Math 199 Final Report

Qingyang (Jimmy) Hou, Anika Misra
{jimmyhou, anikamisra}@g.ucla.edu

June 8th, 2025

1 Introduction

Modeling fluid dynamics and spatiotemporal physical systems is challenging due to nonlinear interactions. The Block Causal Transformer (BCAT) model [6] addresses this by predicting future simulation frames from a short past sequence—capturing spatial patterns and temporal evolution via frame-level autoregression, block-causal masking, and spatio-temporal coupling—and achieves roughly 2.9× better accuracy, 185× faster inference, and state-of-the-art results on multiple PDE benchmarks. In deep learning, optimizers [8] minimize the training loss.

In this project, we investigate how the BCAT model learns by evaluating different optimizers and examining its internal components. In the first part, we compare two alternatives—Adan and Muon—to the default AdamW, with the goal of identifying which optimizer provides the best training stability and accuracy. In the second part, we apply Muon selectively to the feed-forward networks and multi-head attention layers to determine which architectural block benefits most from orthogonalized updates. Finally, we delve into the attention layer itself by comparing Muon's impact on the query–key versus value–output projections to pinpoint which subcomponent gains the greatest performance improvements.

In general, our goal is to extend our findings to general PDE foundation models and beyond.

2 Background

2.1 Optimizers

2.1.1 AdamW

AdamW is built off of Adam, which is a first-order gradient descent algorithm that works well with both sparse gradients and non-stationary data [5]. Adam applies L_2 weight decay by combining the penalty with each adaptive update [5], but this coupling fails to enforce true weight decay, resulting in under-regularization and poorer generalization than SGD with momentum [7]. To address this issue, AdamW decouples weight decay from Adam's adaptive updates by applying it as a separate step, enforcing uniform shrinkage and restoring effective regularization [7]. This can be seen in AdamW's update rule (Equation 1), where θ_t represents the vector of parameter weights at iteration t, η_t is a set scalar, α is the learning rate, m_t is the first moment vector, v_t is the second moment vector, λ_t is the weight decay regularization factor, and the decoupling can be seen in the highlighted term [7]. Thus, AdamW can potentially boost generalization in long-term fluid predictions.

$$\theta_t = \theta_{t-1} - \eta_t \left(\frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_{t-1} \right) \tag{1}$$

2.1.2 Adan

Adan is an optimizer that uses Nesterov acceleration for gradient descent. AdamW and other more traditional optimizers generally rely on a heavy-ball acceleration technique to find a local minimum, but Nesterov acceleration converges faster. Hence, Adan's theoretical guarantees are better than those of AdamW, specifically for vision tasks, which will be useful for the BCAT model [11]. It is also more computationally efficient [11]. Adan's update rule can be seen in Equation 2, where θ_t represents the vector of parameter weights at iteration t, λ_{t-1} is the weight-decay regularization factor, η_{t-1} is the learning rate, and $\bar{m}_{t-1}$ is the Nesterov-accelerated gradient estimate at θ_{t-1} [11]. The specific version of Adan that we implemented is a D-Adaptation approach, which automatically adjusts the learning rate η and selects the value that yields the best performance [1]. Overall, we decided to test Adan because we expected better results than AdamW for the BCAT model, and hence it was worth comparing to Muon.

$$\theta_t = \frac{1}{1 + \lambda_{t-1}\eta_{t-1}} \left[\theta_{t-1} - \eta_{t-1} \circ (\bar{m}_{t-1})\right] \tag{2}$$

2.1.3 Muon

Muon is to take the usual SGD-momentum update for each 2D weight matrix and then orthogonalize it via Newton–Schulz iteration before applying it to the parameters [4]. Muon's orthogonalized momentum updates prevent collapse onto a few dominant directions, enabling more uniform exploration of the parameter space at negligible additional computational cost—an advantage that could help BCAT capture intricate spatio-temporal modes in fluid dynamics [4]. Furthermore, Muon delivers significant empirical speedups on benchmarks, which would be especially beneficial when training BCAT on extremely large datasets [4]. Muon's update rule can be seen in Equation 3, where θ_t represents the vector of parameter weights at iteration t, η is the learning rate, and $\mathcal{O}_t$ is the Newton-Shultz iteration applied to a variant of the current gradient [4].

$$\theta_t = \theta_{t-1} - \eta\mathcal{O}_t \tag{3}$$

2.2 Transformer Architecture and Attention Scores

Transformers involve two main components: multi-head attention layers and feed-forward networks (FFNs). The multi-head attention layers relate different components of the sequence to each other, and the FFNs contain linear transformations with a ReLU activation function [10]. The multi-head attention layers contain four main components: query, key, value, and output vectors [10]. Queries and keys are vectors relate to the input sequence, while values are used to help construct the output, as seen below [2].

In the multi-head attention layers, attention scores are computed for each head. To compute an attention score, a dot product of the queries, represented by a matrix Q, with all the keys, represented by a matrix K, is computed and scaled by their dimension, d_k. Then, the softmax of this result is taken and multiplied by the values, represented by a matrix V, seen in Equation 4 [10]. The attention score reveals how relevant a specific token is to the sequence [10].

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{4}$$

In multi-head attention models, each attention layer works in parallel to each other. More specifically, for the i^{th} head, $\text{head}_i = \text{Attention}(QW_i^Q,\ KW_i^K,\ VW_i^V)$ is computed, where QW_i^Q, KW_i^K, and VW_i^V are the projection matrices of weights in the for Q, K, and V, respectively. Finally, these attention outputs are concatenated to produce the final output result, as seen in Equation 5 [10].

$$\text{Output} = \text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{5}$$

Because queries and keys are more similar to each other, and values and outputs are more similar to each other [2], when analyzing which component of the attention layer benefits most from a specific setting, it is interesting to apply the setting to both the query and the key matrices together, or, the value and the output matrices together, rather than separating the four components.

3 Methods

3.1 Exploration 1: Finding the best optimizer

3.1.1 Standardized Parameters

In this project, we trained the BCAT model with three different optimizers using the following standardized parameters:

- Batch size: 1
- Input and output patches: 8

- Steps per epoch: 1000
- Number of epochs: 20
- Dataset: Shallow Water [9]

3.1.2 AdamW

To run AdamW, we used the settings learning rate 1×10^{-4}, weight decay 1×10^{-4}, $\beta_2 = 0.95$, $\epsilon = 10^{-6}$, and a `cosine` scheduler with 10% warm-up. The results of AdamW appear in Tables 1 and 2. Notably, the relative L_2 error was lower for 5-step predictions (0.00539) than for 1-step (0.00575), and peaked at 0.00606 for 10-step predictions. Figure 1 visualizes, at step 19 and $t = 10.00\,\text{s}$, that the difference between target and output remains within ± 0.06.

Table 1: AdamW performance

Metric	Data Loss	Rel. L_2	Step 1	Step 5	Step 10	Interior
Value	0.002096	0.00606	0.00575	0.00539	0.00606	0.00584

Table 2: AdamW relative L_2 error statistics

Statistic	Mean	Std	Min	Max	Median
Value	0.00606	0.00492	0.00270	0.02921	0.00482

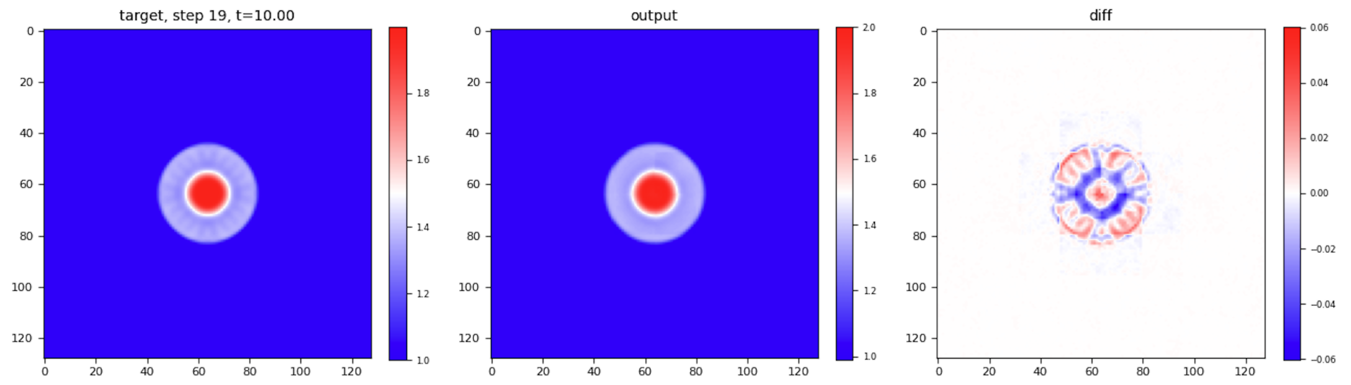

Figure 1: AdamW results visualization at step 19, $t = 10.00\,\text{s}$.

3.1.3 Adan

To run Adan, we set the learning-rate scheduler to cosine and the weight decay parameter to 10^{-4}—matching AdamW—and used the default learning rate of 1 as required by the D-Adaptation documentation [1]. The results of Adan appear in Tables 3 and 4. Notably, the relative L_2 error was higher when predicting 1 step ahead (0.00415) than when predicting 5 steps ahead (0.00395), and peaked at 0.00498 for 10-step predictions. Figure 2 visualizes, at epoch 20 and $t = 10.00\,\text{s}$, that the difference between the target and output ranges from -0.1 to 0.1.

Table 3: Adan performance

Metric	Data Loss	Rel. L_2	Step 1	Step 5	Step 10	Interior
Value	0.001031	0.00428	0.00417	0.00387	0.00428	0.00395

Table 4: Adan relative L_2 error statistics

Statistic	Mean	Std	Min	Max	Median
Value	0.00428	0.00354	0.00136	0.02023	0.00345

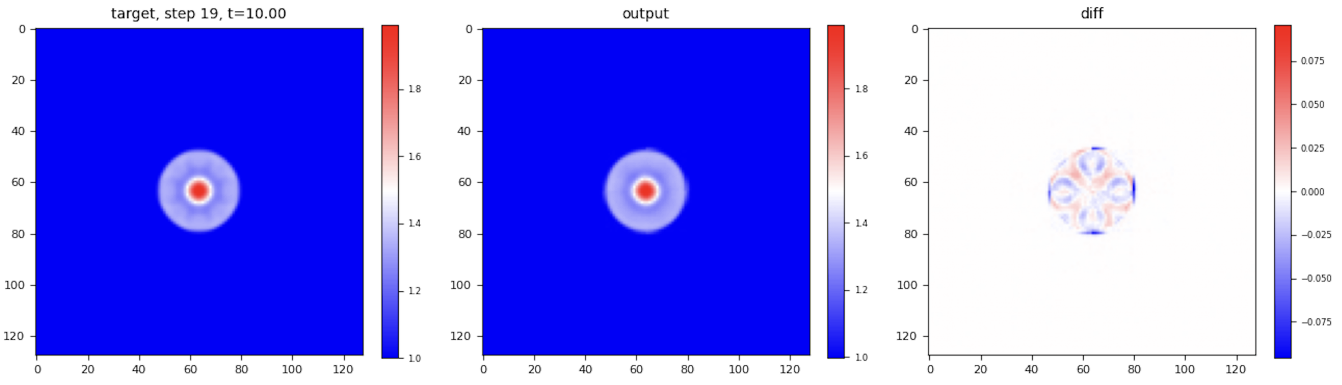

Figure 2: Adan results visualization at epoch 20, $t = 10.00\,\mathrm{s}$.

3.1.4 Muon

To run Muon, we used the settings learning rate 1×10^{-3}, weight decay 1×10^{-4}, $\beta_2 = 0.95$, $\epsilon = 10^{-6}$, and a `warmup_stable_decay` scheduler with 5% warm-up and 50% decay [4]. Muon applies its orthogonalization updates to all 2D weight matrices while routing 1D and embedding parameters through AdamW with betas $(0.9, 0.95)$ and $\epsilon = 10^{-6}$. The results of Muon appear in Tables 5 and 6. Notably, the relative L_2 error was higher for 1-step predictions (0.00298) than for 5-step (0.00277), with the 10-step error (0.00282) closely matching the 1-step level. Figure 3 visualizes the target, output, and difference at step 19, $t = 10.00\,\mathrm{s}$, with errors bounded in $[-0.02, 0.02]$.

Table 5: Muon performance

Metric	Data Loss	Rel. L_2	Step 1	Step 5	Step 10	Interior
Value	0.000386	0.00282	0.00298	0.00277	0.00282	0.00263

Table 6: Muon relative L_2 error statistics

Statistic	Mean	Std	Min	Max	Median
Value	0.00282	0.00133	0.00131	0.00794	0.00252

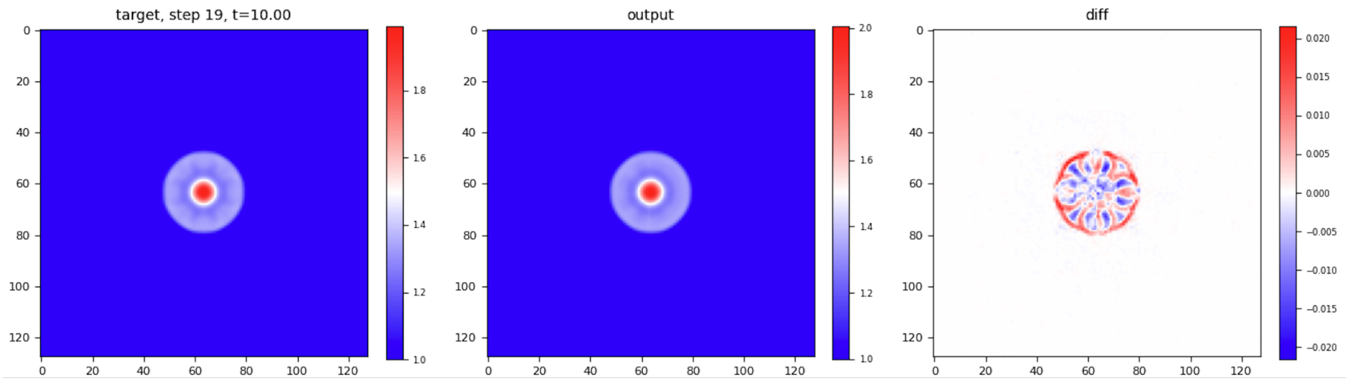

Figure 3: Muon results visualization at epoch 20, $t = 10.00\,\mathrm{s}$.

3.1.5 Results

Muon outperforms both Adan and AdamW across every metric: it achieves lower data loss, lower relative L_2 errors at 1-, 5-, and 10-step forecasts and interior predictions, and a more compact error distribution (mean, std, min, max, median). Notably, Muon's minimum error (0.00131) closely matches Adan's (0.00136) but its maximum error (0.00794) is far smaller than Adan's (0.02023) and AdamW's (0.02921). See Tables 3, 4, 1, 2, 5, and 6. Muon uses a Newton–Schulz step to keep its weight updates balanced and varied, making training more stable and faster so it achieves the lowest forecast errors [4]. Additionally, Adan's results were generally better than AdamW's because Adan improves on AdamW by adding Nesterov momentum and comparing gradient changes [11]—as discussed earlier with Equations 1 and 2—which cuts bias and speeds up learning, giving more stable updates and lower BCAT errors.

3.2 Exploration 2: How Muon Optimizes the Model

3.2.1 Experimental Setup

With the previous exploration showing that Muon showed the best results, we decided to continue the second exploration with this optimizer. Specifically, we decided to compare Muon's optimizing effect on the FFN and attention layers to see which one benefited more. To isolate where Muon's orthogonalized momentum updates have the greatest impact, we ablated its application in three configurations:

- **FFN-only:** Muon updates on all feed-forward (FFN) weight matrices; all attention weights use AdamW.
- **Attention-only:** Muon updates on all self-attention weight matrices (queries, keys, values, projections); all FFN weights use AdamW.
- **Both:** Muon updates on both the FFN and self-attention weight matrices.

Unlike Exploration 1, here we use a reduced-size BCAT—12 layers with $d_{\text{emb}} = 512$, $d_{\text{ffn}} = 2048$, and 8 attention heads—bringing the parameter count down from 162 million to approximately 52 million. We train on the more challenging 2D Navier–Stokes dataset [3] for 20 epochs with batch size 1 and 1000 steps per epoch to amplify any optimizer-induced differences. Any parameter group for which Muon is disabled falls back to AdamW, following the prescription in [4].

3.2.2 Results

Table 7: Evaluation metrics (epoch 19) for different Muon configurations

Config.	**Data Loss**	**Rel. L2**	**Step 1**	**Interior**
Muon on both FFN & Attn	0.001228	0.03256	0.02348	0.03220
Muon on FFN only	0.002939	0.05166	0.03655	0.05077
Muon on Attention only	0.002097	0.04311	0.03152	0.04256

Table 8: Relative L2 error statistics for different Muon configurations

Config.	**Mean**	**Std**	**Min**	**Max**	**Median**
Muon on both FFN & Attn	0.03256	0.00982	0.01567	0.10594	0.03072
Muon on FFN only	0.05166	0.01296	0.02650	0.14072	0.04970
Muon on Attention only	0.04311	0.01161	0.02160	0.12627	0.04106

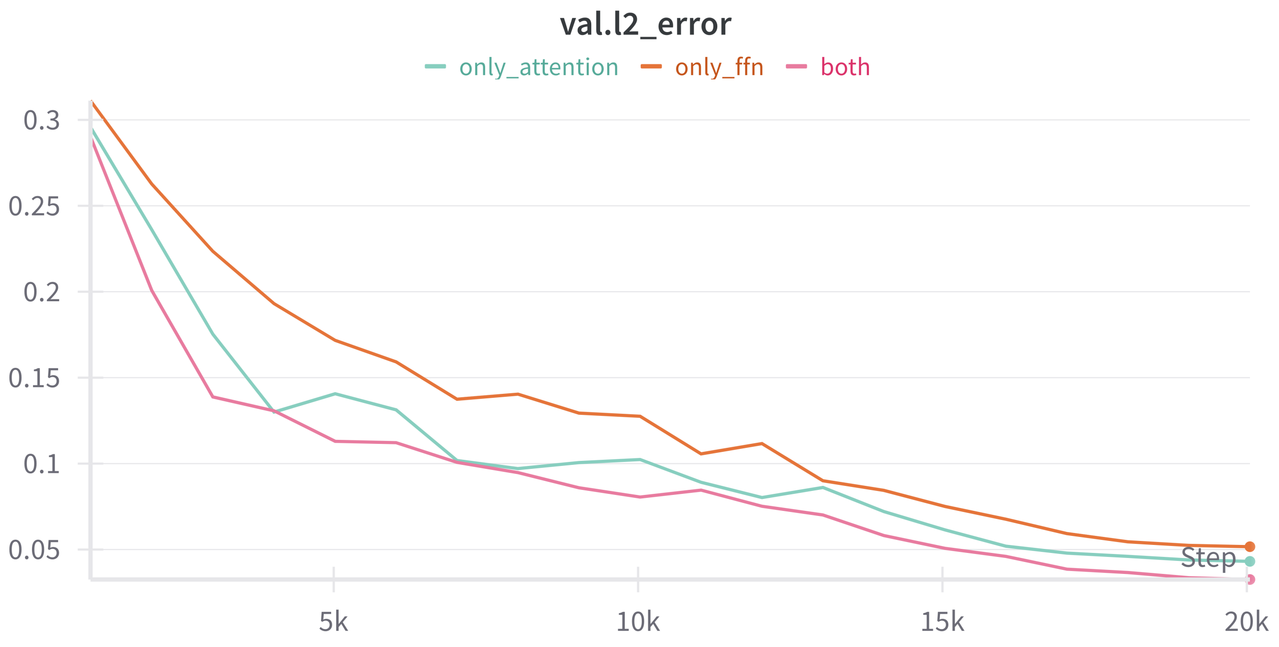

Figure 4: Validation ℓ_2 error over training steps at $t = 10.00\,\text{s}$.

Tables 7 and 8, together with Figure 4, show that:

- **FFN-only:** A modest acceleration over AdamW, but the curve plateaus at a substantially higher final loss.
- **Attention-only:** A much steeper initial descent and a significantly lower final loss, indicating a stronger benefit when Muon is applied to self-attention sublayers.
- **Both:** The fastest overall convergence and lowest final training loss, suggesting additive advantages from optimizing both FFN and attention parameters jointly.

These results demonstrate that while Muon improves convergence across the network, its most pronounced effect is on the attention mechanism, with additional—but smaller—gains from also applying it to the FFN. This makes sense, as the attention mechanism utilizes complex matrices of learned parameter weights, whereas the FFN is a simpler generally linear structure [10] that may not benefit as much from a more advanced optimizer.

3.3 Exploration 3: How Muon Optimizes Multi-Head Attention

For our final analysis, we wanted to learn how exactly the attention layer benefits most from the Muon optimizer. As mentioned in Section 2, queries and keys are similar to each other, both relating to the input, and values and outputs are more similar to each other, both relating to the output [2]. Hence, we group the query and key components (QK) together and the value and output components (VO) together when conducting our experiment. For this exploration, we first applied Muon to QK only and used AdamW for all other components, and then we applied Muon to VO only and used AdamW for all the other components. We used AdamW as it was the standard optimizer that was used in the original BCAT model [6]. In both analyses, we continued with the same parameters from the previous exploration in Section 3:

- Batch size: 1
- Input and output patches: 8
- Steps per epoch: 1000
- Number of epochs: 20
- Number of layers: 12

- Embedding Dimension: 512
- FFN Dimension: 2048
- Number of heads: 8
- Dataset: Navier Stokes 2D Conditioned

In transformer architectures, the core computation in each self-attention head is the scaled dot-product between queries and keys, $QK^\top/\sqrt{d_k}$, which determines how strongly each position attends to every other position [10, 2]. Because Muon's orthogonalized momentum updates are designed to preserve the geometry of weight matrices and prevent collapse onto a small number of directions [4], we hypothesized that applying Muon to the query and key projection matrices (QK) would most improve the conditioning of the attention score computation.

3.3.1 Results

After training the model applying Muon to only QK and then only to VO, we find the following results in Tables 9 and 10.

Table 9: Evaluation metrics (epoch 19) for Muon ablations on self-attention projections

Config.	Data Loss	Rel. L2	Step 1	Interior
Muon on QK only	0.003516	0.05658	0.04131	0.05580
Muon on VO only	0.002171	0.04403	0.03290	0.04351

Table 10: Relative L_2 error statistics for Muon ablations on self-attention projections

Config.	Mean	Std	Min	Max	Median
Muon on QK only	0.05658	0.01351	0.02837	0.13939	0.05451
Muon on VO only	0.04403	0.01136	0.02260	0.13019	0.04182

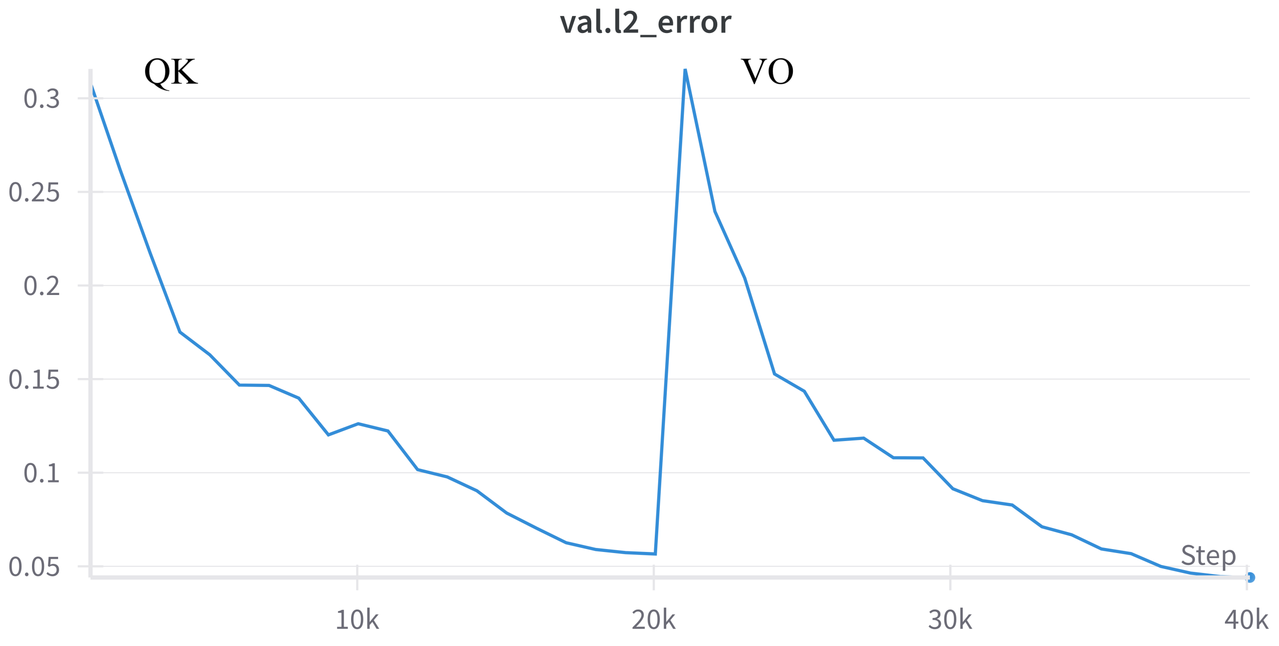

Figure 5: Validation ℓ_2 error over training steps at $t = 10.00\,\text{s}$. The left segment (steps 0–20 k) corresponds to the run with Muon applied only to the QK projections, while the right segment (steps 20 k–40 k) shows the continuation where Muon is applied only to the VO projections.

Contrary to our initial hypothesis, the VO ablation—applying Muon to the value and output projection matrices—yielded a larger reduction in final relative ℓ_2 error (Tables 9 and 10). This potentially suggests that while optimizing Q and K helps stabilize attention scores, optimizing the V and O projections more directly enhances the model's ability to represent and reconstruct the attended features. Nonetheless, the outcomes can also be influenced by factors such as the size of the parameter groups receiving Muon updates and variability arising from different random seeds across runs. Further investigation is detailed in the "Future Studies" section.

4 Conclusions and Discussion

We employed the BCAT model to investigate how transformer architectures in PDE modeling can benefit from advanced optimizers. First, we compared AdamW, Adan, and Muon on a simpler dataset and found Muon—due to its Newton–Schulz–based orthogonalization—delivered the lowest forecast errors. Next, we applied Muon selectively to the BCAT's feed-forward and self-attention blocks, demonstrating that the attention mechanism derived the greatest convergence gains, likely due to its richer inter-token interactions. Finally, within the attention layer we compared Muon's impact on query–key versus value–output projections and observed a larger error reduction for the VO projections; however, we do not attribute this improvement solely to Muon—other confounding factors may be at play, and further investigation is needed.

4.1 Future Studies

We have already begun implementing attention-score distribution analysis to quantify how Muon affects sparsity, entropy, and diversity of attention across both heads and layers. We will extend this methodology to compare other optimizers—Adan and AdamW—to determine whether similar shifts in attention behavior correlate with their performance. Gaining insight into these inner mechanics is crucial, since modeling fluid dynamics and spatiotemporal physical systems remains a major challenge [6], and understanding exactly how BCAT and related PDE models allocate attention can guide the design of more effective transformer-based PDE foundation models in the future, and might also improve transformer models in fields like large language models and beyond.

References

[1] Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation, 2023. URL: https://arxiv.org/abs/2301.07733, https://arxiv.org/abs/2301.07733 arXiv:2301.07733.

[2] Josep Ferrer. How transformers work: A detailed exploration of transformer architecture, 2024. URL: https://www.datacamp.com/tutorial/how-transformers-work.

[3] Jayesh K. Gupta Johannes Brandstetter. Navierstokes-2d-conditioned, 2023. URL: https://huggingface.co/datasets/pdearena/NavierStokes-2D-conditoned.

[4] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL: https://kellerjordan.github.io/posts/muon/.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL: https://arxiv.org/abs/1412.6980, https://arxiv.org/abs/1412.6980 arXiv:1412.6980.

[6] Yuxuan Liu, Jingmin Sun, and Hayden Schaeffer. Bcat: A block causal transformer for pde foundation models for fluid dynamics, 2025. URL: https://arxiv.org/abs/2501.18972, https://arxiv.org/abs/2501.18972 arXiv:2501.18972.

[7] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL: https://arxiv.org/abs/1711.05101, https://arxiv.org/abs/1711.05101 arXiv:1711.05101.

[8] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A survey of optimization methods from a machine learning perspective, 2019. URL: https://arxiv.org/abs/1906.06821, https://arxiv.org/abs/1906.06821 arXiv:1906.06821.

[9] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench Datasets, 2022. https://doi.org/10.18419/DARUS-2986 doi:10.18419/DARUS-2986.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL: https://arxiv.org/abs/1706.03762, https://arxiv.org/abs/1706.03762 arXiv:1706.03762.

[11] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9508–9520, 2024. https://doi.org/10.1109/TPAMI.2024.3423382 doi:10.1109/TPAMI.2024.3423382.