Multi-Objective PSO-PINN

Caio Davi [1] **Ulisses Braga-Neto** [1]

Abstract

PSO-PINN is a class of algorithms for training physics-informed neural networks (PINN) using particle swarm optimization (PSO). PSO-PINN can mitigate the well-known difficulties presented by gradient descent training of PINNs when dealing with PDEs with irregular solutions. Additionally, PSO-PINN is an ensemble approach to PINN that yields reproducible predictions with quantified uncertainty. In this paper, we introduce Multi-Objective PSO-PINN, which treats PINN training as a multi-objective problem. The proposed multi-objective PSO-PINN represents a new paradigm in PINN training, which thus far has relied on scalarizations of the multi-objective loss function. A full multi-objective approach allows on-the-fly compromises in the trade-off among the various components of the PINN loss function. Experimental results with a diffusion PDE problem demonstrate the promise of this methodology.

1. Introduction

Scientific Machine Learning (SciML) (Baker et al., 2019) has attracted tremendous interest over the last few years. The most well-known SciML algorithm is the physics-informed neural network (PINN) (Raissi et al., 2019; Cai et al., 2022), which constrains a neural network to satisfy a differential equation through an extra loss term. Although the idea of training neural networks to satisfy differential equations had been already proposed in the 1990's (Dissanayake & Phan-Thien, 1994), recent advancements in high-performance computational infrastructure (Abadi et al., 2016) and automatic differentiation algorithms (Revels et al., 2016) have allowed deep PINNs to be employed in realistic engineering and scientific problems, which has attracted great interest to this class of machine learning algorithm. Unlike traditional numerical methods, PINN allows data to be assimilated in the model in a natural way, and unlike data-driven machine learning techniques, PINN introduces a strong regularizing prior through the use of differential equations that reflect physical understanding of the problem.

Nevertheless, PINN presents a difficult multi-objective problem through its complex loss function with multiple terms. The solution that has been universally adopted at this point has been scalarization into a single-objective loss function by adding the different terms. In the original PINN algorithm (Raissi et al., 2019), the loss terms are simply added. This has been known to generate imbalance among the various loss terms in problems with complex solutions that display sharp space-temporal transitions (Wang et al., 2021; 2022; Liao et al., 2022). A more sophisticated scalarization adds multiplicative weights to the loss terms, which can be fixed, e.g. based on prior knowledge (Wight & Zhao, 2020; Jin et al., 2021), or using adaptive weights that change during training (Wang et al., 2022). A more complex adaptive approach weights each training point separately and trains these weights together with the neural network weights (McClenny & Braga-Neto, 2020). These weight-based scalarization approaches improve the performance of PINNs, in some cases dramatically, but they still fail in sufficiently complex problems.

In this paper, we explore a different perspective to PINN training, which treats it as a full multi-objective optimization problem rather than relying on linear scalarization of the loss terms. We do this by using particle swarm optimization (PSO) (Kennedy & Eberhart, 1995; Shami et al., 2022), due to the availability of efficient multi-objective PSO algorithms (Marler & Arora, 2004; Emmerich & Deutz, 2018). This extends the previous PSO-PINN algorithm (Davi & Braga-Neto, 2022), which relied on the traditional single-objective scalarized loss function. This allows us to propose a new approach for determining the extent of the influence of each loss component, e.g. the influence of the prior physics information as opposed to the sample data, which does not rely on scalarization. Instead, the practitioner is able to train the model simultaneously over the many possible objectives, and select the best one suited for the problem by means of a post-training analysis.

[1]Department of Electrical and Computer Engineering, Texas A&M University , College Station, USA. Correspondence to: Ulisses Braga-Neto <ulisses@tamu.edu>.

Accepted after peer-review at the 1st workshop on Synergy of Scientific and Machine Learning Modeling, SynS & ML ICML, Honolulu, Hawaii, USA. July, 2023. Copyright 2023 by the author(s).

2. Background

2.1. Deep Neural Networks

Given enough hidden neurons and sufficient training, multi-layer feed forward networks are universal approximators(Hornik et al., 1989). This outstanding property lets this algorithm family evolve in today's deep learning algorithms(LeCun et al., 2015). Usually, the topology consists of a feed-forward fully-connected network, the basic architecture for deep learning. A fully-connected neural network with L layers is a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$ described in Equation 1. This function is composed essentially of three components, the weights, bias, and activation functions. Among popular choices for the activation function are the sigmoid function, the hyperbolic tangent function (tanh), and the rectified linear unit (ReLU) (Glorot et al., 2011).

$$f_\theta(x) \;=\; W^{[L-1]}\sigma\circ(\cdots\sigma\circ(W^{[0]}x+b^{[0]})+\cdots)+b^{[L-1]} \tag{1}$$

where σ is an entry-wise activation function, $W^{[l]}$ and $b^{[l]}$ are respectively the weight matrices and the bias corresponding to each layer l, and θ is the set of weights and biases:

$$\theta = (W^{[0]}, \cdots, W^{[L-1]}, b^{[0]}, \cdots, b^{[L-1]})\,. \tag{2}$$

This framework allows the approximation of $f_\theta(\cdot)$ to any arbitrary function $g : \mathbb{R}^d \to \mathbb{R}^k$. The method for building this approximation is called training and it is typically guided by a loss function $\mathcal{L} : \Theta \times \mathcal{T} \to \mathbb{R}$. Among others, a popular choice in deep learning is the Mean Square Error(MSE):

$$\mathcal{L}(\theta, x) \;=\; |f_\theta(x) - y|^2, \tag{3}$$

where $y \in \mathbb{R}^k$ is the target value. Several other loss functions have been proposed in the last decades, where a particular choice may drastically change the behavior of the neural network model (Janocha & Czarnecki, 2017).

Calculating $f(x)$ and $\mathcal{L}(\theta, x)$ is commonly called forward propagation. It will provide enough information for the subsequent phase, the backpropagation. During the backpropagation, the network's weights and bias are updated based on the gradient of the error given by the loss function:

$$\theta^{t+1} = \theta^t - \alpha\frac{\partial\mathcal{L}(\theta^t, x)}{\partial\theta^t} \tag{4}$$

where θ^t denotes the learnable parameters of the neural network at iteration t in gradient descent and α is the learning rate. Again, there were numerous gradient-based optimization methods proposed in the last years (Duchi et al., 2011; Kingma & Ba, 2014; Ruder, 2016).

2.2. Physics-Informed Neural Networks

Consider a non-linear partial differential equation (PDE)

$$\begin{aligned} \mathcal{N}[u(\mathbf{x},t)] &= f(\mathbf{x},t),\ \mathbf{x}\in\Omega,\ t\in[0,T] \\ u(\mathbf{x},t) &= g(\mathbf{x},t),\ \mathbf{x}\in\partial\Omega,\ t\in[0,T] \\ u(\mathbf{x},0) &= h(\mathbf{x}),\ \mathbf{x}\in\Omega \end{aligned} \tag{5}$$

where $\Omega \subset \mathbb{R}^n$, $u : \Omega \to \mathbb{R}^k$, $\mathcal{N}[\cdot]$ is a spatio-temporal differential operator, and the source, boundary condition, and initial conditions are provided by the functions $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$, respectively. The PINN approach consists in training a deep neural network $u_\theta(\mathbf{x}, \mathbf{t})$ (where θ contains the network weights and any unknown parameters of $\mathcal{N}$) to approximate the solution $u(\mathbf{x}, \mathbf{t})$ of the PDE. This task can be accomplished by minimizing the losses:

$$\mathcal{L}_r(\theta) \;=\; \frac{1}{N_r}\sum_{n=1}^{N_r}|\mathcal{N}[u_\theta(x_r^n, t_r^n)] - f(x_r^n, t_r^n)|^2 \tag{6}$$

$$\mathcal{L}_b(\theta) \;=\; \frac{1}{N_b}\sum_{n=1}^{N_b}|u_\theta(x_b^n, t_b^n)] - g(x_b^n, t_b^n)|^2 \tag{7}$$

$$\mathcal{L}_0(\theta) \;=\; \frac{1}{N_0}\sum_{n=1}^{N_0}|u_\theta(x_0^n, 0)] - h(x_0^n)|^2 \tag{8}$$

$$\mathcal{L}_d(\theta) \;=\; \frac{1}{N_d}\sum_{n=1}^{N_d}|u_\theta(x_d^n, t_d^n)] - y^n)|^2 \tag{9}$$

where $\{x_r^n, t_r^n\}_{n=1}^{N_r}$ are collocation points in $\Omega \times [0,T]$, $\{x_b^n, t_b^n\}_{n=1}^{N_b}$ are boundary condition points in $\partial\Omega \times [0,T]$, $\{x_0^n, 0\}_{n=1}^{N_0}$ are initial conditional points in $\Omega \times \{t = 0\}$, and $\{[x_d^n, t_d^n], y^n\}_{n=1}^{N_d}$ are experimental data points, if any. The collocation, boundary, and initial points are typically sampled randomly in their respective domains.

The parameter vector θ must minimize all these loss functions; this is clearly a multi-objective optimization problem, in which the various losses "compete" against each other. However, in the literature of PINN, single-objective optimization is employed, by linearly scalarizing the multiple losses into a single loss function:

$$\mathcal{L}(\theta) \;=\; \mathcal{L}_r(\theta) + \mathcal{L}_b(\theta) + \mathcal{L}_0(\theta) + \mathcal{L}_d(\theta)\,. \tag{10}$$

2.3. Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm (Eberhart & Kennedy, 1995; Shi & Eberhart, 1999) is a population-based stochastic optimization algorithm that emulates the swarm behavior of particles distributed in a n-dimensional search space (Wang et al., 2018). Each individual in this swarm represents a candidate solution. At each iteration, the particles in the swarm exchange information

and use it to update their positions. Particle $\theta(t)$ at iteration t is guided by a velocity determined by three factors: its own velocity inertia $\beta V(t)$, its best-known position p_{best} in the search-space, as well as the entire swarm's best-known position g_{best}:

$$V(t+1) \;=\; \beta V(t) + c_1 r_1 (p_{best} - \theta(t)) + c_2 r_2 (g_{best} - \theta(t))\,, \tag{11}$$

where c_1 and c_2 are the cognitive and social coefficients, respectively, and r_1 and r_2 are uniformly distributed random numbers in range $[0, 1)$. Then the particle position is updated as

$$\theta(t+1) \;=\; \theta(t) + V(t+1)\,. \tag{12}$$

Many variations of the PSO algorithm have been proposed, including a hybrid PSO and gradient-based algorithm known as PSO-BP(Yadav et al., 2019), which adds a gradient descent component to the swarm framework. This strategy came particularly handy when applying the method to train neural networks, since all partial gradients can be computed efficiently by backpropagation. In the PSO-BP algorithm, the particle's velocity is updated via (contrast this to equation 11):

$$\begin{aligned} V(t+1) \;=\; & \beta V(t) + c_1 r_1 (p_{best} - \theta(t)) \\ & + c_2 r_2 (g_{best} - \theta(t)) - \alpha \nabla \mathcal{L}(\theta(t))\,, \end{aligned} \tag{13}$$

where α is the learning rate and $\nabla \mathcal{L}(\theta(t))$ is the loss gradient. Therefore, the gradient participates in the velocity magnitude and direction, by an amount that is specified by the learning rate.

3. Multi-Objective PSO-PINN

For notational convenience, let us denote the various loss components by $\mathcal{L}_i(\theta)$, $i = 1, \ldots, M$. The goal of the original PINN algorithm is to find θ^* that minimizes the sum $\sum_{i=1}^{M} \mathcal{L}_i(\theta)$. In the multi-objective PINN approach, we contemplate the problem of finding θ^* that minimizes all losses simultaneously. In most cases, this problem does not have a solution. However, one can build a set of "admissible" solutions, which is called the *Pareto Front*. The following two definitions are used to build the Pareto Front:

Definition 1 (Pareto dominance). *A solution θ^* dominates another solution θ (denoted as $\theta^* \prec \theta$) if and only if:*

- *θ^* is not worse than θ in any objective, i.e.: $\mathcal{L}_i(\theta^*) \leq \mathcal{L}_i(\theta)$, for all $i = 1, \ldots, M$.*
- *θ^* is better than θ in at least one objective, i.e.: $\mathcal{L}_i(\theta^*) < \mathcal{L}_i(\theta)$, for some $i = 1, \ldots, M$.*

Definition 2 (Pareto optimality). *A solution θ^* is Pareto optimal if it is not dominated by any other solution. Therefore, the set of all Pareto optimal solutions is $\mathcal{P} := \{\theta \in \Theta \,|\, \nexists \theta' \in \Theta : \theta' \prec \theta\}$. Meanwhile, the Pareto front $\mathcal{F}$ is the m-dimensional manifold of the objective values of all Pareto optimal solutions $\mathcal{F} := \{(\mathcal{L}_1(\theta), \ldots, \mathcal{L}_M(\theta)) \,|\, \theta \in \mathcal{P}\}$.*

The goals in a MOO problem are often conflicting. Thus the Pareto front may work as a tool to select the best model according to the objectives of main interest. The set of Pareto non-dominated models comprises solutions that cannot improve any objective without degrading at least one of the remaining objectives. That particular setup comes in handy when it is needed to compromise to prioritize between solutions.

Algorithm 1 Multi-Objective PSO-PINN

Require: α: step size;
Require: β: inertia;
Require: c_1, c_2: behavioral coefficients;
Require: $\mathcal{L}[\cdot]$: vector of loss functions;
Initialize population Θ;
$\Omega \leftarrow []$;
$p_{best}(i) \leftarrow \theta_i$, for $i = 1, \ldots, |\Theta|$;
$g_{best} \leftarrow$ rand(p_{best});
for $t = 1, 2, \ldots,$ MAX **do**:
  for $i = 1, \ldots, |\Theta|$ **do**:
    $r_1, r_2 \leftarrow U(0, 1]$;
    $V = \beta V + c_1 r_1 (p_{best}(i) - \theta_i) + c_2 r_2 (g_{best} - \theta_i)$;
    $\theta_i = \theta_i + V$;
    if $\mathcal{L}_j(\theta_i) \leq \mathcal{L}_j(p_{best}(i)) \quad \forall \mathcal{L}_j \in |\mathcal{L}|$:
      $p_{best}(i) \leftarrow \theta_i$;
    end
    if $p_{best}(i)$ dominates Ψ:
      $\Psi^\frown \langle \theta_i \rangle$;
    end
  end
  $g_{best} \leftarrow$ rand(Ψ);
  if *coefficient_decay*:
    $c_1 = c_1 - \frac{2c_1}{t}$;
    $c_2 = c_2 - \frac{c_2}{t}$;
end
return Ω

Here we propose a novel paradigm for PINN training using MOO. Instead of making a pre- or during-training weighting of the various loss terms, we introduce a post-training selection strategy. By leveraging the Pareto front generated by MOO algorithms, solutions can be analyzed based on their positions on the Pareto front. Our investigation demonstrates the particular suitability of MOO for the PSO-PINN algorithm due to its innate decentralized behavior, facilitating a search in a multiple objective dimension, and its built-in population tracking capabilities.

Instead of using the equation 10 to guide the optimization of the θ parameters of the PINN, one can jointly minimize the

set of $\mathcal{L}$ loss functions, following the Equation 14, where each objective focuses on minimizing the expectation for all the respective loss functions:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{(\mathbf{x},\mathbf{t})}\{\mathcal{L}_1(\theta, \mathbf{x}_1, \mathbf{t}_1), \mathcal{L}_2(\theta, \mathbf{x}_2, \mathbf{t}_2), \ldots \quad , \mathcal{L}_n(\theta, \mathbf{x}_n, \mathbf{t}_n)\} \tag{14}$$

The multi-objective approach proceeds as follows. An archive Ψ holds the Pareto front during the training. It starts as an empty vector and expands as we find new optimal points. Unlike in the original PSO-PINN algorithm, the particle update does not employ a gradient-based component. Since we now have a vector of losses, this would incur multiple gradient evaluations and a costly update. This losses vector also changes the p_{best} updates, which now require equal or smaller values for all the losses of the new position. After defining the p_{best}, we check if it dominates the archive Ψ (following Definitions 1 and 2). If that is the case, we should append it to Ψ. Lastly, the new g_{best} should be selected among the positions in Ψ, since all positions are theoretically equally suitable solutions. The Algorithm 1 describes in detail the multi-objective PSO-PINN algorithm.

4. Experiments

We demonstrate the performance of MOO-PSO-PINN on the diffusion equation:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \qquad x \in [-1, 1], \quad t \in [0, 1]\,, \tag{15}$$

where $u(x, t)$ is a function of the position x over the time t, and α is the parametrization coefficient, commonly known as the thermal diffusivity constant. The constraints for this equation, the initial (IC) and boundary (BC) conditions are given by:

$$\begin{aligned} &u(0, t) = u(1, t) = 0, \\ &u(x, 0) = \sin\left(\frac{n\pi x}{L}\right), \qquad 0 < x < L\,, \end{aligned} \tag{16}$$

where $L = 1$ is the length of the bar, $n = 1$ is the frequency of the sinusoidal initial conditions. As defined here, this problem is well-posed and has a unique solution:

$$u(x, t) = e^{\frac{-n^2\pi^2\alpha t}{L^2}} \sin(\frac{n\pi x}{L})\,. \tag{17}$$

As said before, the simple accommodation of the losses to the MOO framework (as defined in 14) would not suffice. Disconnected to any constraints, the equation 15 has infinite solutions; in fact, any constant would be a solution.

Therefore, the multi-objective for this experiment is defined by the loss vector (c.f. equations 6 to 9):

$$\begin{aligned} \mathcal{L}_\theta = \{&[\mathcal{L}_r(\theta, \mathbf{x_r}, \mathbf{t_r}) + \mathcal{L}_0(\theta, \mathbf{x_0})], \\ &[\mathcal{L}_b(\theta, \mathbf{x_b}, \mathbf{t_b}) + \mathcal{L}_d(\theta, \mathbf{x_d}, \mathbf{t_d})]\}\,. \end{aligned} \tag{18}$$

Notice that we may use as many objectives as we wanted for this problem, but for the sake of simplicity, we kept the objectives in a two-dimensional vector. The first one contains the sum of the residual and original loss. The second is the sum of the boundaries and data losses.

For this experiment, we set the thermal diffusivity constant $\alpha = 0.4$ as the reference solution. Also, we added uniformly distributed noise in the range $[0.0, 0.3)$ to all data in the problem, including initial points, boundary points, and data points. The neural-net architecture was composed of two input neurons (for x and t), followed by six hidden layers with eight neurons each, and a single output representing $u(x, t)$. The PSO parameters β, c_1, and c_2 were set to 0.99, 0.008, and 0.05, respectively.

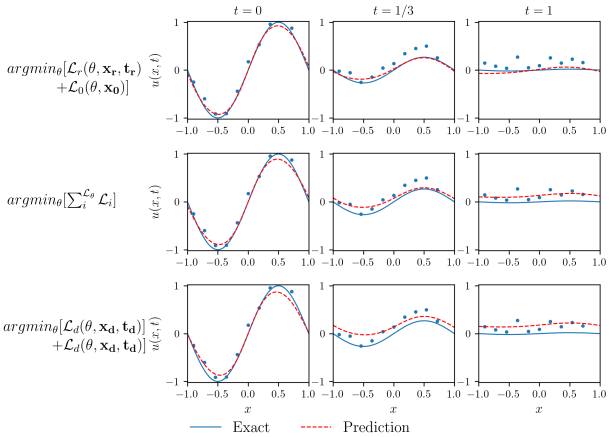

Figure 1. Multi-objective PSO-PINN solutions for the Heat Equation – Each row represents one of the solutions found in the Pareto Front (Figure 2). On top, the solution minimizes the first loss from the Loss vector. In the middle, we have the solution most close to zero, one that minimizes the sum of all in (18). The last row is the best solution for the boundary and data losses.

Figure 1 displays different solutions through the Pareto front. The first row (the solution which minimizes the original and residual losses) fits the proposed PDE well. It does not completely fit the reference solution, mostly due to the heavy noise in all data sources. Even though it is the optimal solution for the PINN loss, it suffers influence from the other losses in the loss vector (18). In the third row, we can see that, although the solution fitted the data points, it was compromised by the noisy data points. Lastly, the second row depicts the solution with the minimum sum of

all objective losses. This solution is at the elbow point for the Pareto Front, as shown in Figure 2. We cannot say that it is the best solution, since all solutions in the Pareto Front are optimal, but we might say that this particular solution at the elbow point holds the best balance among the objectives.

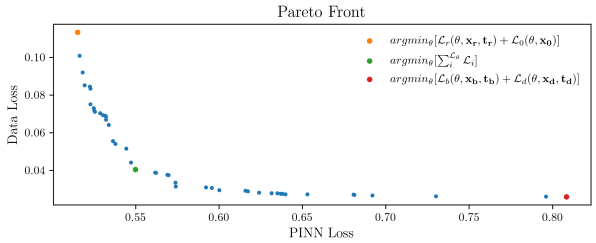

Figure 2. The Pareto front of multi-objective PSO-PINN solutions for the Heat Equation.

5. Conclusion

This paper introduced new training strategies for PINN ensemble training. In particular, we extended the PSO-PINN algorithm, taking advantage of the swarm properties for multi-objective optimization inherited from the PSO. The multi-objective PSO-PINN intends to assist the training in setting the influence from the analytical model and training data. The multi-objective PSO-PINN can be used to appraise the problem and define how the prior information and the available data would best benefit the training.

Much is said about the elasticity of PINNs, which can be capable of bouncing between a strict mathematical PDE solver to a more yielding data-driven approach. Until now, this property was partially explored, most often relying on the weighting of the total loss function. The approach described here yields a set of Pareto non-dominated solutions, where the best can be chosen based on how well-defined the terms are. For example, if the dataset is considerably noisy, one may choose a model respecting the physical constraints. On the other hand, if, for some reason, the environment is not well defined on the PDE, or the PDE parameters are loosely fitted, one can choose a more data-driven solution.

Broader impact

The proposed algorithm for training PINNs holds the potential for broader impacts by improving the efficiency and effectiveness of the training process. Since the algorithm currently does not involve the use of real-world datasets and does not directly address ethical aspects, its implications in terms of societal, economic, or ethical considerations are limited. Nevertheless, the advancements in training methodologies for PINNs can have significant scientific and practical implications. The algorithm opens avenues for more accurate and reliable predictions in various domains, such as physics-based simulations, engineering design optimization, and scientific research. The enhanced capabilities of PINNs can contribute to advancements in fields like fluid dynamics, material science, petroleum engineering and structural engineering, enabling more precise modeling, optimization, and decision-making processes. Furthermore, the innovation in training techniques may inspire further research in developing improved machine learning algorithms and frameworks, influencing the broader landscape of machine learning and its potential applications. It is crucial for future studies to consider the societal and ethical implications that may emerge as these advanced techniques are deployed in real-world scenarios.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Baker, N., Alexander, F., Bremer, T., Hagberg, A., Kevrekidis, Y., Najm, H., Parashar, M., Patra, A., Sethian, J., Wild, S., et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, pp. 1–12, 2022.

Davi, C. and Braga-Neto, U. Pso-pinn: Physics-informed neural networks trained with particle swarm optimization. *arXiv preprint arXiv:2202.01943*, 2022.

Dissanayake, M. and Phan-Thien, N. Neural-network-based approximations for solving partial differential equations. *communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Eberhart, R. and Kennedy, J. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pp. 1942–1948. Citeseer, 1995.

Emmerich, M. and Deutz, A. H. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing*, 17(3):585–609, 2018.

Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323. JMLR Workshop and Conference Proceedings, 2011.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Janocha, K. and Czarnecki, W. M. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.

Jin, X., Cai, S., Li, H., and Karniadakis, G. E. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

Kennedy, J. and Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pp. 1942–1948. IEEE, 1995.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Liao, S., Xue, T., Jeong, J., Webster, S., Ehmann, K., and Cao, J. Hybrid full-field thermal characterization of additive manufacturing processes using physics-informed neural networks with data. *arXiv preprint arXiv:2206.07756*, 2022.

Marler, R. T. and Arora, J. S. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

McClenny, L. and Braga-Neto, U. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Revels, J., Lubin, M., and Papamarkou, T. Forward-mode automatic differentiation in julia. *arXiv preprint arXiv:1607.07892*, 2016.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., and Mirjalili, S. Particle swarm optimization: A comprehensive survey. *IEEE Access*, 2022.

Shi, Y. and Eberhart, R. C. Empirical study of particle swarm optimization. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pp. 1945–1950. IEEE, 1999.

Wang, D., Tan, D., and Liu, L. Particle swarm optimization algorithm: an overview. *Soft Computing*, 22(2):387–408, 2018.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

Wang, S., Yu, X., and Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Wight, C. L. and Zhao, J. Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*, 2020.

Yadav, R. K. et al. Ga and pso hybrid algorithm for ann training with application in medical diagnosis. In *2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS)*, pp. 1–5. IEEE, 2019.