ACTIVATION FUNCTION TRADE-OFFS FOR TRAINING EFFICIENCY OF PHYSICS-INFORMED NEURAL NETWORKS USED IN SOLVING 1D BURGERS' EQUATION

AUTHORS
Rareș Mihail

SUPERVISORS
Dr. Jing Sun
Dr. Alexander Heinlein
Dr. Tiexing Wang

INTRODUCTION

- Physics-Informed Neural Networks (PINNs) are a class of machine learning models that integrate physical laws expressed as partial differential equations (PDEs) into the neural network training process.
- PINNs have gained significant attention due to their ability to solve forward (estimating the solution to a governing mathematical model) and inverse (learning the mathematical model's parameters from observed data) problems in scientific computing, where traditional numerical methods such as the Finite Difference Method often struggle with high-dimensional spaces or ill-posed problems.
- The equation to solve for is the **viscous 1D Burgers' Equation**:

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

OBJECTIVE

Which activation function best optimizes the accuracy and training efficiency of Physics-Informed Neural Networks for solving the 1D Burgers' Equation?

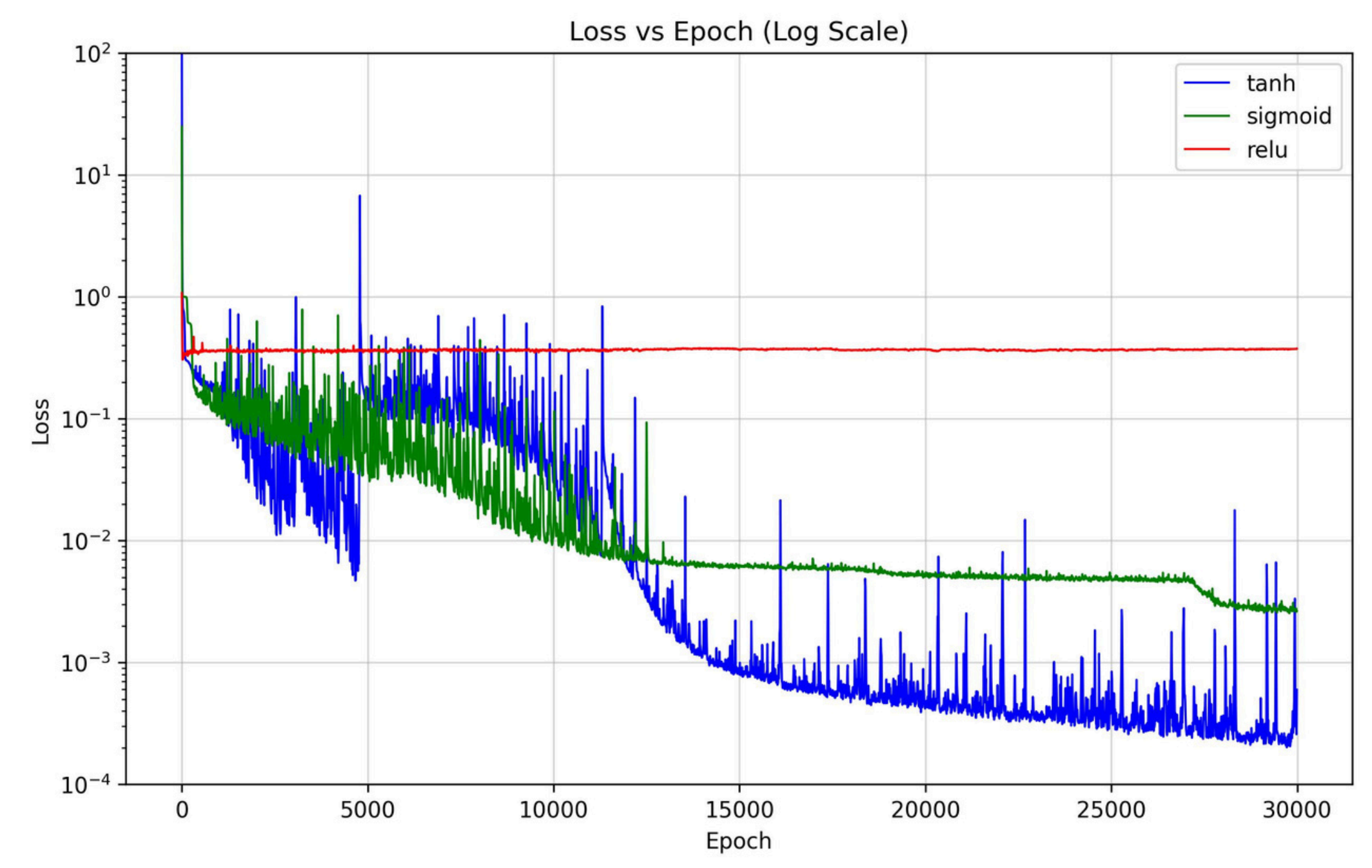
To address this overarching question, the study explores the following sub-questions:

1. How do common activation functions (ReLU, tanh, sigmoid) perform in terms of accuracy and training time?
2. Can adaptive or learnable activation functions improve training dynamics compared to static ones?
3. How do improved activation functions compare to one another in terms of training speed and accuracy?

RESULTS

Comparison of traditional activation functions:

- tanh outperforms sigmoid and ReLU.
- tanh achieves a 2.3x speedup in convergence compared to sigmoid.
- tanh achieves a minimum error over 100x smaller than sigmoid, and 1600x smaller than ReLU



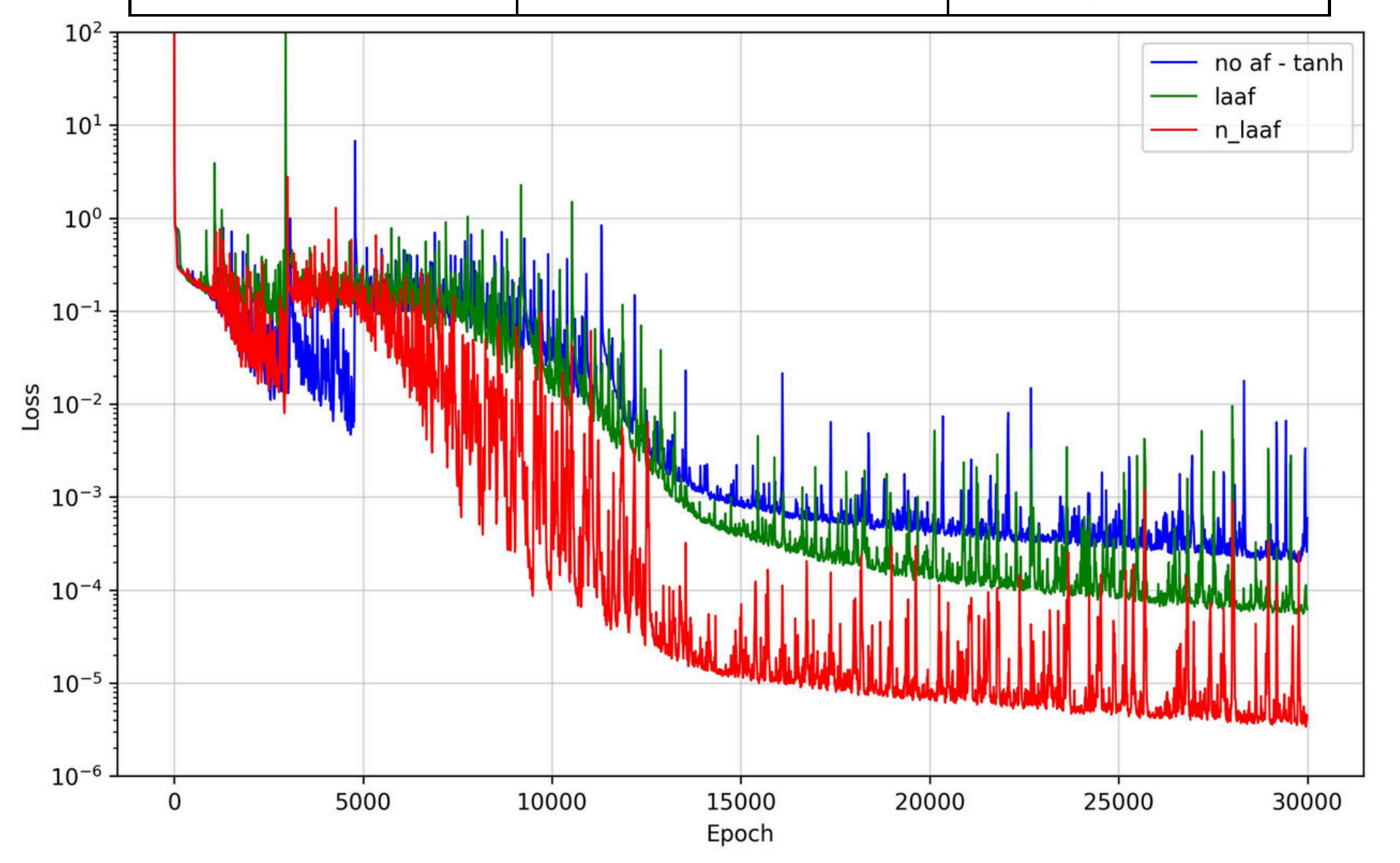
Comparison of adaptive activation functions and tanh:

- Both adaptive functions outperform non-adaptive tanh in accuracy and convergence speed.
- N-LAAF achieves an error threshold (1e-5) 3x faster than tanh and 2x faster than LAAF.
- N-LAAF achieves a minimum error over 100x smaller than tanh, and 61x smaller than LAAF.

Resource trade-offs for tanh, LAAF and N-LAAF

- Training time: N-LAAF takes 21% longer than tanh and 7% longer than LAAF.
- Memory usage: N-LAAF uses 10% more memory than LAAF and 20% more than tanh.

Method	time elapsed	RAM used
no-AF-tanh	1h48m02s	20,79GB
LAAF	2h2m9s	22,85GB
N-LAAF	2h11m47s	25,20GB



CONCLUSION

The improved performance of the adaptive activation functions can be attributed to two phenomena:

- The enhanced performance of N-LAAF is largely attributed to its fine-grained adaptability, enabled by neuron-wise trainable parameters. This adaptability allows the network to better capture localized features, such as sharp gradients and subtle variations in smooth regions, which are characteristic of solutions to the 1D Burgers' equation.
- Spectral bias[2], the tendency of neural networks to prioritize learning low-frequency components, poses challenges in modeling high-frequency regions with steep gradients like those near x = 0 in the 1D Burgers' equation, but N-LAAF's additional trainable parameters effectively mitigate this, enabling superior handling of complex dynamics and reducing error accumulation compared to LAAF and tanh.

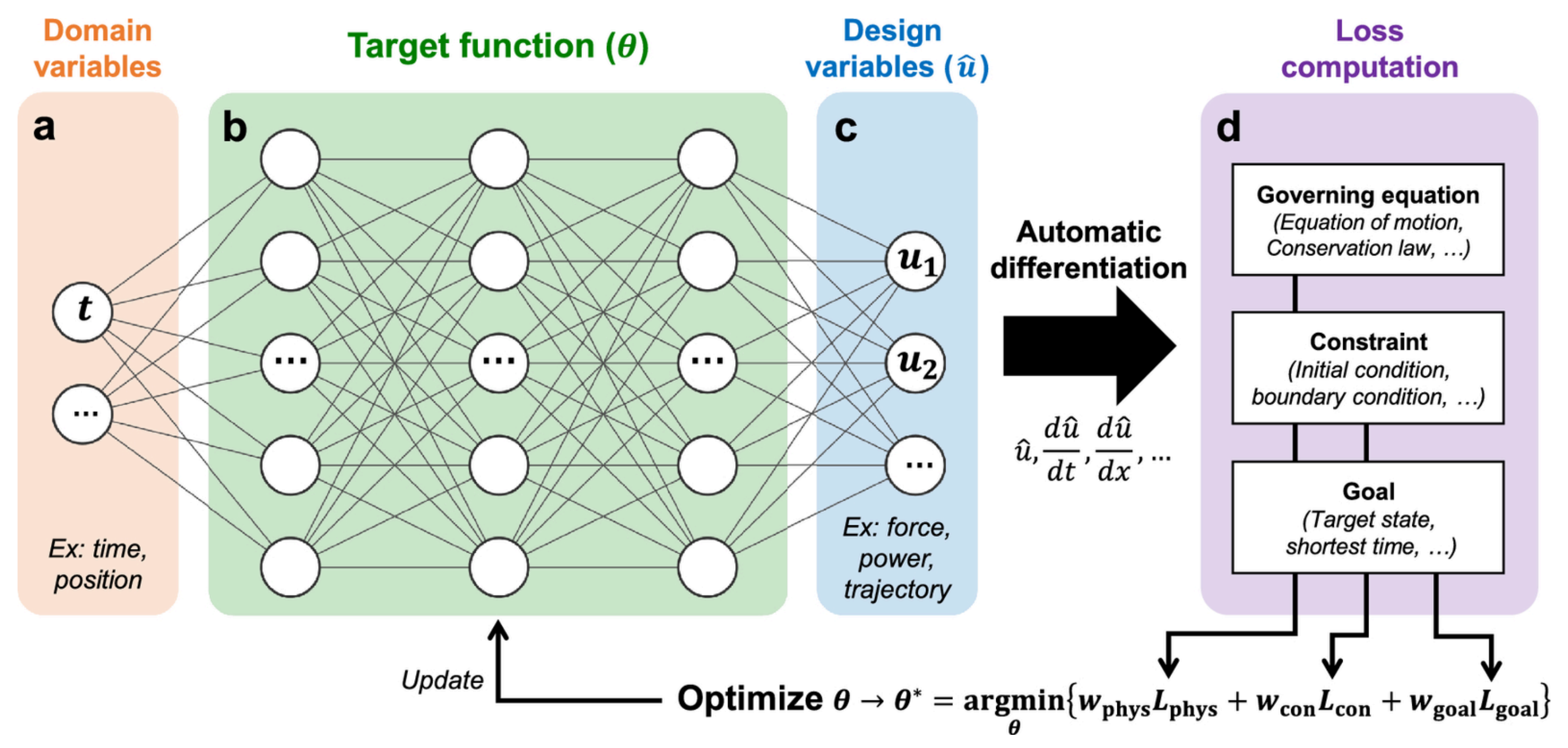
METHODOLOGY

The process of training PINNs is very similar to that of vanilla NN training, with one important distinction, **the loss function**, which is comprised of 3 weighted terms:

1. the **physics loss**, ensuring the solution matches the 1D Burgers' Equation.
2. the **boundary condition loss**, ensuring the solution matches the boundary data.
3. the **initial condition loss**, ensuring the solution matches the initial data.

These terms are part of the equation that defines the PINN's total loss:

$$L(\theta) = \lambda_1 \cdot L_{\text{PDE}}(\theta) + \lambda_2 \cdot L_{\text{BC}}(\theta) + \lambda_3 \cdot L_{\text{IC}}(\theta)$$


Domain variables
Target function (θ)
Design variables ($\tilde{u}$)
Loss computation
a
b
c
d
t
Ex: time, position
u_1
u_2
Ex: force, power, trajectory
Automatic differentiation
$\hat{u}, \frac{d\hat{u}}{dt}, \frac{d\hat{u}}{dx}, \dots$
Governing equation
(Equation of motion, Conservation law, ...)
Constraint
(Initial condition, boundary condition, ...)
Goal
(Target state, shortest time, ...)
Update
Optimize $\theta \rightarrow \theta^* = \underset{\theta}{\operatorname{argmin}}\{w_{\text{phys}}L_{\text{phys}} + w_{\text{con}}L_{\text{con}} + w_{\text{goal}}L_{\text{goal}}\}$

In order to generate results, the PINN was trained for 30000 epochs using Mini-Batch Gradient Descent[3] and each of the proposed activation functions:

1. hyperbolic tangent
2. sigmoid
3. ReLU (rectified linear unit)
4. layer-wise locally adaptive activation function (LAAF)
5. neuron-wise locally adaptive activation function (N-LAAF)

The last two activation functions have been proposed in Kawaguchi et. al. 2020[1], and are **adaptive** activation functions[4], as they incorporate trainable parameters in each layer/neuron, respectively.

TU Delft

[1]Kawaguchi, Kenji, Ameya D. Jagtap, and George Em Karniadakis (2020). "Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks". Journal of Computational Physics.
[2]Rahaman, Nasim et al. (2019). "On the Spectral Bias of Neural Networks". Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research 97, pp. 5301–5310.
[3]Masters, Dominic and Carlo Luschi (2018). "Revisiting Small Batch Training for Deep Neural Networks". arXiv preprint arXiv:1804.07612. URL: https://arxiv.org/abs/1804.07612.
[4]Jagtap, Ameya D., Kenji Kawaguchi, and George Em Karniadakis (2020). "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks". Journal of Computational Physics.