Analyzing the Impact of Adaptive Weighting in Self-Adaptive Physics-Informed Neural Networks for Solving PDEs

Jakub Mańkowski

Supervisor(s): Dr. Jing Sun, Dr. Alexander Heinlein, Dr. Tiexing Wang (Shearwater GeoServices, UK)

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 26, 2025

Name of the student: Jakub Mankowski
Final project course: CSE3000 Research Project
Thesis committee: Dr. Jing Sun, Dr. Alexander Heinlein, Dr. Tiexing Wang, Dr. Hayley Hung

Abstract

Self-Adaptive Physics-Informed Neural Networks (SA-PINNs) are a variation of traditional Physics-Informed Neural Networks (PINNs) designed to solve the challenges of solving "stiff" partial differential equations (PDEs). By using adaptive weighting, SA-PINNs are able to focus their attention on areas of the domain with higher errors, therefore improving accuracy. This work investigates the roles of individual loss components, namely residuals, boundary conditions, and initial conditions, in the performance of SA-PINNs.

1 Introduction

Physics-Informed Neural Networks (PINNs) are a big advancement in solving partial differential equations (PDEs) that at their core work by incorporating physical laws into the neural network architecture [1; 2]. Proposed by [1], and therefore from now on referred to as "baseline" PINNs, they aim to minimize the error of the PDE, initial, and boundary conditions simultaneously, effectively embedding domain knowledge into the learning process [3]. Although PINNs have proven useful in modeling a wide variety of PDEs, they are known to struggle with equations that are "stiff", that means, with solutions characterized by sharp spatial transitions or fast time evolution [4].

Self-Adaptive Physics-Informed (SA-PINNs) introduced by [3] have emerged as a significant advancement over baseline PINNs aimed to solve the problem with convergence when predicting stiff nonlinear equations. Such equations are often encountered in fields such as fluid dynamics and material science. An example of a stiff equation is an Allen-Cahn equation that struggles to converge when traditional PINN is used [5]. SA-PINNs address these challenges by applying fully trainable adaptation weight mask to each training point, which allows the network to autonomously focus on challenging regions of the solution. The multiplicative soft attention mask is the core innovation of SA-PINNs and it works by introducing adaptation weights λ_r, λ_b, λ_0 that are learned for the residual, boundary, and initial condition terms, respectively. These weights are updated alongside the neural network parameters during training through gradient descent/ascent, allowing the model to focus more on areas with larger error. This approach has shown to improve both convergence rates and accuracy when compared to baseline PINNs [3]. For example, when applied to nonlinear equations such as the Allen-Cahn equation, SA-PINNs have demonstrated superior performance in terms of both training efficiency and solution accuracy [3].

Mathematically, the loss function for SA-PINNs is defined as follows:

$$L(w, \lambda_r, \lambda_b, \lambda_0) = L_r(w, \lambda_r) + L_b(w, \lambda_b) + L_0(w, \lambda_0) \quad (1)$$

Where:

- $L_r(w, \lambda_r)$ corresponds to the error caused by not satisfying the PDE, weighted by λ_r,
- $L_b(w, \lambda_b)$ is the error caused by not satisfying the boundary conditions, weighted by λ_b,
- $L_0(w, \lambda_0)$ represents the error caused by not satisfying the initial condition, weighted by λ_0.

The self-adaptation weights are vectors with dimensions equal to the number of points in the training set for each of the components:

- Residue points weights: $\lambda_r = (\lambda_r^1, ..., \lambda_r^{N_r})$
- Boundary points weights: $\lambda_b = (\lambda_b^1, ..., \lambda_b^{N_b})$
- Initial points weights: $\lambda_0 = (\lambda_0^1, ..., \lambda_0^{N_0})$

They are trainable parameters that adjust based on the gradient of the loss with respect to the respective component. The loss is minimized with respect to the network weights w, and maximized with respect to the adaptation weights λ_r, λ_b and λ_0 forming this problem:

$$\min_{w} \max_{\lambda_r, \lambda_b, \lambda_0} L(w, \lambda_r, \lambda_b, \lambda_0) \quad (2)$$

The gradients with respect to the adaptation weights are computed during each iteration of training, as detailed by the gradient update rules:

$$w^{k+1} = w^k - n_k \nabla_w L(w_k, \lambda_r^k, \lambda_b^k, \lambda_0^k) \quad (3)$$

$$\lambda_r^{k+1} = \lambda_r^k + p_r^k \nabla_{\lambda_r} L(w_k, \lambda_r^k, \lambda_b^k, \lambda_0^k) \quad (4)$$

$$\lambda_b^{k+1} = \lambda_b^k + p_b^k \nabla_{\lambda_b} L(w_k, \lambda_r^k, \lambda_b^k, \lambda_0^k) \quad (5)$$

$$\lambda_0^{k+1} = \lambda_0^k + p_0^k \nabla_{\lambda_0} L(w_k, \lambda_r^k, \lambda_b^k, \lambda_0^k) \quad (6)$$

Where $n^k > 0$ and $p^k > 0$ are learning rates for network weights and the adaptation weights, respectively. The adaptation weights λ_r, λ_b and λ_0 are updated using gradient ascent to maximize the loss with respect to the adaptive weight, while the neural network weights w are updated using standard gradient descent. This approach dynamically adjusts the loss components during training, leading to improved convergence rates with fewer epochs compared to traditional PINNs.

The adaptation weights are initialized either randomly on a specified interval, or based on prior knowledge of the problem. For example, if initial conditions are known to be particularly difficult to fit, the weight λ_0 could be initialized larger than the others.

In contrast to traditional PINNs, which use fixed weights across the solution domain, SA-PINNs enable dynamic focusing on harder-to-train regions, offering a flexible and adaptive learning process [3]. The adaptive masks allow SA-PINNs to handle more challenging problems with better efficiency [3]. This has been demonstrated in various experiments, such as those conducted by [3], where SA-PINNs outperformed traditional PINNs in terms of accuracy and convergence speed on stiff problems.

In addition to theoretical improvements, SA-PINNs

have demonstrated practical utility in real-world scenarios. [6] applied SA-PINNs in a real-world scenario to model seismic wave propagation in complex topography. They demonstrated the effectiveness of the adaptive approach, showing improved accuracy and scalability in comparison to traditional methods.

Despite the success of SA-PINNs in various applications, there is still a significant gap in understanding the specific components that contribute to their improved performance. Although the adaptive weighting mechanism has been shown to enhance the training process [3], the exact role of the individual loss components, such as the violation of the PDE, the satisfaction of boundary conditions as well as the satisfaction of starting conditions, remains mostly unexplored. This is particularly relevant in real-world scenarios where maximizing training efficiency is crucial for practical applications of SA-PINNs.

The goal of this study is to investigate the role of adaptive weighting masks in SA-PINNs, with a question on how these masks influence the accuracy of model, as well as the time it takes to train. The main research question is: Which loss components contribute most significantly to the solution accuracy and how do the adaptive weights affect this contribution? To find the answer to that question, this study will systematically turn off the masks for different loss components and evaluate their impact on the solution of the Viscous Burgers equation, a well-known stiff nonlinear PDE described as:

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, x \in [-1, 1], t \in [0, 1], \quad (7)$$

$$u(0, x) = -\sin(\pi x), \quad (8)$$

$$u(t, -1) = u(t, 1) = 0. \quad (9)$$

By systematically examining the effects of these masks on the performance of SA-PINNs, this study will attempt to identify key factors that could enhance the training efficiency and overall accuracy.

Understanding the connection between the adaptive masks and the loss components will provide deeper insights into the mechanics of SA-PINNs and may lead to a discovery of a more efficient and scalable training strategies. These insights could be useful for both theoretical advancements in machine learning and practical applications of SA-PINNs.

2 Methodology

This section details the experimental setup and methodology used to analyze the performance of SA-PINNs in solving the Viscous Burgers equation. The experiments are designed to assess how adaptive weights influence accuracy and of the model as well as training time efficiency. The network architecture is the same setup as [1] uses for the baseline PINN. It has been done this way, so direct comparisons are possible, if necessary. The SA-PINN hyperparameters and overall setup is based on [3], since the main goal of this study is not to optimize hyperparameters, a previously validated configuration was chosen to ensure reliable results. Code used for this research is a modified version of code from [3].

2.1 Neural Network Architecture

The neural network used in all experiments follows a feedforward structure with the following architecture:

- Input layer: 2 neurons corresponding to x and t inputs
- Hidden layers: 8 fully connected layers each containing 20 neurons with the hyperbolic tangent (tanh) activation function.
- Output layer: 1 neuron, representing the approximated solution $u(x, t)$ without applying any activation function.

The hyperbolic tangent activation function (tanh) is used in hidden layers because it can effectively capture nonlinear relationships [7]. The output layer applies no activation function, making it a linear layer that outputs raw numerical values, corresponding to the solution of the equation we want to predict. In the case of this study, it is the Viscous Burgers equation.

2.2 Training process

Each model undergoes two stages of optimization first using Adam [8] and then L-BFGS [9]:

- Adam optimizer: The model is trained for the specified number of epochs using the Adam optimizer.
- L-BFGS Optimizer: The model is trained for an equal number of epochs using the L-BFGS optimizer to fine tune the network weights.

For example, in a scenario with 1000 epochs, the model is trained with Adam for 1000 epochs followed by L-BFGS for another 1000 epochs. The model is saved after the completion of both stages. It is worth mentioning that adaptive weights are only updated when training with Adam, and they are held constant when training with L-BFGS, as done in [3].

Both the network and adaptive weights are updated using a learning rate of 0.005, applied as described in equations (3)-(6). The adaptive weights for all masks are initialized randomly within the interval (0, 1) to ensure no prior knowledge about the problem is introduced. This initialization approach allows for a more generalized and unbiased analysis of the adaptive weighting mechanisms.

2.3 Scenarios and Experimental Setup

To investigate the impact of adaptive weights on training, five scenarios are evaluated for each epoch category (100, 1000, and 10000 epochs). The scenarios are as follows:

- All Masks Off: Adaptive masks are disabled for all components. This mimics the baseline PINN.
- All masks On: Adaptive masks are enabled for all components. This is a regular SA-PINN implementation.
- Residual Mask On: The adaptive mask is enabled for the PDE residual component, the remaining components have their adaptive weightings disabled.
- Boundary Condition Mask On: Adaptive mask is enabled for the boundary component, the remaining components have their adaptive weightings disabled.

- Initial Condition Mask On: Adaptive mask is enabled for initial component, the remaining components have their adaptive weightings disabled.

For each scenario, five independent models are trained to ensure robustness and reduce variability. Then L2-error and time are averaged and reported. Standard deviations are reported as well.

2.4 Data

- Initial Condition Points (N_0): 100 points are sampled so 50 per boundary.
- Boundary Points (N_b) 200 points are sampled.
- Collocation Points (N_r): 10000 points are sampled uniformly from the dataset.

While other combinations of numbers of points could be explored, this is the setup used by [3] and it was proven to be effective. The points for initial and boundary conditions were sampled randomly. Collocation points were sampled using Latin Hypercube Sampling (LHS) [10] to ensure more uniform coverage of the input points. The data used for training the models was taken from [1].

2.5 Evaluation Metrics

The performance of each scenario is evaluated based on:

- L2-Error: The L2 normal of the difference between the predicted and exact solution. It is defined as follows:

$$\text{L}_2 \text{ error} = \frac{\sqrt{\sum_{i=1}^{N_U} |u(x_i, t_i) - U(x_i, t_i)|^2}}{\sqrt{\sum_{i=1}^{N_U} |U(x_i, t_i)|^2}} \tag{10}$$

  where $u(x, t)$ is the trained approximation, $U(x, t)$ is a reference solution obtained on a detailed grid $\{x_i, t_i\}$ consisting of N_U points.
- Time: The training duration is recorded in seconds after each training session.

L2-error directly translates to the accuracy of the model, while the time it took to train it allows measure what configurations are the fastest ones to train. These two benchmarks might help in finding the optimal balance between accuracy and time to train depending on specific needs.

2.6 Tools and Implementation

The experiments are implemented using Python, with TensorFlow 2.3. The exact code can be found on GitHub under this link: https://github.com/hope-I-can-change-it/research-project

3 Results

As described in the previous section, the experiments were split into five different scenarios. In the first scenario, all adaptive masks were turned off, mimicking the baseline PINN. In the next three scenarios, adaptive masks were turned on for different loss components one by one. In the last scenario all masks were turned on simultaneously. Figures (1)-(3) illustrate the average L2-error for each configuration across 100, 1000, and 10000 epochs of training. The standard deviations of the L2-errors are also included in the graphs to provide insights into the consistency of the models across different configurations and training durations. Additionally, Figure (4) illustrates the corresponding average training times for the 10000-epoch experiments, as the differences for shorter runs were not significant.

It was observed that sometimes the model failed to train entirely. This was characterized by the loss stopping to decline or even starting to rise after a number of epochs, for the rest of the training. This was observed in both Adam and L-BFGS training phases. Since the problem appeared to happen randomly and was not connected to enabling or disabling the adaptive masks, the affected runs were repeated to ensure that each scenario included five runs for calculating the average results.

3.1 PDE Residual Mask Performance

Turning only the PDE residual mask showed mixed results depending on the number of training epochs. For short training durations (100 epochs), enabling the PDE residual mask led to slightly improved performance compared to the traditional PINN. The performance gain was very small, and it could change if the average from more runs was gathered. However, for longer training durations (1000 and 10000 epochs), this mask significantly degraded performance.

- 1000 epochs: The average L2-error was over 16 times higher than the traditional PINN.
- 10000 epochs: The error remained high, approximately 6 times greater than the traditional PINN.

This suggests that focusing only on residual errors without incorporating adaptive weights for boundary and initial conditions is not only insufficient to achieve good performance, but it also gives worse performance than just leaving all of the masks turned off. The average training time for the PDE residual mask configuration was recorded as 617.26 ± 3.34 seconds, making it 304.54 seconds more than the baseline PINN, indicating a significant increase.

3.2 Initial Conditions Mask Performance

While having only initial conditions mask on turned on when training for 100 epochs yielded the worst performance from all 100 epoch scenarios, for 1000 and 10000 epochs turning it on always resulted in a better performance than the baseline PINN.

- 1000 epochs: The model achieved approximately 2.5 times better performance compared to the baseline PINN.
- 10000 epochs: The improvement, while smaller, still outperformed the baseline. The improvement was very small however and this result could change if the average from more runs would be gathered.

These results highlight the effectiveness of the initial conditions mask in ensuring stability and convergence, especially for medium to long training periods. However, the training

time for this configuration was 392.19 ± 29.90 seconds, which is 79.47 seconds higher than the baseline PINN.

3.3 Boundary Conditions Mask Performance

Turning on the boundary conditions mask resulted in marginal improvements, only noticeable during short training runs (100 epochs). The improvement was relatively small compared to the initial conditions mask and could vary if average from more runs was collected.

- 100 epochs: Slightly better performance than the baseline PINN.
- 1000 epochs: Slightly worse performance than the baseline PINN.
- 10000 epochs: The model achieved approximately 2.5 times worse performance compared to the baseline PINN.

The results show that turning on this mask alone for training is not worth it in most cases, although for very short training periods it may result in better performance than the baseline PINN. The training time for this configuration was 384.11 ± 12.32 seconds, so 79.47 seconds higher than the baseline PINN.

3.4 Full SA-PINN Configuration

When all adaptive masks were enabled simultaneously, the model consistently outperformed the baseline PINN across all training durations. That aligns with findings in [3] and clearly shows the positive impact adaptive masks have on the training process. However, the training time increased significantly to 635.94 ± 9.42 seconds, making it 323.22 seconds higher than the baseline PINN.

Further analysis in the discussion section will explore the trade-offs between accuracy and computational efficiency.

4 Responsible Research

This section discusses the ethical aspects and reproducibility of the methods described in the paper. Responsible research ensures that the results are not only scientifically valid but also ethically sound and reproducible.

4.1 Ethical Aspects

The data used in this study is based on the Viscous Burgers equation. This equation has been widely used in studies related to fluid dynamics and other physical processes. No personally identifiable information or sensitive data was included in the dataset, and ethical considerations related to data privacy are not applicable.

4.2 Reproducibility

The code for training the models and analyzing the results has been uploaded to https://github.com/hope-I-can-change-it/research-project. The repository contains a modified version of the code from [3], with adaptations to suit the specific needs of this study. Clear documentation has been provided to ensure that others can easily reproduce the experiments, including setup instructions, configuration

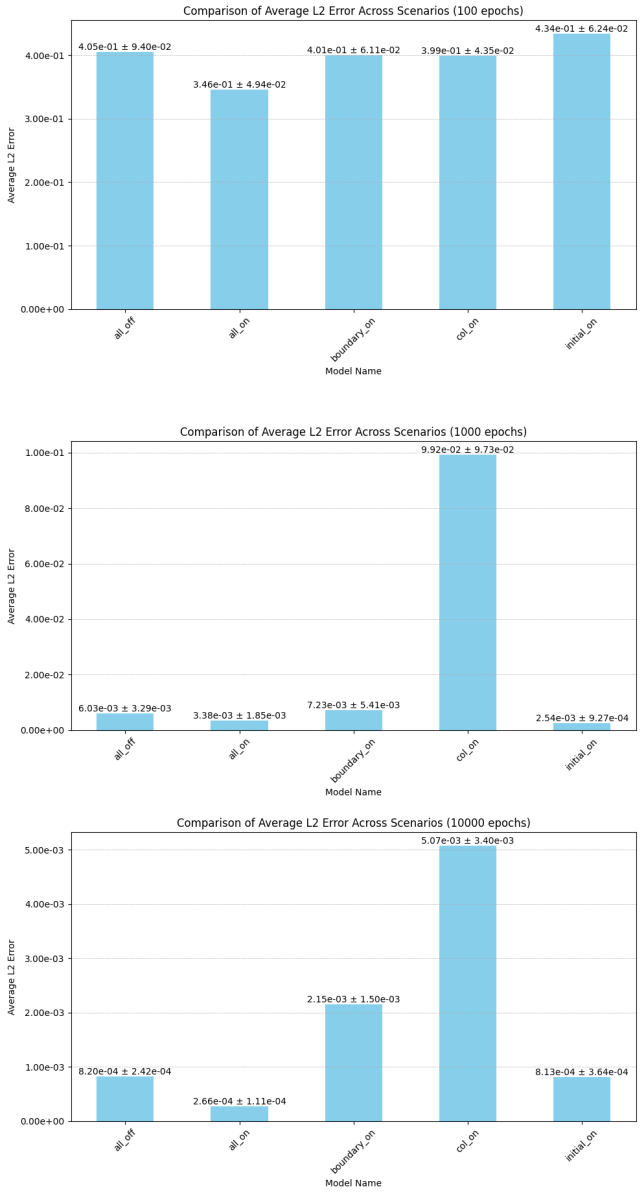

Figure 1: Average L2-error when training for (top) 100, (middle) 1000, and (bottom) 10000 epochs.

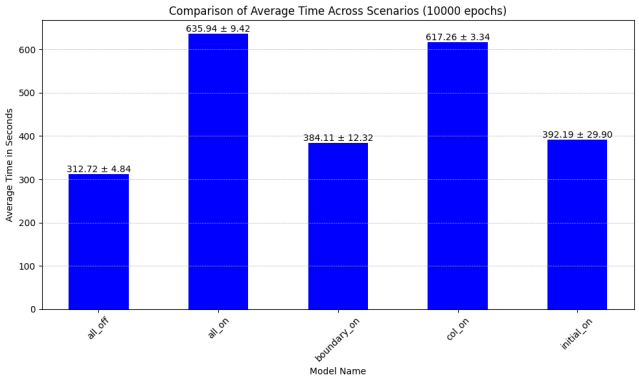

Figure 2: Average training time when training for 10000 epochs

details, and descriptions of the model parameters used.

However, challenges to reproducibility exist. The dataset used in this study is based on the Viscous Burgers equation and it is the same dataset used by [1], and while the code for data preprocessing is available, the specific configurations of the equation used for the experiments may not be easily replicated without the exact setup provided in the repository.

Additionally, some parts of the experiments rely on computational resources that may not be available to all researchers. For example, the models were trained on Intel i5-12600k processor, which may not be accessible to everyone.

4.3 Conclusion

Efforts were made to ensure that the research is both responsible and reproducible. The code and methodology are publicly available, and others are encouraged to replicate and build upon this work.

5 Discussion

The results obtained from the experiments provide valuable insights into the effectiveness of adaptive loss weighting mechanisms in Physics-Informed Neural Networks (PINNs). This section discusses key findings in comparison with existing literature and explores the trade-offs between accuracy and computational cost.

5.1 Comparison with Existing Studies

The results from this study align with what has been found by [3], who used a similar SA-PINN setup. Their recorded L2-error after 10000 epochs of training was 4.08e-04$\pm$1.01e-04., which is slightly higher than the 2.66e-04$\pm$1.11e-04 obtained in this study. A possible reason for this is that their implementation did not include the boundary condition mask, utilizing only the PDE residual and initial conditions masks. They also reported achieving the same accuracy with their setup in one fifth of the epochs compared to the baseline PINN. While this study did not track the exact number of epochs it takes to reach a certain accuracy, it is important to note that although SA-PINNs may require fewer epochs to converge, the training time is significantly longer. Figure 2 shows that the training time for the full SA-PINN configuration with all adaptive masks enabled is more than twice that of the baseline PINN. This highlights the trade-off between improved performance and increased computational cost.

5.2 Performance Trade-offs

As mentioned in the section before, a notable observation is the trade-off between accuracy and computational cost. While the full SA-PINN configuration resulted in the lowest L2-error across all training durations, it required more than twice the training time compared to the baseline. This finding suggests that, for real-world applications with limited computational resources, using selective adaptive masks, might be a more viable option.

The initial conditions mask consistently improved model accuracy with a moderate increase in training time. This supports the hypothesis that accurate initial condition enforcement plays a critical role in PINN convergence and stability, making it a promising choice for improving performance without increasing the training time as much as full SA-PINN implementation.

5.3 Contribution of Adaptive Components

The analysis of individual adaptive masks showed how each mask contributes to the overall accuracy. The initial conditions mask provided the most consistent improvements, while the boundary conditions mask had a very small positive effect in shorter runs, and degraded performance over long ones. The PDE residual mask, when used alone, resulted in significant performance degradation over prolonged training, suggesting potential issues with overemphasizing residuals without balancing other loss components. However, it is worth noting that the PDE residual mask is crucial for the full SA-PINN implementation. Although the boundary and initial condition masks yield better results individually, they do not surpass the performance of the full SA-PINN configuration. Moreover, when an additional scenario was run, where initial and boundary condition masks were turned on, and the PDE residual was turned off, over 5 runs, it resulted in an average error of 1.11e-03 $\pm$ 8.18e-04. This error is still higher than the full SA-PINN implementation with all 3 masks turned on. This indicates that even though the PDE residual mask may substantially decrease performance when used alone, it is crucial for the full SA-PINN implementation.

The interdependence of different adaptive masks can also be seen when looking at the weight distributions after the training on Figure 3. The top image illustrates the scenario where only the PDE residual loss mask was enabled, while the bottom image represents the full SA-PINN implementation with all masks turned on. A comparison of these two figures reveals that the full SA-PINN implementation produces a more structured and meaningful weight distribution, closely resembling the original function depicted in Figure 4. Notably, larger weights are concentrated in regions of deep blue and yellow areas of the solution, indicating that

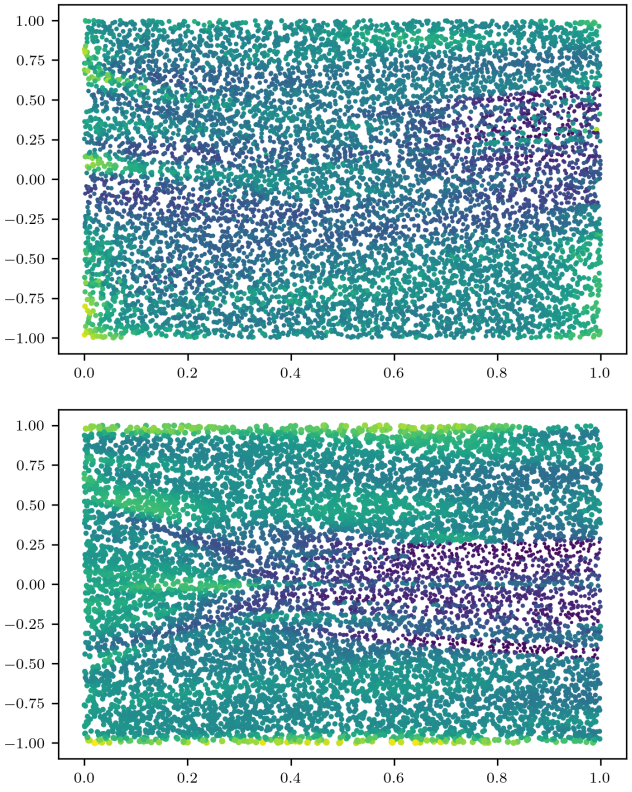

Figure 3: Adaptive weights after training with only PDE residual mask on (top) and full SA-PINN implementation (bottom) for 10000 epochs. Brighter colored points correspond to larger weights.

these are the areas where the model struggled to achieve accurate predictions. A visible line can also be observed at $x = 0$, that can also be seen on the reference solution graph. It is also worth noting that the full implementation graph was also included in [3] and it is a bit different than the one obtained by this study. In their version, the line at $x = 0$ is more visible, and the weights are not as high in the yellow and blue regions.

6 Conclusions and Future Work

This study has explored the impact of adaptive weighting in Self-Adaptive Physics-Informed Neural Networks (SA-PINNs) for solving stiff partial differential equations (PDEs), with a focus on the individual roles of residual, boundary, and initial condition components and adaptive masks applied to them. The results align with the findings of [3], confirming that SA-PINNs benefit from the self-adaptive weights and demonstrate increased performance when tested on challenging problems like the Viscous Burgers Equation. The experiments conducted in this study showed that enabling adaptive weights can have different results depending on the component they are enabled for. Enabling adaptive weights for the boundary condition component improved accuracy during the early stages of training (100 epochs) but led to a decline in later stages (1000, 10000 epochs). Enabling adaptive weights for the initial condition component resulted

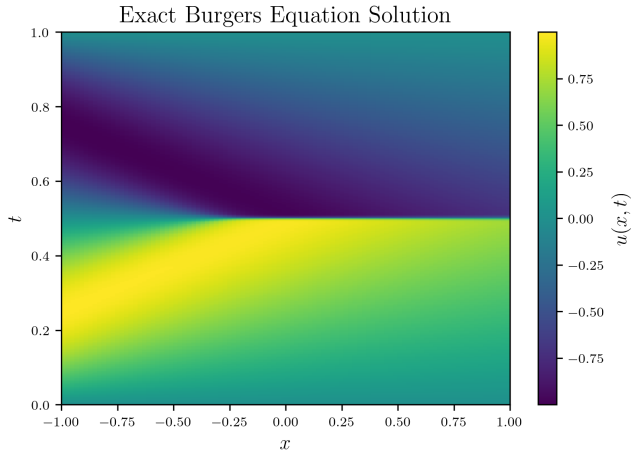

Figure 4: High-fidelity solution for Viscous Burgers equation.

in slightly worse performance in the early stages of training, but better accuracy in the later stages. Enabling adaptive weights for the PDE residual component led to worse model performance across all tested scenarios. However, the full SA-PINN implementation, with all adaptive masks enabled, outperformed the traditional PINN in terms of accuracy across all training epochs.

It has been found that the adaptive masks have a major impact on the time it takes the model to train. Full SA-PINN implementation training takes two times as much time as the baseline PINN. While enabling singular masks does not have such a big impact, the difference is still noticeable and should be considered.

Future research could explore whether these results are replicable for other PDEs, such as the Allen-Cahn or Helmholtz equations, to assess the generalizability of the findings. It may also provide further insights into how these masks interact and contribute to optimizing model training times. This study shows the importance of understanding how each term in the loss function interacts with the adaptive weighting mechanism, and how the masks interact with each other.

References

[1] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[2] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[3] Levi D. McClenny and Ulisses M. Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of*

Computational Physics, 474:111722, February 2023.

[4] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks, 2020.

[5] Colby L. Wight and Jia Zhao. Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks, 2020.

[6] Yi Ding, Su Chen, Xiaojun Li, Suyang Wang, Shaokai Luan, and Hao Sun. Self-adaptive physics-driven deep learning for seismic wave modeling in complex topography. *Engineering Applications of Artificial Intelligence*, 123:106425, 2023.

[7] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2022.

[8] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[10] Wei-Liem Loh. On latin hypercube sampling. *The annals of statistics*, 24(5):2058–2080, 1996.