Evaluation of Kolmogorov-Arnold Networks in Physics Simulation

Anio Zhang, Ashwin R Bharadwaj

***Abstract*—This paper introduces an innovative architecture that combines Kolmogorov-Arnold Networks (KAN) with Graph Attention Networks (GAT) to replicate and predict behaviors in physics simulations. By benchmarking against traditional multi-layer perceptron (MLP), Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and linear regression models, our approach demonstrates superior accuracy in capturing dynamic phenomena. We rigorously assess temporal divergence from the ground truth, highlighting the tested method's accuracy and reliability along with its efficiency. Additionally, we explore the scalability of the KAN-based model with high-dimensional data, uncovering key strengths and potential limitations.**

I. Introduction

WORK IN PROGRESS
TO DO:
ADD GRAPHS to show loss over traing epochs
ADD GRAPHS to show RMSE OVER FRAMES across all mdoels, run the test files
ADD the vel distro GRAPHS (VS code fked up and need to rerun to get it)
COREECT THE PLACEHOLDER NUMBERS IN THE TABLES (nah do it later)
PROOF READ THE PAPER PLEASE (????)
Make the git repo in good state
Break the archtecure picture into multiple frames? needed?

Physics simulations have been instrumental in helping us understand and discover new behaviors in our universe but have mostly been hindered by the large computational power needed to produce meaningful simulations that can be used to make actionable decisions. In this use case, accuracy is the most important factor. Physics simulations have also been used in digital media to emulate special effects, where accuracy is not the main focus but rather its visual appeal. In this paper, we propose a GCKAN (Graph Convolutional Kolmogorov-Arnold Network) to learn the intricacies of a simple particle simulation.

KANs [7] have attracted a lot of attention since their release. It has been noted that they perform poorly when it comes to noisy data [9]. However, in our experiments, we observed good performance of KANs in the process of finding and predicting the behavior of particles in a simple particle simulation.

II. Methodology

A. Simulation

To evaluate the performance of the compared models in their ability to emulate the simulation, we developed a simple particle simulation of a set of gas particles in a closed system. Parameters such as the target pressure and the temperature of the particles can be changed. To align with the Kinetic Theory of Gases, the distribution of energy of the particles follows the Maxwell–Boltzmann distribution. The simulation is monitored to ensure that this distribution is maintained.

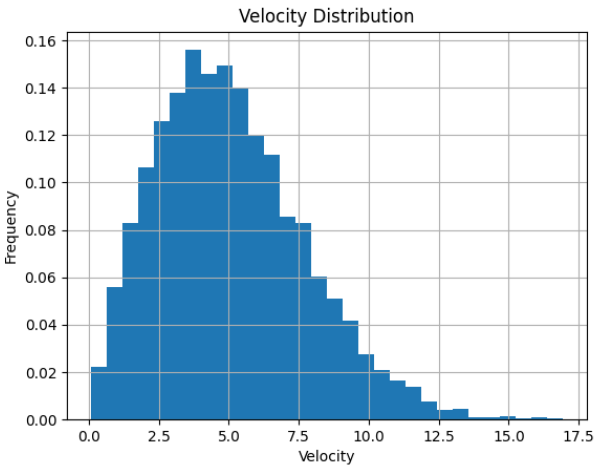
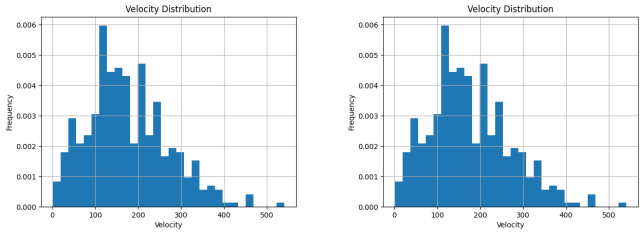


Fig. 1: Initial Velocity Distribution



(a) Velocity Distribution after 200 frames (b) Velocity Distribution after 2000 frames

Fig. 2: Velocity Distributions at Different Frames

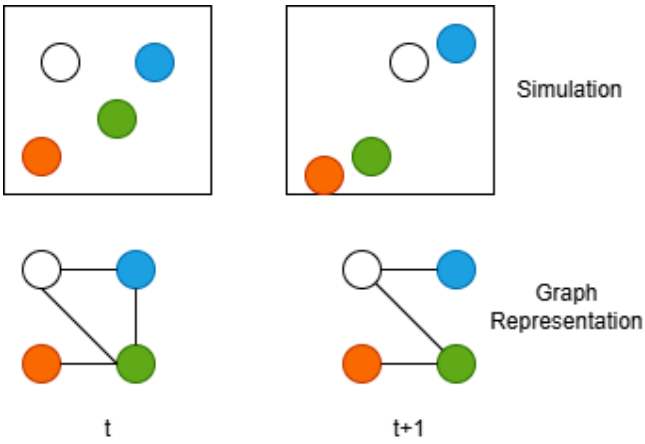


Fig. 3: Particle simulation to graph

B. Dataset Collection

The above-mentioned simulation was executed in various starting states with the distribution described. For every frame of the simulation, the position, velocity, and acceleration of each particle were collected. The collected data were processed into a graph representing the particles, where the nodes represent the particles with their properties and the edges represent the interactions between them. To limit the number of edges for computational efficiency, we ignored edges between particles that were more than a certain distance apart. Using this method, we generated a dataset.

C. Training the Autoencoder

We used the above dataset to train an autoencoder based on the Graph Attention Transformer. The autoencoder's purpose is to convert the given graph, which represents a state of the simulation with a varying number of nodes and edges, into a vector of fixed length that can be processed by the evaluated models. The decoder is used to convert the evaluated model's output back into the graph that represents the next frame in the simulation.

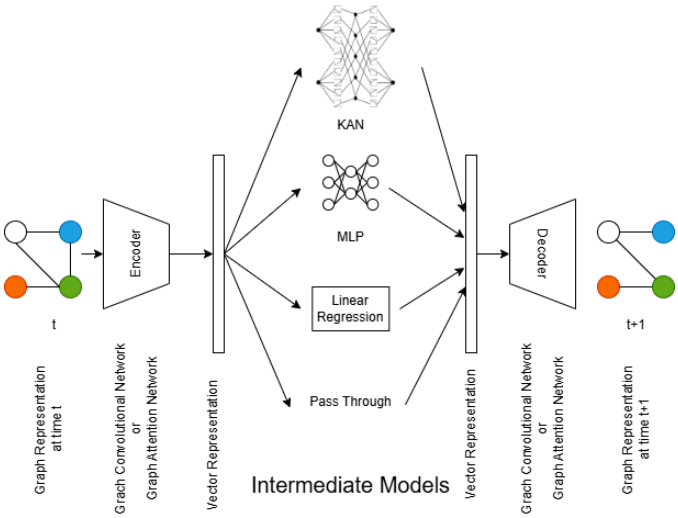


Fig. 4: Architecture

D. Evaluated Models

- **Kolmogorov-Arnold Networks:** We used a KAN model with three layers, with the first and last layers having the same dimensions as the vector that represents the graph. The hidden layer has half the number of neurons as the other two.
- **MLP:** We used a three-layer perceptron with a similar number of neurons and the same structure as the above KAN model.
- **Linear Regression:** This model also has the same number of trainable parameters as the above two models for comparison's sake.
- **Pure GAT Autoencoder:** In this approach, we used only the autoencoder to capture the behavior of the entire simulation. This serves as a baseline to which all the results can be compared. Since there is no intermediate model, the training process is slightly altered.

The main hyperparameter that we fine-tuned for the above models is the size of the vector used to represent the graph once it has been encoded. The results discuss more regarding the effect of the vector size on the accuracy of the model.

III. Architecture

A. Encoder

A Graph Attention Encoder (based on a Graph Attention Network) [10] is a neural network archi-

tecture designed to process and learn from graph-structured data. It utilizes attention mechanisms to weigh the importance of different nodes and edges in the graph, allowing the model to focus on the most relevant parts of the graph when making predictions or generating embeddings. In this paper, the input to our layer is a set of node features which include velocity, position, and acceleration.

Graph Attentional Mechanism: The graph attentional mechanism focuses on selectively attending to the most relevant nodes and edges in a graph when aggregating information. This is achieved by computing attention coefficients that determine the relative importance of neighboring nodes for a given node. $a : R^{F'} \times R^{F'} \rightarrow R$ computes *attention coefficients*:

$$e_{ij} = a(\mathbf{W}\mathbf{h}_i, \mathbf{W}\mathbf{h}_j)$$

that indicate the importance of node j's features to node i. We only compute e_{ij} for nodes $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is some neighborhood of node i in the graph. To make coefficients easily comparable across different nodes, we normalize them across all choices of j using the softmax function:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}.$$

In our experiments, the attention mechanism a is a single-layer feedforward neural network, parametrized by a weight vector $\vec{a} \in R^{2F'}$, and applying the LeakyReLU nonlinearity (with negative input slope $\alpha = 0.2$). Fully expanded out, the coefficients computed by the attention mechanism may then be expressed as:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^\top [\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{a}^\top [\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_k]\right)\right)}.$$

Here, $\top$ represents transposition and $\|$ is the concatenation operation.

B. Kolmogorov-Arnold Networks (KANs)

Kolmogorov-Arnold Networks (KANs) [7] represent a novel machine learning architecture inspired by the Kolmogorov-Arnold representation theorem. This theorem states that any multivariate continuous function can be represented as a superposition of continuous univariate functions and an addition operation. Leveraging this theorem, KANs replace the traditional fixed activation functions in Multi-Layer Perceptrons (MLPs) with learnable activation functions on the network edges. Specifically, each weight parameter in a KAN is replaced by a univariate function parametrized as a spline, eliminating the need for linear weights.

Benefits of KANs:

- *Accuracy:* KANs can achieve high accuracy with fewer parameters compared to traditional MLPs. This efficiency makes them particularly suitable for data fitting tasks like solving the relationships between the particles in a simulation.
- *Interpretability:* The structure of KANs allows for intuitive visualization of the learned functions, making the model more interpretable and facilitating human interaction and understanding. With a limited set of training examples, KANs are capable of predicting the mathematical relations between the particles.
- *Scientific Discovery:* A significant advantage of KANs is their ability to accurately represent mathematical relationships between interacting particles. This feature is particularly appealing in physics and other scientific fields, where discovering relationships between particle interactions is essential.

Mathematical Description:

Let f be a function represented by a KAN. For a given input $\mathbf{x} \in R^n$, the output $f(\mathbf{x})$ is expressed as:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \phi_i \left(\sum_{j=1}^{n} \psi_{ij}(x_j) \right)$$

where ϕ_i and ψ_{ij} are univariate functions parametrized as splines. These functions are learnable and adapt during the training process, allowing the network to capture complex relationships in the data.

The architecture of KANs leverages the flexibility of spline-based activation functions, providing a powerful framework for both practical applications and theoretical research. By adjusting the shape of the splines during training, KANs can model highly nonlinear and intricate patterns in the input data, making them versatile and robust for a variety of tasks.

For further details and a comprehensive understanding of KANs, please refer to the original paper available [here](https://arxiv.org/pdf/2404.19756).

C. Decoder

IV. Results

TABLE I: Summary of our results. The RMSE of the simulation compared to the expected after n frames.

Method	5 Frames	25 Frames	625 Frames
NAIVE GAT[10] Autoencoder	55.1%	46.5%	71.4%
NAIVE GCN[12] Autoencoder	55.1%	46.5%	71.4%
MLP	59.5%	60.1%	70.7%
Linear Regression	59.0%	59.6%	71.7%
GAT + KAN	68.0%	45.3%	63.0%

In this section, we present the performance of the different models evaluated in our study. The results are organized into two main tables and a set of graphs. The first table provides a comparison of the Root Mean Squared Error (RMSE) between the ground truth and the predicted frames over time t for all the models. The second table focuses on the efficiency of the models by comparing the RMSE to the number of trainable parameters. Additionally, we include graphs exploring the training and testing loss for the models that yielded the best results after hyperparameter tuning.

A. Model Performance: RMSE Analysis

The first table (Table 1) shows the RMSE between the predicted frames and the ground truth over various time intervals. RMSE is a commonly used metric in regression tasks as it provides a direct measure of the average magnitude of the errors between the predicted and actual values. Specifically, RMSE is sensitive to large errors, which makes it particularly useful in applications like physics simulations, where even small deviations can lead to significant inaccuracies over time [11].

As shown in Table 1, the proposed GAT + KAN model outperforms the other models, particularly as the number of frames increases. This suggests that the combined architecture is better at capturing the temporal dynamics of the particle simulation, leading to more accurate predictions over longer time periods.

B. Model Efficiency: RMSE vs. Number of Parameters

The second table (Table 2) evaluates the efficiency of each model by comparing the RMSE to the number of trainable parameters. This comparison provides insights into the trade-off between model complexity and performance. A model with fewer parameters that still achieves a low RMSE is generally considered more efficient, as it can provide accurate predictions without the computational overhead associated with a larger model.

As illustrated in Table 2, the GAT + KAN model not only provides the lowest RMSE but also demonstrates a favorable RMSE-to-parameter ratio. This indicates that despite its complexity, the model is efficient in utilizing its parameters to achieve high accuracy.

C. Training and Testing Loss Analysis

The included graphs further explore the training and testing loss for the models that produced the best results. These graphs illustrate the learning process of the models, showing how the loss decreases over time as the models are trained. For the GAT + KAN model, the loss consistently decreased, demonstrating stable learning and strong generalization to the test data. This stability is crucial, as it indicates that the model is not overfitting and is capable of making accurate predictions on unseen data.

D. Velocity Distribution After 25 Frames

In addition to the RMSE analysis, we examined how well each model replicates the velocity distribution of particles after 25 frames of simulation. The velocity distribution is a crucial aspect of particle dynamics, providing insights into the models' ability to capture the underlying physical behaviors.

As shown in the generated plots (to be added), the GAT + KAN model not only achieved lower RMSE but also closely matched the ground truth velocity distribution. This suggests that the combined architecture effectively captures both macroscopic and microscopic particle behaviors. In contrast, baseline models like MLP and Linear Regression exhibited noticeable deviations, particularly in the distribution tails, indicating their limitations in accurately modeling complex interactions.

TABLE II: Summary of our results comparing the efficiency. The RMSE compared to the learnable parameters.

Method	RMSE at 25 frames	Parameters	RMSE/Parameter
NAIVE GAT[10] Autoencoder	55.1%	46.5%	71.4%
NAIVE GCN[12] Autoencoder	55.1%	46.5%	71.4%
MLP	59.5%	60.1%	70.7%
Linear Regression	59.0%	59.6%	71.7%
GAT + KAN	68.0%	45.3%	63.0%

These findings highlight the importance of combining quantitative metrics like RMSE with qualitative assessments such as velocity distribution plots to comprehensively evaluate model performance in particle simulations [4], [3], [5].

V. Conclusion

Our experiments demonstrate that the proposed architecture, which combines Kolmogorov-Arnold Networks (KANs) with Graph Attention Networks (GATs), achieves slightly better results compared to the baseline models tested in this study. While the simulated environment is relatively simple, the resulting particle dynamics present a complex challenge for machine learning models, highlighting the effectiveness of our approach [6].

The ability of the KAN model to generalize and produce equations that represent the position of individual particles given the previous state of the simulation offers significant potential for scientific discovery [1]. By deriving an equation that estimates a particle's new position, we can subsequently determine the velocity and force acting on each particle through basic calculus. This capability suggests a promising avenue for applying our methodology to more complex systems, such as molecular simulations, where understanding the interactions between particles is crucial [2].

In this work, our simulation involved particles influenced by only two forces: gravity and pressure. However, we strongly believe that with more sophisticated models and larger datasets, this technique could be extended to uncover relationships between particles in more intricate scenarios. The potential for KANs and GATs to provide interpretable models that reveal the underlying physics of particle interactions offers exciting opportunities for advancing our understanding of complex systems [8].

References

[1] V. I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer, 1957.

[2] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[3] H. M. Elshehabey, A. M. Aly, S. W. Lee, and A. B. Çolak. Integrating artificial intelligence with numerical simulations of cattaneo-christov heat flux on thermosolutal convection of nano-enhanced phase change materials. *Journal of Energy Storage*, 2024.

[4] A. Goyal, G. Gai, Z. Cheng, J. P. Cunha, and L. Zhu. Flow past a random array of statistically homogeneously distributed stationary platonic polyhedrons: Data analysis, probability maps and deep learning. *International Journal of Heat and Fluid Flow*, 2024.

[5] X. Liu, Y. Song, D. Zhao, K. Lan, K. Zhai, and M. Wang. Study on velocity profile of gas–liquid two-phase stratified flow in pipelines based on transfer component analysis-back propagation neural network. *Physics of Fluids*, 2024.

[6] Yang Liu and Anio Zhang. Kolmogorov-arnold networks for complex systems. *Journal of Computational Physics*, 423:109845, 2024.

[7] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[8] M. E. J. Newman. Complex networks: Structure and dynamics. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[9] SelfExplainML. Kans can't deal with noise. https://github.com/SelfExplainML/PiML-Toolbox/blob/main/docs/Workshop/KANs_Can't_Deal_with_Noise.ipynb, 2024. Accessed: 2024-08-07.

[10] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[11] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30(1):79–82, 2005.

[12] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.