Model Architectures for Foundational Models in Scientific Machine Learning

Alex Kiefer

Introduction

Recent work in machine learning and artificial intelligence has demonstrated how large models are able to improve our ability to predict and learn across various[1] [2] [3] [4] [5] [6] [7] contexts. These large models, referred to as **foundational models**, reduce the cost of training accurate models from scratch by offering a broad baseline for others to build upon. The process of refining pre-trained models for a specific task is called **fine tuning**.

Though some models can generalize between multiple contexts, such as text to image, image to text, or molecules to proteins, this typically requires specific consideration of the **model architecture**. Model architecture refers to the underlying structural components that make up models, such as activation functions, loss functions, data normalization, and connectivity. This paper focuses on a field called scientific machine learning (SciML)[8]. Within this field, we focus on the specific context of solving partial differential equations (PDEs) and how model architectures affect the capabilities of foundational models. Solutions to these equations govern the dynamics of many physical systems, such as electrostatics [9], fluid dynamics [10] [11], and astrophysics [12]. Calculating these at scale with classical methods requires large supercomputers, which can run for days and require large amounts of electricity. Moreover, these systems often fail to take advantage of the highly efficient accelerators present in modern datacenters, including graphics processing units (GPU) and tensor processing units (TPU). For these reasons, foundational models for SciML present a great opportunity to reduce energy usage, calculation times, and costs for PDE solutions.

We will begin by discussing the evolution of machine learning architecture for PDE solving, focusing on how model architecture affects (i) scalability (ii) and transferability. After this, we will discuss all architectures and put them in context of their ability to be used for foundational models.

Scalability

In order for foundational models to make sense, they need to provide measurable performance advantages when compared to classical methods like the finite element method (FEM). Key considerations for AI models include memory requirements and inference time. For example, using an architecture with poor memory characteristics can make it impractical for real world use when compared to classical alternatives. Additionally, if such a large model is capable of very accurate predictions, but does not complete them on a timescale similar to that of traditional methods, then it is much less useful.

Transferability

Additionally, for foundational models to truly be "foundational", they must be able to be fine tuned to solve more specific out of distribution (OOD) problems. Without the ability to generalize, such models will be limited in the range of problems they can can solve, reducing their advantage over models trained from scratch.

Model Architectures

Multilayer Perceptron

The multilayer perceptron (MLP) model is a class of artificial neural network which is comprised of multiple layers of interconnected **neurons**. Each neuron behaves according to a defined **activation function**, which when provided an input signal, transforms the output according to this function. This is then feed forward to any n number of layers until its evaluation in the final layer. Such networks of neurons are useful because of their ability to approximate any function. This ability is defined mathematically through the **universal approximation theory** [13], which states that for any function f, there exists a multilayer perceptron with layers $\phi_1, \phi_2, ...$ such that $\phi_n \to f$.

The first application [14] of MLP's to initial and boundary problems demonstrated that by training an MLP with just one hidden layer, it is possible to find a differentiable, closed analytical solution to both PDE's and ODE's. This contrasts with traditional methods which rely on discretization to calculate solutions, which cannot be easily used for further calculations. Furthermore, this model theoretically scales to higher dimensions. This, however, along with a number of other characteristics, presents itself as a weakness in practice.

As the dimensionality of the problem increase, which in practice happens often for three and four dimensional (time dependent) systems, the memory and runtime of the model increases according to the power of the problem dimensionality. This also entails a significant increase in the number of data points required during the training of the network, which for many

physical science fields presents a large problem. If a large amount of data for a physical system already exists, then there also likely already exists a solution to the underlying dynamics of the system. Additionally, for each problem, an entirely new model needs to be trained, making such methods unusable for general foundational models. Finally, though the internal representation of the solution learned by the network is **mesh-free**, the input data for the training still needs to be provided as a grid of data (collocation points), which if incorrectly posed, can lead to poor solutions.

Physically Informed Neural Networks (PINNs)

PINNs [15] approach the problem of increasingly large training dataset requirements by introducing constraints upon a model based on already known principles. By incorporating partial differential equations that describe the physics of the system into model training, the solution to an unknown system can be found with orders of magnitude less data than a traditional neural network. This also has the benefit of providing robustness to noisy data, which is the reality for most real-world applications. Another benefit of including the physical knowledge of the system is that is allows for solutions to the **inverse problem**, which is the discovery of previously unknown parameters of the system.

Though PINNs are a definite improvement over ANNs, they still suffer from many of the same problems, namely, the curse of dimensionality, large training data sizes, lack of scalability, and lack of transferability between problems. Perhaps most importantly, there doesn't yet exist theoretical proof that PINNs always converge to the global minimum of the system, though they have shown empirical results which seem to indicate that they do. This is particularly important when dealing with problems which are ill-posed or unstable, which may result in incorrect solutions.

Neural Operators (NO)

NOs are an extension of classical neural networks to **operator** learning. Rather than learning the specific solution to a given problem, as PINNs and ANNs do, NOs learn the operator, or the function, which maps the input space to the output space. This shift in the objective of the problem enables NOs to generalize between many different PDE's without retraining. Additionally, certain NOs can generalize across different problem resolutions, where methods like PINNs and ANNs, though mesh-free, still depend upon the input data's collocation points for training. However, NOs, as first posed in [16], still lack the scaling capabilities necessary for large foundational models. This will be the focus of the following variants.

DeepOnet

DeepOnet [16] is the earliest work demonstrating that it is possible to efficiently learn nonlinear operators. DeepOnet consists of two subnetworks, one called the **branch** and the other the **trunk**. The branch network encodes the input function $u(x)$ at certain fixed locations called **sensors**, while the trunk network encoded the output location where the learned operator $G(u)(y)$ is evaluated. The dot product between these two subnetworks is then taken, providing the prediction of the overall network.

DeepONet was the first architecture to demonstrate the ability to learn nonlinear operators. The separation of input functions and output locations into two distinct subnetworks enables the model to handle high-dimensional, complex systems with flexibility and generality. This design allows DeepONet to generalize well across different PDEs and boundary conditions, providing an effective method for solving both forward and inverse problems. Additionally, DeepONet is capable of handling noisy or sparse data efficiently, making it particularly suitable for real-world applications where data might be limited or irregular. Its structure enables the model to learn an entire family of operators, allowing it to be reused in different contexts without retraining for each specific PDE instance.

Despite these strengths, DeepONet has several limitations. One of the main challenges is its dependence on the fixed sensor locations in the branch network, which can hinder the models ability to generalize when the input function is sampled differently or when new locations are introduced. This can result in suboptimal performance in tasks where the input data distribution changes. Additionally, DeepONet struggles with irregular domains or geometries that are not easily represented by the fixed sensor locations, limiting its applicability to complex, unstructured meshes. Finally, the method requires a large number of training samples to effectively learn the operator, especially in high-dimensional problems, making it less efficient when data is scarce.

Fourier Neural Operators (FNO)

The following section is based upon the original work in [17].

FNOs significantly enhance the performance of operator learning by leveraging the Fourier transform, which can be computed in $O(n \log(n))$ time. This transform encodes the global interactions present in partial differential equations (PDEs). Although the input to FNOs consists of grid-based spatial data, it is immediately transformed from the spatial domain to the frequency domain via a Fourier layer. In this domain, the low-frequency modes, which capture global interactions, undergo a linear transformation. Conversely, the high-frequency modes, which represent finer local details, are either ignored or subjected to smaller transformations to retain efficiency. After these transformations, the data is projected back into the spatial domain, where it is passed through non-linear layers, typically with ReLU activations, to capture complex local behaviors in the system. This process is repeated over several layers,

with the number of layers acting as a user-defined hyperparameter that controls the depth of the network and its capacity to model complex PDE dynamics.

FNOs are by far the most popular architecture in the operator learning spaces as they address three important weaknesses present in DeepOnet. The first, which all prior models lacked, was mesh invariance. FNOs do not require discretized collocation points when training which allows for generalization across grid resolutions in downstream predictions without retraining. This is especially important for learning across different problems, as the resolution of training data can now vary, enabling the used of mixed resolutions in the same model. This full mesh invariance also results in the ability to perform **super resolution** on the data. This is the ability to take low resolution input data during training and make predictions at resolutions higher than those of the input. FNOs also exhibit the first true accuracy gains over PINNs and discrete CFD methods for high dimensional turbulent flows, making them the state-of-the-art method for solving such problems. Finally, and most importantly, the do not suffer from the curse of dimensionality, making them optimal for scaling.

The use of the Fourier transform in FNOs introduces some challenges for this architecture. While the Fourier transform is highly effective for certain types of problems, particularly those with periodic boundary conditions, it struggles with non-periodic boundaries. Though mitigations like padding can help, they do not fully resolve the issue, and non-periodic boundaries remain problematic. Furthermore, the strength of the Fourier transform lies in decomposing the problem into global components, which means systems dominated by local behaviors, such as those with sharp discontinuities, are often poorly handled by this method. This is because sharp transitions are typically represented by high-frequency components, which are either ignored or underrepresented in the Fourier-based approach, reducing the method's accuracy in such cases.

U-FNO

U-FNO [18] incorporates elements from computer vision, namely a form of **convolutional neural network**(CNN) called a U-Net, into the FNO method to improve performance on problems with more local dynamics. The first modification to the architecture is the introduction of a **lifting layer**, which increases the input dimensionality of the data. In doing so, this allows the low mode frequency modes of the fourier transform to include more information about the local dynamics of the system. The non-linear layer after each fourier layer is then replaced by a U-fourier layer, which is a U-Net that applies local convolutions to the output of the fourier layer to capture high frequency information about the system. The final layer then projects the output back into the original input dimension.

This addition of U-Nets into the architecture improves the accuracy of the model for more complex systems greatly, particularly those that the traditional FNO had trouble solving.Though this method improves upon the weaknesses of FNOs for predicting local dynamics, it does so at the cost of reintroducing mesh-independence, which was one of the largest successes of the

original FNO architecture. For this reason, this architecture isn't suitable for scaling and is more successful as a domain specific method for multiphase flows[18].

Transformers

The transformer architecture was first introduced in the context of NLP in [19]. The core mechanism of transformers is called **self-attention**, which allows the model to weigh the importance of different parts of the input sequence dynamically, based on the relationships between tokens, enabling it to capture context more effectively. Traditional transformers are modeled as an encoder-decoder architecture. The **encoder** takes as its input a sequence of values, such as words, and through a series of self-attention layers and feedforward networks, transforms this sequence into a set of hidden representations (or feature vectors). Each token in the sequence attends to every other token, capturing dependencies regardless of their distance in the input.

The **decoder** also uses self-attention to generate predictions, but it does so in an **auto-regressive** manner, meaning it generates one token at a time, using the previously generated tokens to inform the prediction of the next one. It takes the hidden representations from the encoder along with the tokens generated so far as input, learning to predict the next token in the sequence. This process continues until the entire output sequence is generated. The architecture is designed to process both the input and output sequences in parallel, leveraging attention mechanisms to efficiently handle long-range dependencies.

Such transformers have been adapted to the domain of PDE solving through a reinterpretation of the self-attention mechanism to handle spatial and temporal dependencies in physical systems. In their original form, transformers use attention to capture relationships between words or tokens in a sequence, but for PDEs, this idea is extended to capture interactions between points in a discretized spatial domain. Instead of focusing on tokens, attention is applied to physical states or grid points, allowing the model to learn complex relationships between spatial locations and time steps. Variants such as the Fourier Transformer [20] introduced the idea of replacing traditional dot-product attention with Fourier transforms to capture global dependencies in regular grids, making transformers more efficient for structured PDE problems. Other adaptations, like the Galerkin Transformer[20], reformulate attention as a Galerkin projection, allowing efficient handling of irregular meshes and unstructured domains, making them suitable for complex geometries. These adaptations enable transformers to generalize across different PDEs by learning operators, enhancing their scalability and efficiency in solving scientific problems.

GrapH TrAnsforMer NeuraL OpEraTor (HAMLET)

Graph neural networks (GNNs) [21] are a kind of neural network designed to work efficient on graph data structures, where information is represented as a set of nodes and edges

connecting them. By training three separate networks, one each for nodes, edges, and global graph structure, predictions about each of these qualities can be made while incorporating information from each of the networks. This is done by pooling information about each of these attributes into a pooling layer and preforming **message passing** to train the GNN. Message passing is the process of distributing local information about the graph to neighboring nodes and aggregating this together. However, this method, when applied to real world problems, has trouble capturing the global interactions of the system.

The application of GNNs to operator learning is presented through an interesting manipulation of the kernel formulation of a general PDE. For a general PDE of the form $\mathcal{L}u = f$, where $\mathcal{L}$ is a differential operator that maps the function u to f, the kernel integral can be formulated as an iterative solver, such that u is approximated through u_t timesteps $1 - t$. Finally, by discretizing the kernel integral and approximating it is a sum, we are left with a formulation of the PDE that is equivalent to the message passing mechanism of GNNs. By formulating our input mesh as a graph describing the boundary of the problem, GNNs can be used to find solutions to the operator learning problem.

In this way, HAMLET [22] is able to learn the neural operator for a given graph (irregular mesh). However, on its own, a GNN doesn't provide effective solutions to the global interactions in such problems. In order to correct this, HAMLET introduces transformer layers along with the traditional GNN layers of the network. Transformers, which rely upon the mechanism of self-attention, are able to effectively make global, non-local predictions on the mesh by calculating an attention score for all pairwise combinations of nodes. This is then convolved with the output of the GNN to produce a solution operator that effectively captures both local and global dynamics of the problem.

However, just as U-FNOs reintroduced dependence upon meshes, so does HAMLET. Though not entirely the same, HAMLET is **mesh-flexible** rather than mesh invariant, as though it does require the input data to be represented as a mesh, this mesh does not have to be a regular grid, as many prior methods require. This makes scaling and transferability of this architecture much more difficult than for FNOs.

Transolver

The most recent contribution to PDE solution architectures, Transolver is a flexible, transformer based architecture. This section is all based on the original text [23].

Transolver is capable of solving both structured and unstructured problems. The most significant advancement in this architecture is the domain specific formulation of attention in the transformer. **Physical attention** simplifies the complexity of solving PDEs on inputs by grouping points with similar physical properties like temperature, pressure, or velocity together. Such points are grouped together into slices, where the classical form of attention is applied to all points in this region. This process is adaptive, allowing for its application to PDEs within different domains.

This greatly reduces the computational complexity of the problem from the traditional $O(n^2)$ to approximately $O(n)$. For this reason, Transolver provides a highly efficient method for operator learning, especially on irregular geometries, where grouping points together simplifies the problem in the most significant way. Additionally, physical attention provides a more interpretable formulation of the solution, as the interaction of slices can be understood within the physical context of the problem, rather than all points on the mesh. Its partitioning of input points also enables strong OOD performance, as by grouping physically similar points together, complex boundary conditions on unseen problems can be learned based on the context of physically similar slices. This context also greatly benefits when learning highly dissimilar physical problems, such as fluid dynamics with new flow behaviors, as rather than the traditional spatial context of transformers for PDE solving, points are learned in context of their physical properties.

Transolver present a very efficient method for operator learning, but still struggles in certain respects. Most obviously, physical attention is only effective if the problem being solved has groups of points with similar physical properties. For highly heterogenous systems with large amount of variation between points, this formulation of attention does not present as much of a benefit in a performance context when compared to the classical single point attention mechanism. Furthermore, though the transformer layer of this model is reduced in complexity through the use of point aggregation into slices, this introduces a separate step into the training process that has its own computational cost. Though this is still more computationally efficient than traditional attention, it introduces additional complexity into the process of model training that isn't present for architectures like PINNs and FNOs. Finally, though point aggregation into slices makes learning more efficient, it comes at the cost of potentially losing fine grained details in the mesh. For problems with highly localized behavior like turbulence or phase transitions, grouping points together may result in the loss of important local behavior of the system.

Discussion

In SciML, the emergence of foundational models has revolutionized how we approach the modeling and solution of complex physical systems, especially those governed by partial differential equations (PDEs). These models promise to provide general, reusable architectures that can be fine-tuned for specific tasks, reducing the need for training models from scratch for each new problem. Key foundational model architectures, such as Multilayer Perceptrons (MLPs), Physically Informed Neural Networks (PINNs), Neural Operators (NOs), and Transformers each offer distinct advantages and challenges when applied to SciML tasks, particularly in the context of scalability and transferability.

The MLP architecture, though foundational in its ability to approximate any function through universal approximation theory, suffers from scalability issues as problem dimensionality increases. The memory and runtime demands grow exponentially with the number of dimensions,

making it impractical for high-dimensional PDEs often encountered in scientific applications. This is especially problematic for foundational models, which require efficiency across a range of tasks. Furthermore, MLPs lack transferability, requiring a new model to be trained for each specific problem. In contrast, PINNs attempt to address the data demands by incorporating physical laws directly into the training process, reducing the need for extensive datasets. However, despite the gains in data efficiency, PINNs still struggle with the curse of dimensionality, limiting their scalability for more complex problems. Moreover, PINNs lack strong theoretical guarantees for convergence to the global minimum, and their transferability between different problem domains remains limited. For these reasons, this class of architectures, though crucial in the evolution of PDE solving, do not present the necessary requirements for the basis of a foundational model.

Neural Operator (NO) models, such as DeepONet and Fourier Neural Operators (FNOs), introduce a significant shift in SciML by learning operators rather than specific solutions. This enables greater generalization across PDEs, providing foundational models that can be reused without retraining for each new instance of a problem. In DeepONet, by separating input functions and output locations into branch and trunk networks, it is possible to efficiently learn nonlinear operators. However, this is limited by its dependence on fixed sensor locations, hindering its generalization to irregular domains. FNOs, which utilize the Fourier transform to handle global interactions in PDEs, offer mesh invariance, enabling them to generalize across grid resolutions and perform super-resolution tasks. This scalability makes FNOs well-suited for foundational models. However, FNOs struggle with non-periodic boundary conditions and sharp discontinuities, reducing their effectiveness for localized phenomena. U-FNOs attempt to correct this weakness by integrating U-Net layers to capture more local dynamics, but reintroduces mesh dependence, limiting their scalability and transferability compared to FNOs. This class of architectures has shown the most promise for foundational models, with the majority of existing foundational models, [8] [24] [25] [26] being built upon variants of the FNO architecture.

Recent advances in the adaptation of transformers to SciML have further enhanced the ability of foundational models to handle a wider range of scientific problems. HAMLET, which combines Graph Neural Networks (GNNs) and transformers, demonstrates powerful performance on irregular domains, particularly for problems involving complex geometries. However, HAMLET's mesh flexibility, while allowing it to handle unstructured meshes, adds complexity to scaling and transferability. Transolver, with its novel Physical Attention mechanism, represents one of the most recent innovations, reducing the computational complexity from $O(n^2)$ to $O(n)$ by grouping physically similar points into slices for more efficient learning. This approach greatly enhances its scalability and out-of-distribution (OOD) performance for large and complex systems but may struggle in highly heterogeneous systems where points are dissimilar, limiting its full potential for certain classes of problems. With their ubiquity in other domains [4] [3] [5], transformers have *transformed* the field of artificial intelligence with their impact. This has incentivized hardware manufacturers to optimize hardware [27] to efficiently handle such architectures. For this reason, transformers present a promising architecture for future foundational models.

Conclusion

Foundational models in SciML are rapidly evolving, with neural operator-based methods leading the way in terms of scalability, mesh independence, and transferability. While traditional architectures like MLPs and PINNs laid the groundwork, newer architectures like FNOs, DeepONet, and advanced models such as Transolver are pushing the boundaries of what can be achieved, offering more generalizable and efficient solutions for a broad range of scientific problems. However, creating truly foundational models in SciML still presents significant challenges, particularly when it comes to data availability and the need to balance the tradeoffs between various architectures.

Foundational models require large and diverse datasets for pre-training to generalize effectively across different physical systems, but collecting high-quality, labeled scientific data can be difficult, costly, and time-consuming. For many scientific domains, particularly in the context of rare or expensive experiments, such as in climate science [28], materials discovery [29], or astrophysics [30], the lack of sufficient data becomes a bottleneck. Synthetic data generation through simulations helps, but these models need to handle the inherent noise and discrepancies between simulated and real-world data, further complicating their performance and generalizability.

Moreover, there is a delicate balance in choosing the right architecture depending on the specific problem domain. Models like FNO offer scalability and mesh invariance but may struggle with sharp discontinuities or highly localized phenomena. On the other hand, architectures such as U-FNO and HAMLET, which excel at capturing local dynamics and handling irregular domains, often come at the cost of increased computational complexity or a return to mesh dependence. DeepONet, with its ability to learn nonlinear operators efficiently, faces challenges in irregular domains and when required to generalize across non-standard inputs. Meanwhile, Transolver introduces innovations like Physical Attention that improve scalability and out-of-distribution performance but may face limitations when dealing with highly heterogeneous systems.

For these reasons, the key to developing successful foundational models lies in balancing tradeoffs between data efficiency, computational cost, generalizability, and handling of irregular or complex domains. Achieving this balance requires innovation in model architectures, better integration of physics-informed constraints, and improvements in data augmentation techniques. Only by addressing these challenges can foundational models fully deliver on their promise to revolutionize scientific discovery and simulation.

[1] M. Abdin *et al.*, "Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone." arXiv, Aug. 2024. doi: 10.48550/arXiv.2404.14219.

[2] J. Abramson *et al.*, "Accurate structure prediction of biomolecular interactions with AlphaFold 3," *Nature*, vol. 630, no. 8016, pp. 493–500, Jun. 2024, doi: 10.1038/s41586-024-07487-w.

[3] A. Dubey *et al.*, "The Llama 3 Herd of Models." arXiv, Aug. 2024. doi: 10.48550/arXiv.2407.21783.

[4] OpenAI *et al.*, "GPT-4 Technical Report." arXiv, Mar. 2024. doi: 10.48550/arXiv.2303.08774.

[5] R. Anil *et al.*, "PaLM 2 Technical Report." arXiv, Sep. 2023. Accessed: Sep. 10, 2024. [Online]. Available: https://arxiv.org/abs/2305.10403

[6] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved Baselines with Visual Instruction Tuning." arXiv, May 2024. doi: 10.48550/arXiv.2310.03744.

[7] T. Brooks *et al.*, "Video generation models as world simulators," 2024.

[8] S. Subramanian *et al.*, "Towards Foundation Models for Scientific Machine Learning: Characterizing Scaling and Transfer Behavior," *Advances in Neural Information Processing Systems*, vol. 36, pp. 71242–71262, Dec. 2023.

[9] C. G. Gray and P. J. Stiles, "Nonlinear electrostatics: The Poisson–Boltzmann equation," *European Journal of Physics*, vol. 39, no. 5, p. 053002, Jul. 2018, doi: 10.1088/1361-6404/aaca5a.

[10] E. Pardoux and Y. Veretennikov, "On the Poisson Equation and Diffusion Approximation. I," *The Annals of Probability*, vol. 29, no. 3, pp. 1061–1085, Jul. 2001, doi: 10.1214/aop/1015345596.

[11] È. Pardoux and A. Y. Veretennikov, "On Poisson equation and diffusion approximation 2," *The Annals of Probability*, vol. 31, no. 3, pp. 1166–1192, Jul. 2003, doi: 10.1214/aop/1055425774.

[12] W. Schmidt, "Large Eddy Simulations in Astrophysics," *Living Reviews in Computational Astrophysics*, vol. 1, no. 1, p. 2, Oct. 2015, doi: 10.1007/lrca-2015-2.

[13] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989, doi: 10.1016/0893-6080(89)90020-8.

[14] M. Magill, F. Qureshi, and H. de Haan, "Neural Networks Trained to Solve Differential Equations Learn General Representations," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2018.

[15] 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*, vol. 378, pp. 686–707, Feb. 2019, doi: 10.1016/j.jcp.2018.10.045.

[16] L. Lu, P. Jin, and G. E. Karniadakis, "DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators," *arXiv.org*. https://arxiv.org/abs/1910.03193v3, Oct. 2019. doi: 10.1038/s42256-021-00302-5.

[17] Z. Li *et al.*, "Fourier Neural Operator for Parametric Partial Differential Equations," *arXiv.org*. https://arxiv.org/abs/2010.08895v3, Oct. 2020.

[18] G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson, "U-FNO—An enhanced Fourier neural operator-based deep-learning model for multiphase flow," *Advances in Water Resources*, vol. 163, p. 104180, May 2022, doi: 10.1016/j.advwatres.2022.104180.

[19] A. Vaswani *et al.*, "Attention is All you Need."

[20] S. Cao, "Choose a Transformer: Fourier or Galerkin," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2021, pp. 24924–24940.

[21] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, PMLR, Jul. 2017, pp. 1263–1272.

[22] A. Bryutkin, J. Huang, Z. Deng, G. Yang, C.-B. Schönlieb, and A. Aviles-Rivero, "HAMLET: Graph Transformer Neural Operator for Partial Differential Equations," *arXiv.org*. https://arxiv.org/abs/2402.03541v1, Feb. 2024.

[23] H. Wu, H. Luo, H. Wang, J. Wang, and M. Long, "Transolver: A Fast Transformer Solver for PDEs on General Geometries," *arXiv.org*. https://arxiv.org/abs/2402.02366v2, Feb. 2024.

[24] Z. Song, J. Yuan, and H. Yang, "FMint: Bridging Human Designed and Data Pretrained Models for Differential Equation Foundation Model." arXiv, Sep. 2024. Accessed: Oct. 02, 2024. [Online]. Available: https://arxiv.org/abs/2404.14688

[25] M. Herde *et al.*, "Poseidon: Efficient Foundation Models for PDEs." arXiv, May 2024. Accessed: Sep. 10, 2024. [Online]. Available: https://arxiv.org/abs/2405.19101

[26] T. Chen *et al.*, "Building Flexible Machine Learning Models for Scientific Computing at Scale," *arXiv.org*. https://arxiv.org/abs/2402.16014v1, Feb. 2024.

[27] D. Salvator, "H100 Transformer Engine Supercharges AI Training, Delivering Up to 6x Higher Performance Without Losing Accuracy," *NVIDIA Blog*. https://blogs.nvidia.com/blog/h100-transformer-engine/, Mar. 2022.

[28] J. Tollefson, "Climate scientists push for access to world's biggest supercomputers to build better Earth models," *Nature*, Jul. 2023, doi: 10.1038/d41586-023-02249-6.

[29] A. Merchant, S. Batzner, S. S. Schoenholz, M. Aykol, G. Cheon, and E. D. Cubuk, "Scaling deep learning for materials discovery," *Nature*, vol. 624, no. 7990, pp. 80–85, Dec. 2023, doi: 10.1038/s41586-023-06735-9.

[30] NASA, "Research Opportunities in Space and Earth Science (ROSES)-2023 Released - NASA Science." https://science.nasa.gov/researchers/solicitations/roses-2023/research-opportunities-space-and-earth-science-roses-2023-released/.