Improving performance of Physics-Informed Neural Networks performance for 1D Burgers

Lam (Max) Nguyen

1 Introduction

The Physics-Informed Neural Network (PINN) is a novel framework introduced in [1] as a low to no data machine learning model that is used to predict physical processes (often encoded as PDEs) numerically by incorporated the physics of the process into the training. The main advantage that PINNs offered is that they are continuous solvers meaning there will not be a need for domain partitioning procedure that is required by most traditional numerical models.

Despite this, there are limitations however, mostly stem from the Spectral Bias problem present with neural networks in general where the network will tend to learn low-frequency (smooth) functions much faster than high-frequency (rapidly varying) functions. Therefore, the repository will be an exploratory project on implementing not only baseline models for predicting behaviors of sharp gradient processes, but also examining research literatures for ways to improve upon the performance of PINNs for such processes as well.

2 Case Study

We will primarily investigate the baseline performance of PINN algorithm and improve upon that for the Burger's equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0 \iff u_t + uu_x - \nu u_{xx} = 0$$

where $\nu \to 0$ and. As noted in [1], for $\nu < 0.01$, the solution approaches the inviscid and the PINN algorithm will fail to converge to the correct solution, therefore we see the behavior for the value of $\nu = 0.001$. In particular, for benchmarking, we will compare the model's performance against the analytical solution within the spatial domain $x \in [-1, 1]$ and time-evolution $t \in [0, 1]$ which has initial and boundary conditions

$$u(x, 0) = -\sin(\pi x) \qquad u(-1, t) = u(1, t) = 0$$

the solution to this differential equation will have the sharp gradient and near-singularity around $x = 0$, so our goal is not only to mitigate loss around this line, but also globally improve accuracy on the whole domain as well.

Before we get into the implementation of the PINNs, it is worth noting how the analytical is derived and how we are going to capture this numerically for the project. Indeed, since the viscosity parameter $\nu > 0$, the Burger's equation considered is a parabolic PDE, therefore under the Cole-Hopf transformation (non-linear variable change) of the solution

$$u = -2\nu\frac{\partial}{\partial x}\ln\phi = -2\nu\frac{\phi_x}{\phi}$$

for some smooth function ϕ with the initial condition $\phi(x, 0) = \phi_0(x)$ then substituting this transformation back into the Burger's converts the solution into the heat equation

$$\frac{\partial \phi}{\partial t} = \nu\frac{\partial^2 \phi}{\partial x^2}$$

which can be solved explicitly via the heat kernel

$$\phi(x, t) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{4\pi\nu t}} e^{-\frac{(x-y)^2}{4\nu t}} \phi_0(y) dy$$

and of course, ϕ_x can be obtained by differentiating under the integral.

With that said, for representing this solution numerically, we will use the 20-points Hermite quadrature to approximate the solution to the Burgers that is based on [4] in which we will export the grid solution into a .csv file to compare with PINN predictions.

To keep things as balanced as possible, we will set a fixed seed and fixed the general deep MLP structure which have 2 input nodes (one for spatial and one for time variable) and 5 latent layers of 64 neurons each and one output node for the output as we are considering a solution to 1-dimensional Burger's equation.

3 Models and Improvements

3.1 PINN

Since we are solving a specific case of a differential equation of the form

$$\begin{aligned}
&\frac{\partial}{\partial t}u(\mathbf{x},t) + \mathcal{N}_{\mathbf{x}}(u(\mathbf{x},t)) = 0 \qquad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d \text{ and } t \in [0,T] \\
&u(\mathbf{x},t) = g(\mathbf{x},t) \qquad \text{for } \mathbf{x} \in \partial\Omega \text{ and } t \in [0,T] \\
&u(\mathbf{x},0) = u_0(\mathbf{x}) \qquad \text{for } \mathbf{x} \in \Omega
\end{aligned}$$

where Ω is the spatial domain in which points are representable as vectors $\mathbf{x}$ and $\mathcal{N}_x(\cdot)$ is the nonlinear differential operator on the domain. The goal is to construct an approximation model $u_{NN}(\mathbf{x},t,\mathbf{w})$ which is trained by incorporating the physics that is embedded into the PDE itself which is captured by minimizing the mean squared error (MSE) for the weights parameter $\mathbf{w}$ of the model

$$MSE(\mathbf{w}) = MSE_r(\mathbf{w}) + MSE_b(\mathbf{w}) + MSE_0(\mathbf{w})$$

where within the collocation (interior) points $\{\mathbf{x}_r^i, t_r^i\}_{i=1}^{N_r}$ in which the differential condition is enforced,

$$MSE_r(\mathbf{w}) = \frac{1}{N_r}\sum_{n=1}^{N_r}(u_{NN}(\mathbf{x}_r^i, t_r^i, \mathbf{w}))^2$$

similarly, on the boundary points $\{\mathbf{x}_b^i, t_b^i\}_{i=1}^{N_b}$,

$$MSE_b(\mathbf{w}) = \frac{1}{N_b}\sum_{n=1}^{N_b}(u_{NN}(\mathbf{x}_b^i, t_b^i, \mathbf{w}) - u(\mathbf{x}_b^i, t_b^i))^2$$

and on the points $\{\mathbf{x}_0^i\}_{i=1}^{N_0}$ in which the initial condition is enforced,

$$MSE_0(\mathbf{w}) = \frac{1}{N_0}\sum_{n=1}^{N_0}(u_{NN}(\mathbf{x}_0^i, \mathbf{w}) - u_0(\mathbf{x}_0^i))^2$$

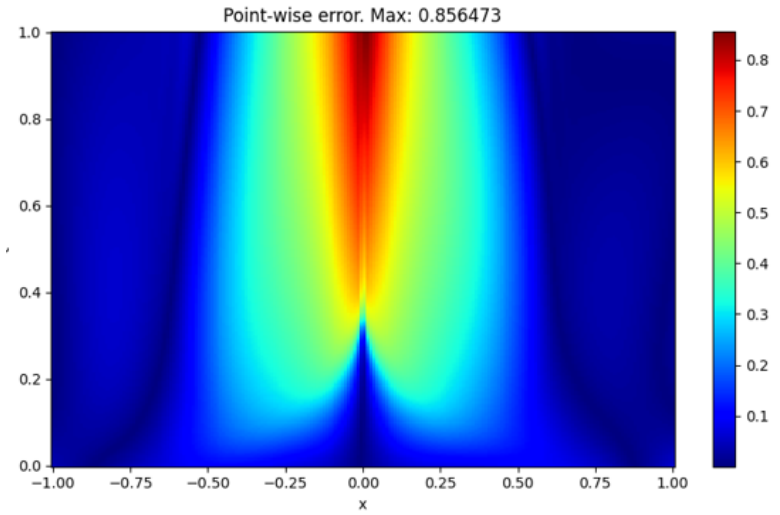
It is perfectly fine to take the MSE as the loss function for the model, but in practice there will be a component that will consistently outweighs the other errors, therefore we can balance this by resizing the weights based on the importance of types of errors encountered meaning we have better performance if we take the loss function as a linear combination of these MSE

$$\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \alpha_r\mathcal{L}_r(\mathbf{w}) + \alpha_b\mathcal{L}_b(\mathbf{w}) + \alpha_0\mathcal{L}_0(\mathbf{w}) \\
&= \alpha_r MSE_r(\mathbf{w}) + \alpha_b MSE_b(\mathbf{w}) + \alpha_0 MSE_0(\mathbf{w})
\end{aligned}$$

and derivatives of the network output needed to obtain the PDE residues are computed using automatic differentiation.

In our experimentation with the testing solution of the Burger's equation $u_t + uu_x = 0.001u_{xx}$ with initial and boundary conditions $u(-1,t) = u(1,t) = 0$, we were able to to find that the best parameters for the basic PINN algorithm to be $\alpha_r = 1$ and $\alpha_b = \alpha_0 = 10$. Despite of this, the vanilla algorithm is still very much lacking in prediction

capabilities as by plotting against the analytical, we see a big pointwise error margin



and with the mean absolute error (MAE) of about 0.1995.

3.2 Adaptive Viscosity (AV)

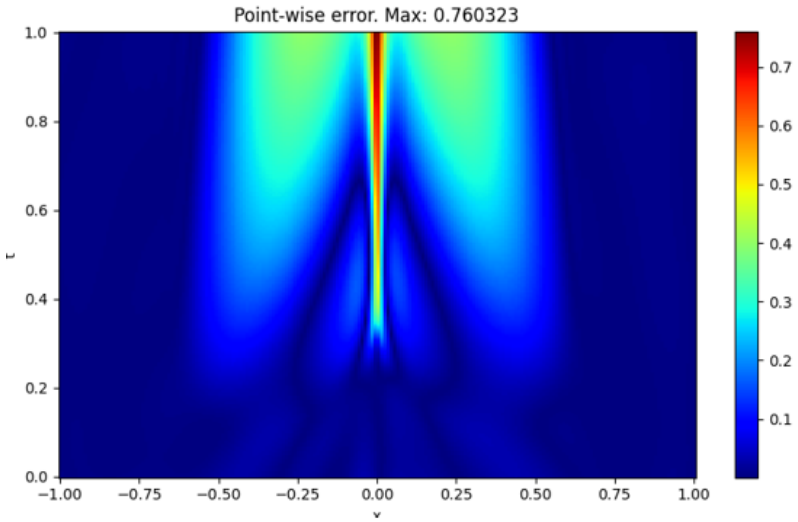
Countinho, et al in the paper [2] proposes an extension to PINN to mitigate this problem by adding artificial viscosity parameter to the training process of PINN. We will implementing a variation of this method by considering the modified residual

$$\mathcal{L}_r(\mathbf{w}, \nu_{max}, \mathbf{w}_\nu) = \frac{\partial}{\partial t}u(\mathbf{x}, t, \mathbf{w}) + u\frac{\partial}{\partial \mathbf{x}}u(\mathbf{x}, t, \mathbf{w}) - \nu_{max}\nu(\mathbf{x}, \mathbf{w}_\nu)\frac{\partial^2}{\partial \mathbf{x}^2}u(\mathbf{x}, t, \mathbf{w})$$

where $\nu(x, \mathbf{w}_\nu) = \exp(-\frac{x^2}{2\mathbf{w}_\nu^2})$ is used to update the adaptive viscosity learnable parameter ν_{max} by the optimizer. The loss function is also update to include the viscocity parameter ν_{max} and modified residual as

$$\mathcal{L}(\mathbf{w}, \nu_{max}, \mathbf{w}_\nu) = \alpha_r\mathcal{L}_r(\mathbf{x}, \mathbf{w}, \nu_{max}, \mathbf{w}_\nu) + \alpha_b\mathcal{L}_b(\mathbf{w}) + \alpha_0\mathcal{L}_0(\mathbf{w}) + \alpha_\nu\nu_{max}^2$$

With this setup, we were able to improve the performance to 0.0808 for mean absolute error and for pointwise and max error was reduced down to 0.7603. The best parameters that are used to give this result is $\alpha_r = 20, \alpha_b = \alpha_0 = 30$ and $\alpha_\nu = 360$



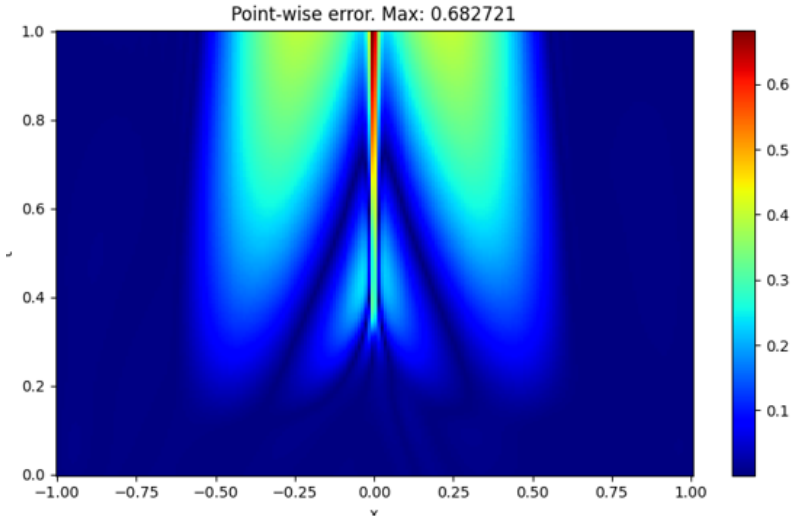
3.3 Attention

From here, we implemented improvements that can be apply to more general settings of PINN predictions for PDEs, mainly methods coming from [3]. The most notable of which is by adding attention mechanism within the feedforward

of the MLP network. With the model's activation function σ, at every input $\mathbf{x}$, we will create 2 encoders $U = \sigma(\mathbf{x}\mathbf{w}_U + b_U)$ and $V = \sigma(\mathbf{x}\mathbf{w}_V + b_V)$, denote the neuron at the ℓ-th layer as α^ℓ then the forward propagation becomes

$$\alpha^\ell(\mathbf{x}) = (1 - \sigma(\alpha^{\ell-1}(\mathbf{x}) + b_\ell)) \odot U + \sigma(\alpha^{\ell-1}(\mathbf{x}) + b_\ell) \odot V$$

the improvements of implementing the attention into the model is that the input variable is embedded into the hidden states of network itself via the encoders which effective addresses some limitations of the basic PINN algorithm. With that said, with the mechanism in place, we were able further reduces to mean square error to 0.0768 and maximum point-wise error to 0.683 with $\alpha_r = 2, \alpha_b = \alpha_0 = 20$ and $\alpha_\nu = 200$

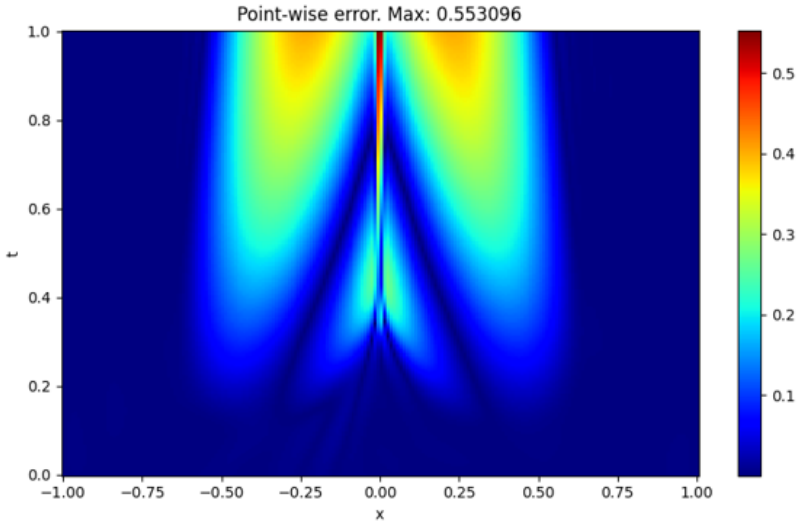


3.4 Adaptive weighting loss (AW)

Finally, we will further extend the loss function to the adaptive weight (AW) loss functions based on weight upper bounds of [3] where the trainable uncertainty parameters $\sigma_r, \sigma_b, \sigma_0$ for the residual, boundary condition and initial condition, respectively then introducing a weight upper bounds γ to prevent the diminishing of the uncertainty parameter to zero and constraining the magnitude of the weights as we transform the model's loss into the the negative log-likelihood function (for notational sake, we will just simply denote all relevant parameters as θ)

$$\mathcal{L}(\theta) = \alpha_\nu \nu_{max}^2 + \log(\nu_{max} + \gamma^{-1}) + \sum_{j \in \{r,b,0\}} \frac{\alpha_j}{\sigma_j^2 + \gamma^{-1}} (\mathcal{L}_j(\theta) + \log(\sigma_j^2 + \gamma^{-1})$$

then this give us a slight improvements in mean-square error down to 0.0757 while a boost in max pointwise error of 0.5531.



This model have the associated parameters of $\alpha_r = 2, \alpha_b = \alpha_0 = 20, \alpha_\nu = 0.001$ and $\gamma = 1000$. It is worth pointing out that the improvements that come from this adaptive weighting loss seems to be it seems to be concentrating to

get more correct predicting at the shock location $x = 0$ while lessen the attention to other regions so this might not be so much of an improvements rather a trade-off of accuracy.

4 Summary & Remarks

After iteratively improving on the model, here is the summary table of the performance through for each that is discussed

Model	Mean Absolute Error	Max pointwise error
Vanilla	0.1995	0.8565
AV	0.0809	0.7603
AV + attention	0.0768	0.6827
AV + attention + AW	0.0757	0.5531

therefore we see that the MAE is reduced by around 62% and Max pointwise error is reduced by 35% which is a significant result but there are still a noticeable error still present that our model still needs to improve upon, but fixing this issue is still an active area of research so hopefully, we will able to improve upon this.

Finally, a few remarks on how we can improve PINNs, first and most importantly, nothing beats knowing as much about the PDE properties, or domain knowledge about the process as possible since these will give more guidance into applying effective improvements to the model. Secondly, adding the (additive) attention mechanism is a simple way to improve performance of the MLP network structure. Another important point to not overlook is that of parameters tuning, by knowing which parts of the structure is more important, we can reflect its weights within the loss function, we can see a sizeable impact on the model's performance.

From here, there are multiple directions one can further take to improve upon the prediction of the model. The paper [5] suggest a variety of different ways to do so in which we have implemented a a few ways that is referenced. The most promising and key point that we have not tried is that of adaptive sampling where we sample points with a higher density around location of interests (sharp-gradients locations).

But one can not overlook the fact that MLP, despite being one of the cornerstone models of machine learning is just inadequate when it comes to predict and model complex processes as even though universal approximation theorem ensures theoretically the nonlinear capabilities of MLP, to capture this in practice would requires much more parameters among this, the model is susceptible to spectral bias problem as well. So to address these problems, recent advancements have introduced an alternative model to MLP which is that of Kolmogorov-Arnold network (KAN) which is based on the Kolmogorov-Arnold representation theorem [6]. There have already been extension to this models in literature already that seems to have success, namely that of incorporating Augmented-Lagrangians to the KAN models [7].

References

[1] M.Raissi, P. Perdikaris, G.E. Karniadakis. *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.* https://doi.org/10.1016/j.jcp.2018.10.045

[2] E. Coutinho, M. Dall'Aqua, L. McClenny, M. Zhong, U. Braga-Neto, E. Gildin. *Physics-Informed Neural Networks with Adaptive Localized Artificial Viscosity.* arXiv:2203.08802

[3] P. Niu, Y. Chen, J. Guo, Y. Zhou, M. Feng, Y. Shi. *Improved physics-informed neural network in mitigating gradient-related failures.* arXiv:2407.19421

[4] John Burkardt. *burgers_solution. Link.*

[5] J. Abbasi, A. Jagtap, B. Moseley, A. Hiorth, P. Andersen. *Challenges and Advancements in Modeling Shock Fronts with Physics-Informed Neural Networks: A Review and Benchmarking Study.* arXiv:2503.17379

[6] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, M. Tegmark. *KAN: Kolmogorov-Arnold Networks.* arXiv:2404.19756

[7] Zhang, Z., Wang, Q., Zhang, Y. et al. *Physics-informed neural networks with hybrid Kolmogorov-Arnold network and augmented Lagrangian function for solving partial differential equations.* Sci Rep 15, 10523 (2025). https://doi.org/10.1038/s41598-025-92900-1