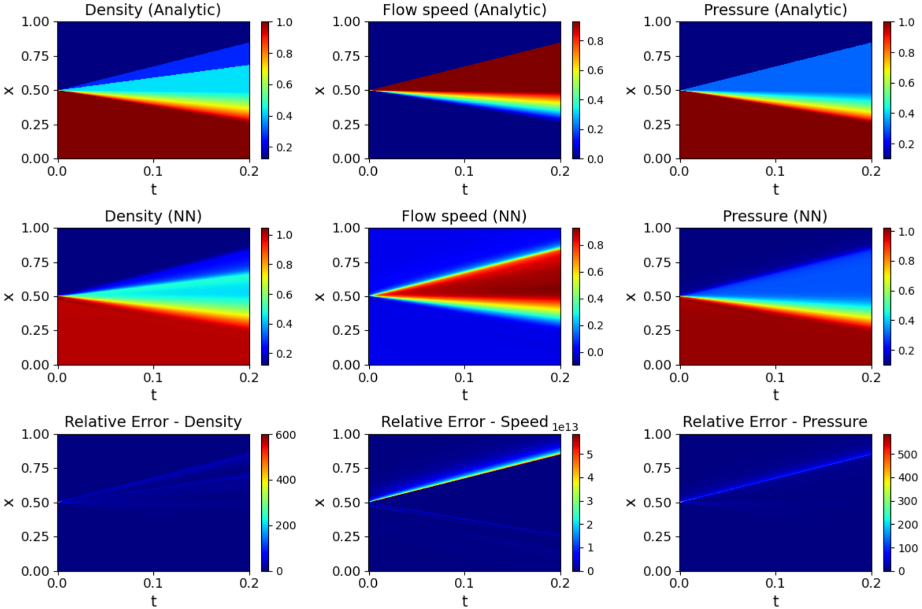
Results and Analysis - W. + Ext. Dom



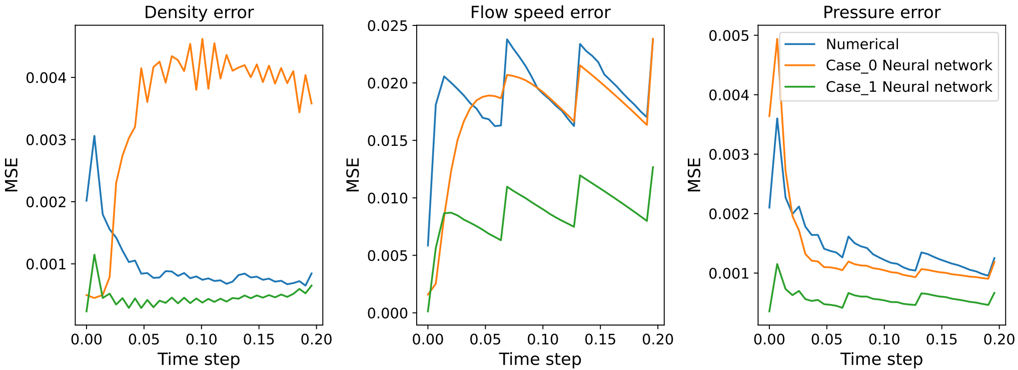
Results and Analysis - W. + Ext. Dom



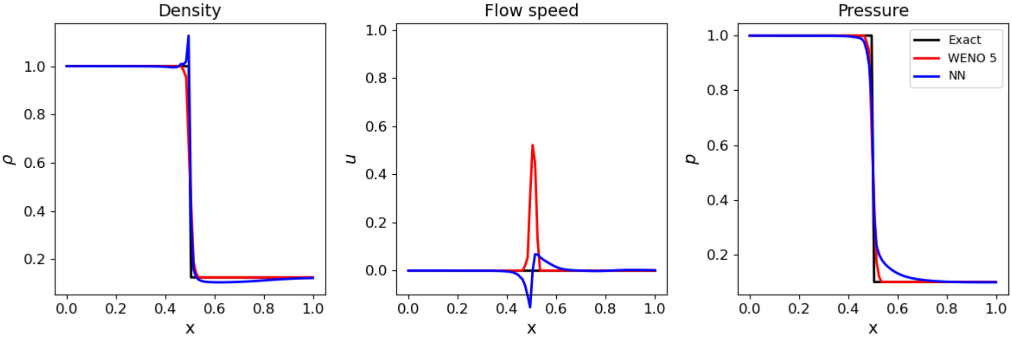
Results and Analysis - W. + Ext. Dom



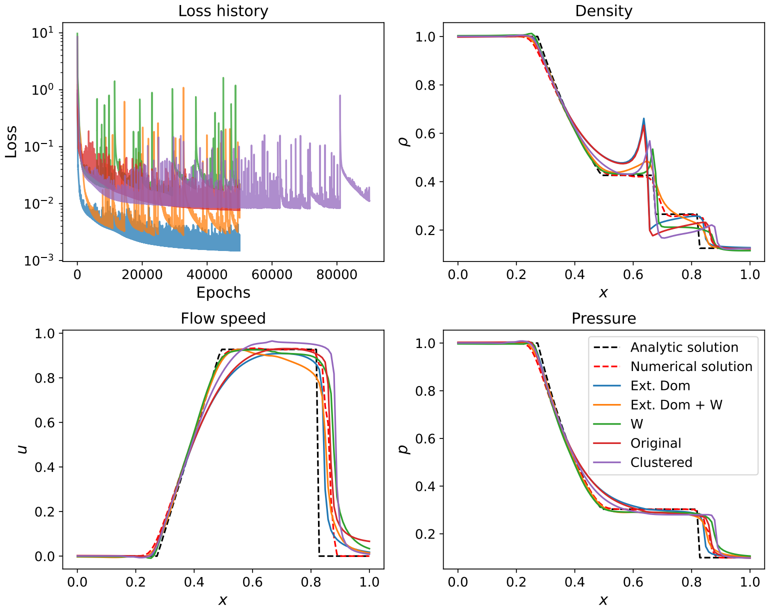
Results and Analysis - W. + Ext. Dom



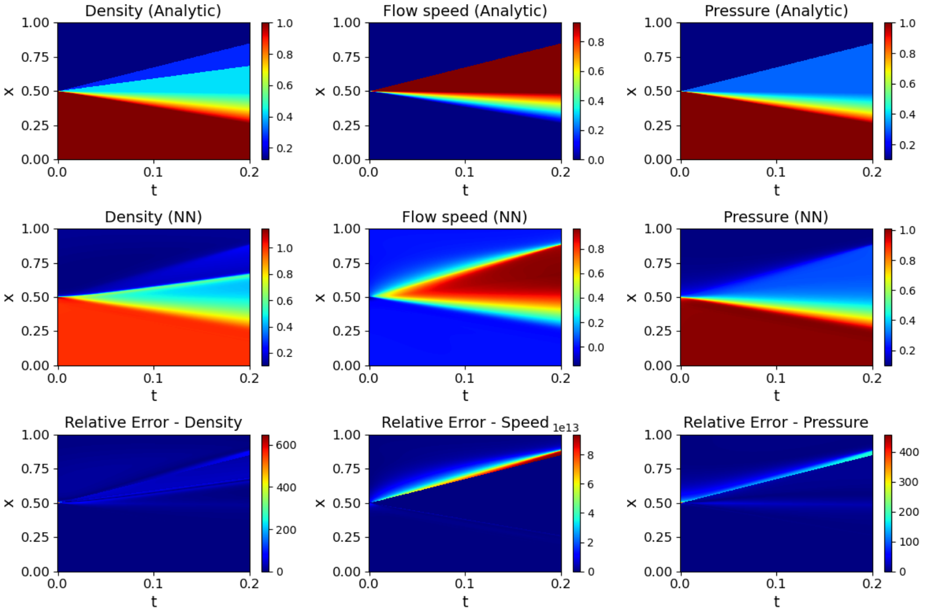
Results - Clustered Sampling



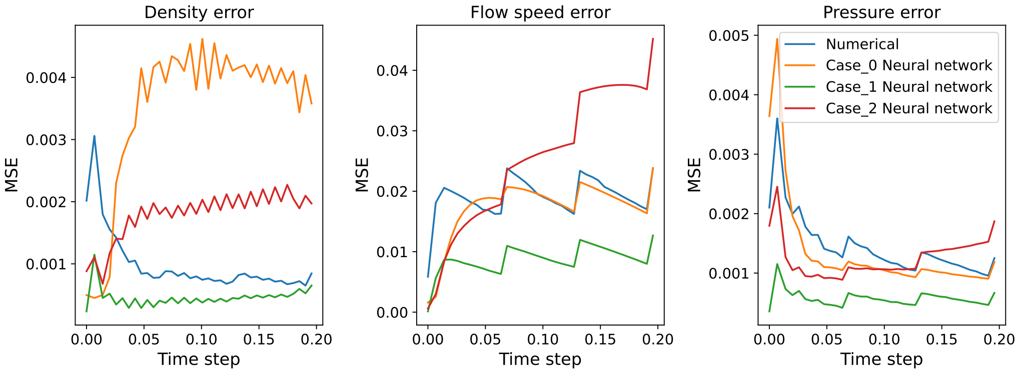
Results - Clustered Sampling



Results - Clustered Sampling



Results - Clustered Sampling



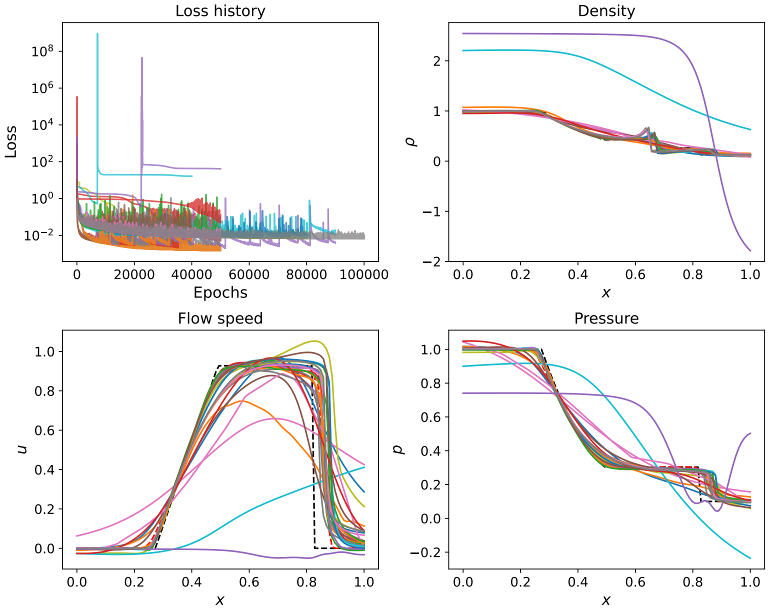
Computational cost comparison

- Is it a fair comparison against the performance of the classical numerical methods?

Method	Evaluation time ($T = 2$) [s]	
	$n_x = 100$	$n_x = 1000$
Analytic	0.07	0.52
WENO (5^{th} - RK45)	0.19	3.09

Case	Tr/epoch [s]	Tot. Tr. [min]	Eval. ($T = 2$) [s]
Nominal	175	2.11	0.008
Ext. Dom	129	1.55	0.007
Loss W.	85	1.02	0.009
Ext. Dom. + Loss W.	63	0.76	0.01
E.D. + L.W. + C.S.	162	1.95	0.015

Additional Results - Hyperparameter tuning



Summary and Conclusions

- A public repository link was setup to host the models implemented
- Hyperbolic PDEs highly depend on the initial conditions and its propagation parameters $\rightarrow$ Accuracy of NN increased if this learns the IC first (at higher rate).
- In all cases, the accuracy for the velocity and pressure is much higher than that for the density, which is due to the fact the velocity and pressure are smooth while the density has a contact discontinuity.
- Domain extension and loss weighting demonstrated an improvement in the prediction of the neural network
- Clustered sampling showed a better prediction when keeping the NN hyperparameters fixed but it highly depends on knowing where the discontinuity located beforehand.
- An informal hyperparameter tuning was carried out with no significant improvement in the prediction

Future Work

- Add artificial viscosity to rectify the oscillations near shock and contact discontinuities $\rightarrow$ it might be really expensive to train the model and it is considered a non-physical adjustment to the mathematical formulation
- Impose a *total variation diminishing* condition together with artificial viscosity into the NN
- Perform a formal hyperparameters tuning study where the paramters are set according to an optimization problem formulation and a gradient free or gradient descent method is employed $\rightarrow$ Heads up, this might be really expensive

References

1. Mao, Z., Jagtap, A. D., & Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. Computer Methods in Applied Mechanics and Engineering, 360, 112789
2. Papados, A. Solving hydrodynamic shock-tube problems using weighted physics-informed neural networks with domain extension.
3. Michoski, C., Milosavljević, M., Oliver, T., & Hatch, D. R. (2020). Solving differential equations using deep neural networks. Neurocomputing, 399, 193-212.