Scientific Machine Learning (SciML) – How the fusion of AI and physics is giving rise to promising simulation methodologies

Andreas Rosskopf
Department Modeling and AI
Fraunhofer IISB
Erlangen, Germany
andreas.rosskopf@iisb.fraunhofer.de

Xuepeng Cheng
Department of Data Science
Friedrich-Alexander-Universität Erlangen-Nürnberg
Erlangen, Germany
xuepeng.cheng@foxmail.com

Christopher Straub
Department Modeling and AI
Fraunhofer IISB
Erlangen, Germany
christopher.straub@iisb.fraunhofer.de

Daniel Tenbrinck
Department of Data Science
Friedrich-Alexander-Universität Erlangen-Nürnberg
Erlangen, Germany
daniel.tenbrinck@fau.de

***Abstract*—The combination of physical modeling and machine learning, known as Scientific Machine Learning (SciML), is enabling a new generation of simulation methodologies. In this work, we demonstrate the potential of SciML for simulating silicide formation in Ni-SiC systems, a process highly relevant to TCAD and device engineering. Four neural network architectures – standard MLPs, enhanced mMLPs with residual connections, interpretable Kolmogorov-Arnold Networks (KANs), and Chebyshev KANs (cKANs) – are benchmarked, each trained solely on the governing physical laws. All models tested achieve accurate results and enable rapid evaluation, providing large speedups over traditional solvers. Among these, the mMLP yields the best accuracy. These findings underscore the strong potential of SciML for efficient and accurate TCAD simulations, paving the way for scalable, data-integrated modeling of complex material interactions in semiconductor technology.**

***Index Terms*—Physics-informed neural networks, numerical simulation, diffusion-reaction equations, scientific machine learning, silicidation**

I. Introduction

Understanding and optimizing the interaction between metals and semiconductors is fundamental for the design and fabrication of current and future semiconductor devices [1]. Technology Computer-Aided Design (TCAD) simulation plays a critical role in advancing this understanding, enabling detailed modeling and predictive analysis of materials and processes. TCAD simulations are indispensable for tackling challenges in the development of high-performance devices, including power transistors, diodes [2], and sensors [3]. This holds in particular true for applications requiring high efficiency, reliability, and operation under extreme conditions, e.g., high temperatures or high voltages [1].

The fundamental mathematical modeling and numerical simulation of these processes typically rely on diffusion-reaction equations, multi-phase systems modeling, and kinetic rate equations. These underlying equation systems are traditionally solved using finite difference, finite elements, or implicit numerical schemes. While these approaches provide a robust framework for simulating material interaction dynamics, they face significant challenges considering the computational complexity because solving multi-dimensional, non-linear partial differential equations (PDEs) for multi-component systems is computationally expensive. Furthermore, they are limited by simplified physical assumptions, e.g., constant diffusion coefficients or isotropic reactions that fail to capture the full complexity of real-world material interactions. Finally, realizing scalability of such simulations is difficult as traditional methods struggle to efficiently scale when handling multi-physics phenomena or coupling with external effects.

To address these challenges, this article proposes the combination of AI-driven methods with traditional physical simulation methods for simulating interactions between metals and semiconductors. This approach has become known as Scientific Machine Learning (SciML) in recent years and offers the potential to overcome computational inefficiencies, improve the accuracy of predictions, and integrate measured data into simulation models to create more realistic surrogate models for processes and devices. To demonstrate the potential of SciML for interaction simulation, we focus on the interaction between Nickel (Ni) and Silicon Carbide (SiC) in the context of creating ohmic contacts for wide-bandgap semiconductor devices. The Ni-SiC system is a TCAD-relevant use case that embodies the challenges of modeling metal-semiconductor interactions, including multi-phase silicide formation, carbon redistribution, and defect dynamics during high-temperature annealing. As the foundation of our analysis, we leverage a quantitative

This work was supported by the Fraunhofer Internal Programs under Grant No. PREPARE 40-08394.

model from Aleksandrov and Kozlovski [5], which describes the mutual diffusion of components and silicide-formation reactions in the Ni-SiC system. We quantitatively evaluate the impact of the chosen neural network (NN) architecture of four SciML approaches on the solution accuracy and simulation runtime of the investigated models.

II. Diffusion-reaction model for Ni-SiC systems

Many TCAD solutions, like Synopsys Sentaurus [4], use mathematical models to describe the interaction between metals and semiconductors. In the following, we summarize the mathematical modeling of Ni-SiC systems described in [5] which is representative for many related models in this field. This approach falls into the category of diffusion-reaction models with boundary and phase-change dynamics, which in particular includes mathematical terms for

- **Diffusion Equations:** Governed by non-linear PDEs including multi-component diffusion terms.
- **Kinetics:** Reaction terms with different rate constants for silicide formation, increasing dimensionality.
- **Kirkendall Effect:** Introduces asymmetry in diffusion coefficients, requiring advanced numerical schemes.

Concretely, the model from [5] describes the silicidation process via the concentration functions for Ni (C_{Ni}), SiC (C_{SiC}), carbon (C_{C}), and the relevant Ni silicides (C_{NiSi}, $C_{\mathrm{Ni_2Si}}$). It is assumed that all concentrations are homogeneously distributed in all directions except for the depth, hence reducing the number of effective space dimensions to 1. The model covers the two reactions $\mathrm{Ni} + \mathrm{SiC} \rightarrow \mathrm{NiSi} + \mathrm{C}$ and $\mathrm{Ni} + \mathrm{NiSi} \rightarrow \mathrm{Ni_2Si}$ with reaction rates k_1 and k_2, respectively. The values of these constants depend on the annealing temperature during the process. Similar to [5, Figs. 2 and 3], we consider an annealing temperature $T < 900^\circ\,\mathrm{C}$ here. By suitably adapting the model's parameters and including other reactions, it is straight-forward to extend the model to higher temperatures. The diffusion-reaction equations for the concentrations of Ni, SiC, and C are of the form

$$\frac{\partial C_{\mathrm{Ni}}}{\partial t} = \nabla \cdot (D^* \,\nabla C_{\mathrm{Ni}}) - k_1 C_{\mathrm{Ni}} C_{\mathrm{SiC}} - k_2 C_{\mathrm{Ni}} C_{\mathrm{NiSi}}, \quad (1)$$

$$\frac{\partial C_{\mathrm{SiC}}}{\partial t} = \nabla \cdot (D^* \,\nabla C_{\mathrm{SiC}}) - k_1 C_{\mathrm{Ni}} C_{\mathrm{SiC}}, \quad (2)$$

$$\frac{\partial C_{\mathrm{C}}}{\partial t} = \nabla \cdot (D^* \,\nabla C_{\mathrm{C}}) + k_1 C_{\mathrm{Ni}} C_{\mathrm{SiC}}, \quad (3)$$

with effective heterodiffusion coefficient given by

$$D^* = D_{\mathrm{Ni}} \frac{C_\Sigma - C_{\mathrm{Ni}}}{C_{\mathrm{Ni}}}, \quad (4)$$

where C_Σ denotes the sum of all components' concentrations and D_{Ni} is the metal diffusivity. They are coupled to the reaction equations for C_{NiSi} and $C_{\mathrm{Ni_2Si}}$. Similar to [5], the initial Ni and SiC layers are assumed to be of the form

$$C_{\mathrm{Ni}}(x, t=0) = \begin{cases} C_{0,\mathrm{Ni}}, & 0 \le x \le h, \\ 0, & h < x \le L, \end{cases} \quad (5)$$

$$C_{\mathrm{SiC}}(x, t=0) = \begin{cases} 0, & 0 \le x \le h, \\ C_{0,\mathrm{Si}}, & h < x \le L, \end{cases} \quad (6)$$

where x is the depth coordinate, h is the initial metal thickness, L is the total depth, and $C_{0,\mathrm{Ni}}$ and $C_{0,\mathrm{Si}}$ are the intrinsic Ni and Si concentrations, respectively. The carbon and silicide concentrations are assumed to be zero initially. At the spatial boundaries ($x = 0, L$), zero Neumann boundary conditions are imposed to model reflecting boundaries. For all parameters, the same values as in [5, Figs. 2 and 3] have been chosen.

Such systems of equations are commonly solved numerically using the finite difference method (FDM). Note that a fine discretization both in time and space or more advanced numerical methods are required to solve the system accurately due to the non-linear structure of the diffusion coefficient (4) and the discontinuous initial conditions (5)–(6).

III. SciML – General idea and architectures

A. General idea

The general idea of SciML is to combine machine learning methods with scientific models. In order to solve a model like the one described above, the system's solution is approximated by a NN. The *training* of the NN, i.e., the optimization of its internal parameters, is based on the physical model, reformulated into an optimization task. Concretely, the governing equations are evaluated at a finite number of points at which the difference between the left-hand and right-hand sides, commonly referred to as the *loss*, is minimized. Notice that only the evaluation of the governing equations is necessary, hence allowing SciML to be applicable to equations of high complexity. The coupling of multiple equations – like the PDEs (1)–(3), initial conditions (5)–(6), and boundary conditions – is achieved by the summation of the individual losses to a composite loss function. A NN trained in this way is referred to as a *physics-informed neural network* (PINN) [6]. The composite loss function used for a PINN's training can be further extended by incorporating data-driven loss terms. This enables the PINN to simultaneously adhere to the governing physical laws and align with given data, e.g., from measurements.

B. Architectures

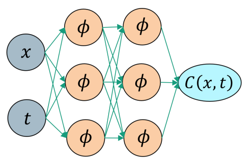
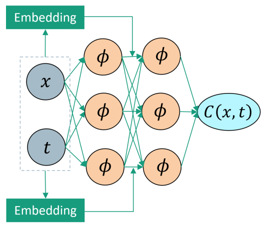
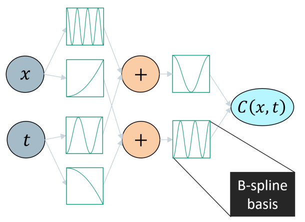
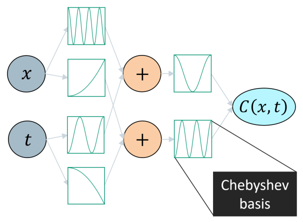
The performance of a PINN is crucially affected by the underlying type of architecture used for the NN. In this section, we describe four types of architectures commonly used in the literature, see Table I for an overview.

i. The most common architecture used for a (physics-informed) NN is a *multilayer perceptron* (MLP) [6] consisting of neurons with nonlinear activation functions that are structured in fully connected layers.

ii. To enhance the expressivity of MLPs, it has been proposed in [7] to add residual connections to the architecture. This is inspired by the so-called attention mechanism that has greatly improved NN performances [8] and helps to efficiently capture interactions between input coordinates. The resulting architecture is referred to as a *modified MLP* (mMLP).

iii. A fundamentally new architecture is the *Kolmogorov-Arnold Network* (KAN) [9]. Unlike MLPs, which learn

TABLE I
OVERVIEW OF COMMON ARCHITECTURES IN PINNS. TRAINABLE PARTS ARE HIGHLIGHTED IN GREEN.

MLP	**mMLP** [7]	**KAN** [9]	**cKAN** [10]
			
Generic neural network architecture	Additional residual connections inspired by attention mechanism	Learnable activation functions via B-spline basis representation	Learnable activation functions via Chebyshev polynomial basis

edge weights and use fixed activation functions, KANs learn the activation functions themselves, typically using a B-spline basis. KANs decompose the problem into 1D functions, enabling interpretation of trained models – an advantage over traditional MLP-based architectures.

iv. KAN architecture using an alternative basis consisting of Chebyshev polynomials that lead to *Chebyshev KANs* (cKANs) [10].

Several works have compared the use of these different types of architectures for solving PDEs [11], [12], but only for benchmark problems that significantly differ in solution properties and complexity from those encountered in TCAD-relevant use cases. The present work closes this gap by comparing these architectures for the practically relevant TCAD problem described in Section II, representative of real-world challenges in semiconductor device design.

IV. Implementation and Training

To approximate the solution of the system from Section II via SciML, the first step is to apply a suitable non-dimensionalization to ensure a stable training behavior. Concretely, the input variables in space and time as well as all output dimensions have been non-dimensionalized to the range $[0, 1]$. The non-dimensionalized PDEs are then evaluated at 10000 randomly sampled collocation points, the boundary and initial conditions at 2500 points each. The composite loss function is the equally weighted sum of the L^2-residuals of these evaluations.

For MLPs and mMLPs, the $\tanh$ activation function has been used and all combinations of $\{3, 4, 5, 6\}$ layers with $\{64, 96, 128\}$ neurons per layer have been tested. For KANs and cKANs, all combinations of $\{3, 4, 5\}$ layers with $\{32, 64\}$ and $\{64, 128\}$ nodes per layer, respectively, have been examined. The training of MLPs, mMLPs, KANs, and cKANs has been conducted for 50000, 30000, 10000, and 30000 Adam iterations, respectively, with learning rate decay (cosine annealing) [13]. All these hyperparameters have been chosen according to the values typically used in the literature and in accordance with the available computational resources.

The implementation is based on the PyTorch framework [14] and is available at [15]. The mMLP implementation is based on [7], the KAN implementations are based on the efficient implementation from [16]. The training was carried out on a consumer GPU (GeForce RTX 2060 SUPER).

V. Results

To assess the accuracy of the PINNs, we compare their predictions to a reference solution that has been computed numerically via the FDM. For each type of architecture, the relative L^2-error compared to the reference solution at the annealing time $t = 1\,\mathrm{h}$ (averaged over all concentrations) of the most accurate PINN and its respective training time are displayed in Table II; see [15] for the results of all PINNs.

TABLE II
MOST ACCURATE PINNS SETUPS FOR ALL FOUR TYPES OF ARCHITECTURES FOR MODELING SILICIDATION.

Model	Relative L^2-error	Training time
MLP (3×128)	4.4%	$1\,\mathrm{h}$
mMLP (4×128)	3.2%	$1.5\,\mathrm{h}$
KAN (4×32)	5.8%	$3\,\mathrm{h}$
cKAN (5×128)	3.9%	$8\,\mathrm{h}$

The best accuracy in our experiments is obtained by a mMLP. This shows that the additional residual connections in mMLPs indeed allow the NN to capture the silicidation process more accurately compared to generic MLPs, which are less accurate despite the use of more training iterations. The best accuracy for the cKANs is worse than for mMLPs, but slightly better than for MLPs. The KANs considered in this study exhibit lower accuracy compared to the other types of architectures. The predictions of the most accurate PINN together with the reference solution at the annealing time $t = 1\,\mathrm{h}$ are shown in Fig. 1.

It should be noted that the training times in Table II are not directly comparable due to differing NN hyperparameters. For example, the longer training time of cKAN versus KAN is because a larger NN was trained for more iterations. For networks of similar size, cKANs need fewer computational resources per iteration than KANs. Generally, the training times in Table II show that mMLPs are more computationally expensive to train than MLPs, and KAN-based architectures

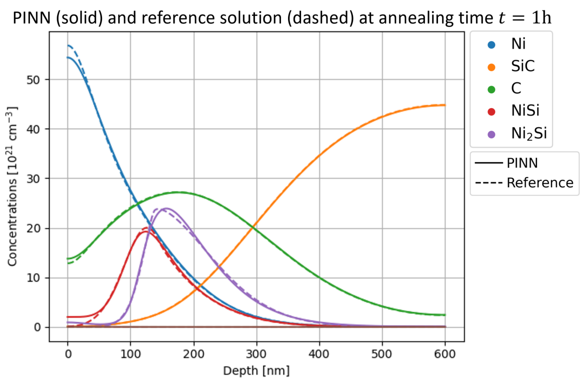


Fig. 1. Predictions of the most accurate PINN compared to the reference simulation based on the FDM.

require significantly more computation in training than MLP-based ones. Despite the higher training cost of PINNs, a key advantage is their fast evaluation compared to classical simulations. For instance, the PROMIS solver [17] (using FDM with adaptive time steps on a 2 nm grid) needs a few seconds per simulation on a single core, whereas evaluating the PINNs from Table II on the same grid for 13 timesteps (once every 5 min over 1 h annealing time) takes only a few milliseconds on a standard laptop CPU. Hence, for the present problem, PINNs offer evaluation speedups of about three orders of magnitude over classical simulations.

VI. Conclusion and Outlook

We have demonstrated the applicability of SciML to accurately model the TCAD relevant use case of Ni-SiC silicidation. Concretely, we have implemented and trained PINNs solely based on the governing physical laws of the process, without requiring measurement or simulation data. The focus of this work has been to compare the performances of different types of architectures for this task. The most accurate PINN obtained here is based on an attention-inspired extension of a generic MLP, which agrees with classical simulations up to a relative L^2-error of 3.2%. When comparing Fig. 1 with the model's calibration [5], it becomes evident that the PINN's accuracy is sufficient from an application's perspective. This demonstrates that the simulation of Ni-SiC systems is part of a plethora of applications in which SciML enables an accurate approximation of a process' solution [18].

To further improve PINN accuracy, promising directions include using feature embeddings for high-frequency phenomena and directly encoding initial as well as boundary conditions into the model structure [19]. A key opportunity of SciML is the drastically reduced evaluation time of trained models compared to conventional solvers. While standard PINNs still require costly retraining for each new configuration, this limitation can be overcome by extending them to physics-informed neural operators (PINOs) [20], which generalize across different parameter sets or initial conditions. PINOs can be evaluated rapidly for new scenarios without retraining, making them particularly advantageous for optimization tasks requiring many model evaluations.

Moreover, the composite loss function in PINN training can be extended with data-driven loss terms, allowing models to both satisfy physical laws and fit experimental or simulated data. This enables simultaneous system solving and parameter calibration based on measurements, which is particularly valuable for real-world TCAD applications.

In summary, SciML approaches open new possibilities for handling complex, coupled systems and for integrating scientific models with experimental data. These methods pave the way for scalable, data-integrated modeling of complex material interactions in semiconductor technology. Future work will focus on further advancing PINO architectures and integrating measurement data for robust, efficient simulation and optimization pipelines.

References

[1] A. May et al., *A 4H-SiC CMOS Technology enabling Smart Sensor Integration and Circuit Operation above 500 °C*, SSI 2024.
[2] H.G. Medeiros et al., *On How to Implement Experimentally Obtained Defect Characteristics in SiC Device Simulation*, IEEE IRPS 2025.
[3] H. Okeil, T. Erlbacher, G. Wachutka, *Very High In-Plane Magnetic Field Sensitivity in Ion-Implanted 4H-SiC PIN Diodes*, Adv. Electron. Mater. **10**, 2300531 (2024).
[4] Sentaurus™ *Process User Guide*, Version W-2024.09-SP1, Synopsys Inc., 2024.
[5] O.V. Aleksandrov, V.V. Kozlovski, *Simulation of Interaction Between Nickel and Silicon Carbide during the Formation of Ohmic Contacts*, J. Semicond. **43**, 885–891 (2009).
[6] 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*, J. Comput. Phys. **378**, 686–707 (2019).
[7] S. Wang, Y. Teng, and P. Perdikaris, *Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks*, SIAM J. Sci. Comput. **43**, A3055–A3081 (2021).
[8] A. Vaswani et al., *Attention is All you Need*, NIPS **30** (2017).
[9] Z. Liu et al., *KAN: Kolmogorov-Arnold Networks*, ICLR 2025.
[10] S.S. Sidharth, R. Gokul, K.P. Anas, A.R. Keerthana, *Chebyshev Polynomial-Based Kolmogorov-Arnold Networks: An Efficient Architecture for Nonlinear Function Approximation*, arXiv:2405.07200 (2024).
[11] K. Shukla, J.D. Toscano, Z. Wang, Z. Zou, and G.E. Karniadakis, *A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks*, Comput. Methods Appl. Mech. Eng. **431**, 117290 (2024).
[12] P. Niu et al., *Improved physics-informed neural network in mitigating gradient-related failures*, Neurocomputing **638**, 130167 (2025).
[13] D.P. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization*, ICLR 2015.
[14] A. Paszke et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, NeurIPS **32**, 8024–8035 (2019).
[15] X. Cheng, Diffusion-Reaction, (2025), GitHub repository, https://github.com/Boroboroda/Diffusion_Reaction
[16] Blealtan, Efficient-KAN, (2024), GitHub repository, https://github.com/Blealtan/efficient-kan
[17] P. Pichler, W. Jüngling, S. Selberherr, E. Guerrero, and H.W. Pötzl, *Simulation of critical IC-fabrication steps*, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **4**, 384–397 (1985).
[18] J.D. Toscano et al., *From PINNs to PIKANs: Recent Advances in Physics-Informed Machine Learning*, Mach. Learn. Comput. Sci. Eng. **1**, 15 (2025).
[19] C. Straub, P. Brendel, V. Medvedev, and A. Rosskopf, *Hard-constraining Neumann boundary conditions in physics-informed neural networks via Fourier feature embeddings*, ICLR 2025 Workshop Machine Learning Multiscale Processes (2025).
[20] Z. Li et al., *Physics-Informed Neural Operator for Learning Partial Differential Equations*, ACM/IMS J. Data Sci. **1**, (2024).