# Towards Learning High-Precision Least Squares Algorithms with Sequence Models

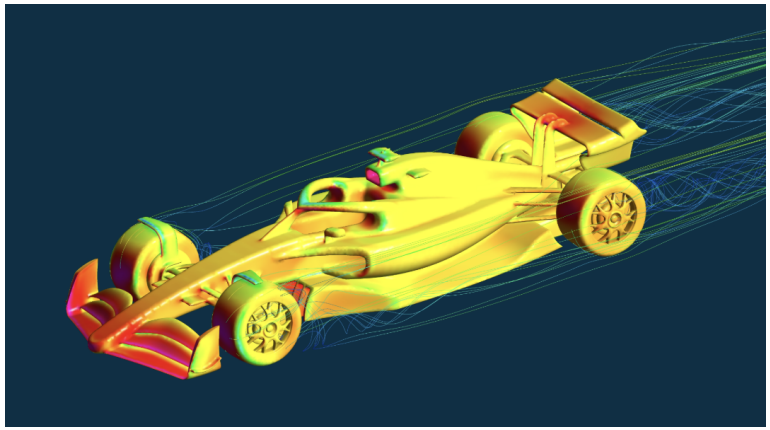
*Jerry Liu[1], *Jessica Grogan[1], Owen Dugan[1], Ashish Rao[1], Simran Arora[1], Atri Rudra[2], Chris Ré[1]

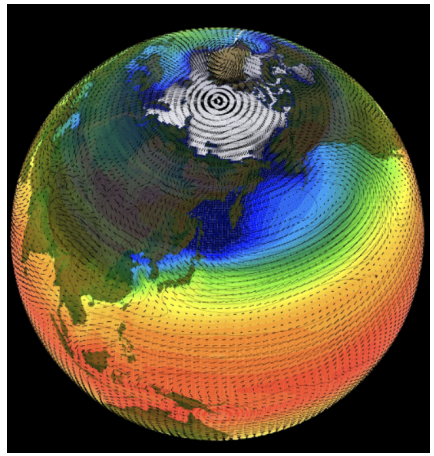
[1]Stanford University [2]University at Buffalo

**ICLR 2025**

# Towards foundation models for science/engineering

- Consistent, surprisingly simple recipe. Dump raw data into a large model $\rightarrow$ emergent capabilities.
- Unique desiderata in science: **numerical precision**
  - Nonlinear problems
  - Long time horizon forecasting





Are our current training recipes (architectures/optimizers) sufficient?

# Roadmap

- **Background.** Transformers, least squares, and in-context learning as gradient descent.
- **Observing precision limitations.** Transformers fail to solve least squares precisely.
- **Identifying precision bottlenecks.** We identify issues with architecture and optimization.
- **Addressing the bottlenecks.** We investigate alternative architectures and propose an improved training recipe.

# Least squares as a testbed

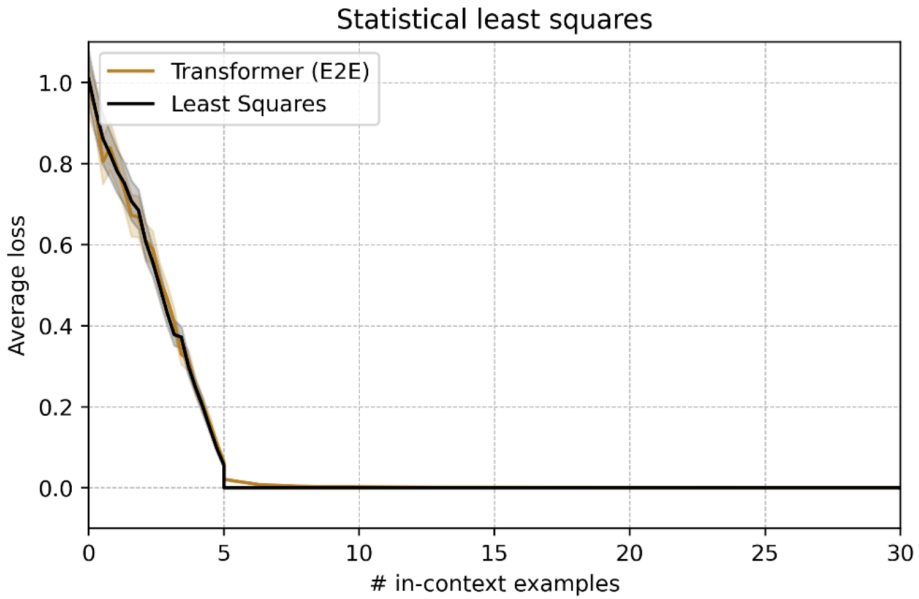
We investigate the ability of Transformers to solve least squares problems in-context.

$$\begin{bmatrix} \leftarrow & \mathbf{a}_1^T & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{a}_N^T & \rightarrow \end{bmatrix} \mathbf{x} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$$

Why least squares?

- Simple and well-understood with theoretical guarantees.
- Useful primitive for differential equations: spectral and pseudo-spectral methods.

# Transformers can solve least squares in-context



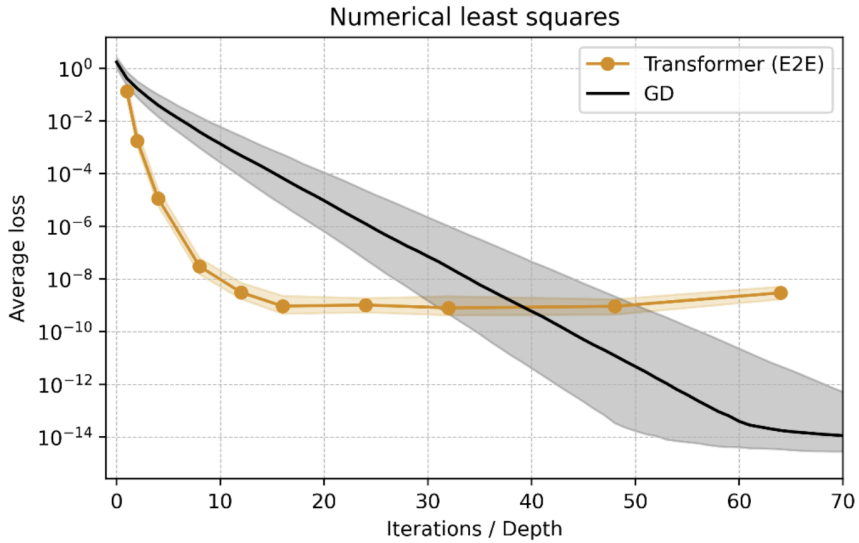

$D = 5$. Transformers approximate Bayes-optimal estimators.

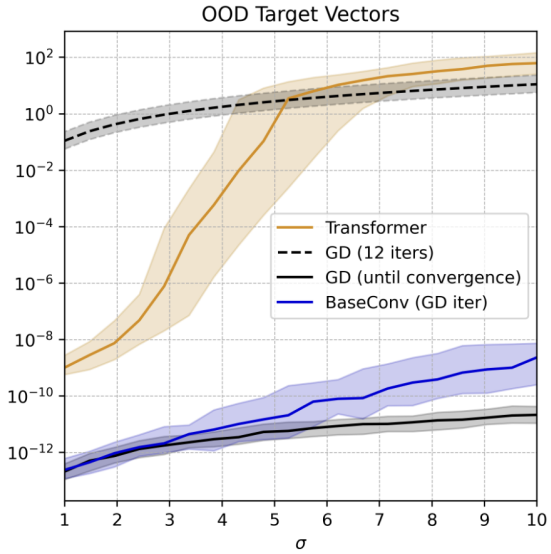
Hypothesis from the **in-context learning** literature*: each layer implements a step of **gradient descent** on the objective

$$\mathcal{L}(x) = \frac{1}{2} \sum_{i=1}^{N} \left( a_i^T x - b_i \right)^2$$

*Transformers Learn In-Context by Gradient Descent. von Oswald et al., ICML 2023.

# Transformers can solve least squares in-context?




$$\mathbf{A}\mathbf{x} = \sigma\mathbf{b}$$

However, we find that Transformers fail to learn a proper numerical algorithm for least squares:

- **Poor precision** which saturates with deeper networks.
- **Brittle generalization** on out-of-distribution targets.

Q: Can we identify sources of low precision (architecture/optimizer)?

# Architecture-based precision bottlenecks

We look at basic algorithms for solving least squares, **gradient descent** (GD) and **Newton's method**, and identify three *linear algebra primitives* that comprise them.

For input $\mathbf{u} \in \mathbb{R}^{N \times D}$:

$$\textsc{Read}(i, j, a, b)(\mathbf{u}) = \begin{cases} \mathbf{u}[k, a{:}b] & k \neq j \\ \mathbf{u}[i, a{:}b] & k = j \end{cases},$$

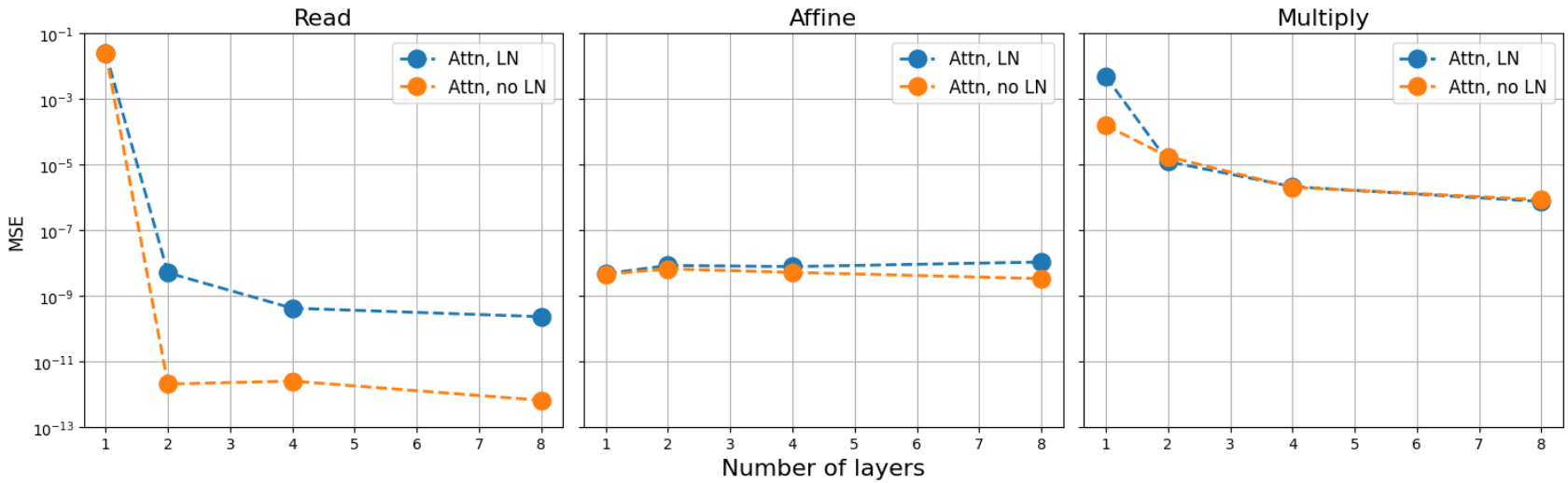
$$\textsc{Linear}(\mathbf{H})(\mathbf{u}) = \mathbf{u}\mathbf{H}, \quad \text{where } \mathbf{H} : \mathbb{R}^{D} \to \mathbb{R}^{d_{out}} \text{ is linear,}$$

$$\textsc{Multiply}(a, b, d_{out})(\mathbf{u}) = \mathbf{u}[:, a{:}a{+}d_{out}] \odot \mathbf{u}[:, b{:}b{+}d_{out}]$$

- $\textsc{Read}$ transfers information across the sequence dimension
- $\textsc{Linear}$ transfers information across the hidden dimension
- $\textsc{Multiply}$ computes high-degree interaction terms (e.g. dot products, element-wise squaring)

# Architecture-based precision bottlenecks

We find that attention-based models struggle to precisely implement these **linear algebra primitives**.

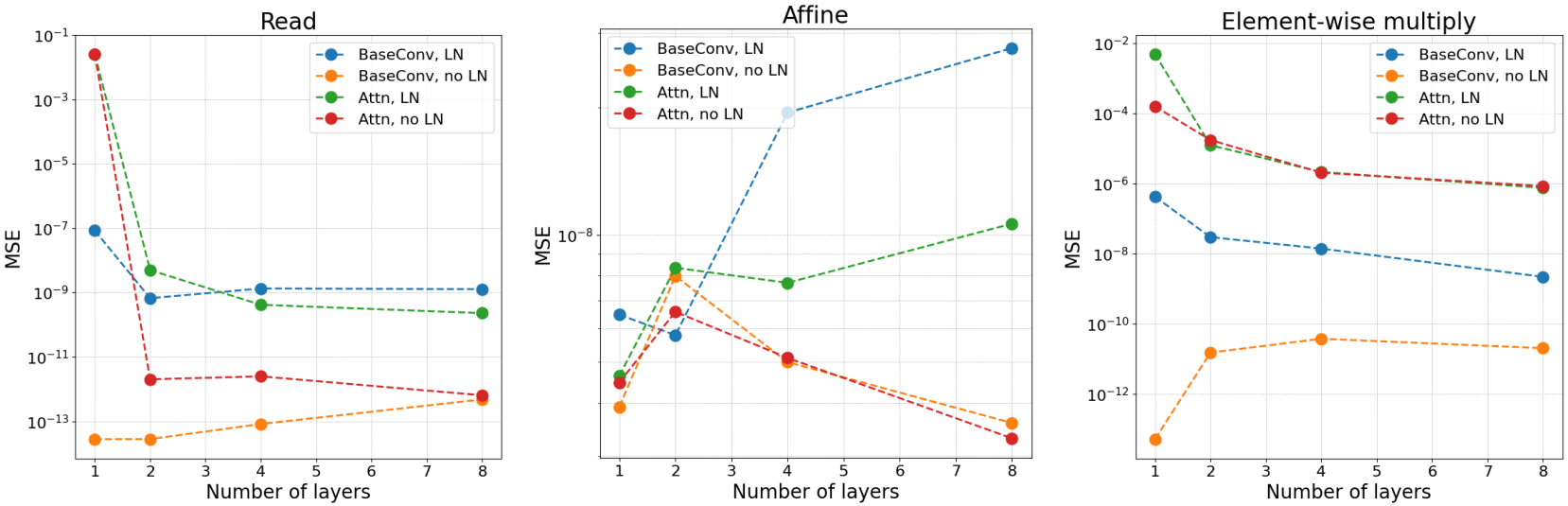



Attention struggles with floating-point multiplication!

# Polynomial-based architectures for high precision

Given an input $\mathbf{u} \in^{N \times D}$, $\textsc{BaseConv}^*(\mathbf{u})$ is defined as:

$$\underbrace{(\mathbf{u}\mathbf{W}_{gate} + \mathbf{b}_{gate})}_{\textbf{Linear Projection}} \odot \underbrace{(\mathbf{h} * (\mathbf{u}\mathbf{W}_{in} + \mathbf{b}_{in}) + \mathbf{b}_{conv})}_{\textbf{Convolution}} \mathbf{W}_{out} + \mathbf{b}_{out}$$




$\textsc{BaseConv}$ learns to precisely implement primitives.

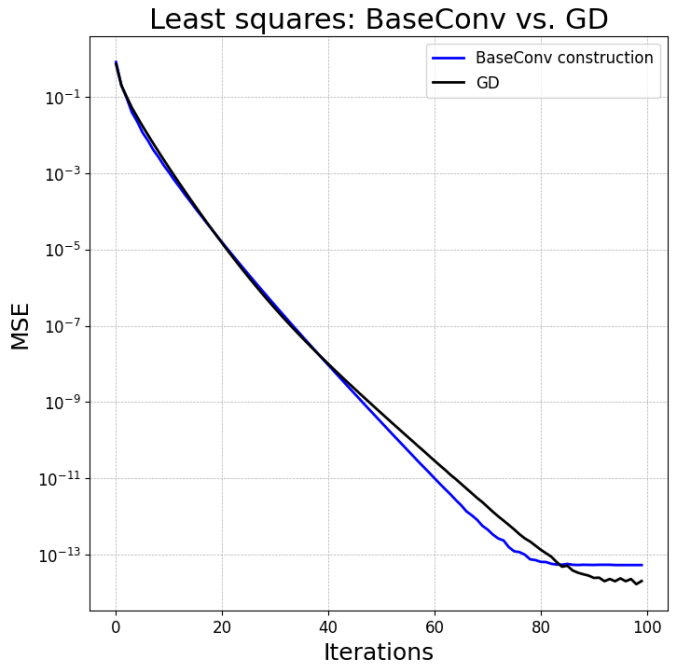
*Zoology: Measuring and Improving Recall in Efficient Language Models. Arora et al., ICLR 2024.

# Gated convolutions can express precise algorithms...

Given a least squares problem instance $\boldsymbol{Ax} = \boldsymbol{b}$ and an initial iterate $\boldsymbol{x}_0$, a single iteration of gradient descent computes

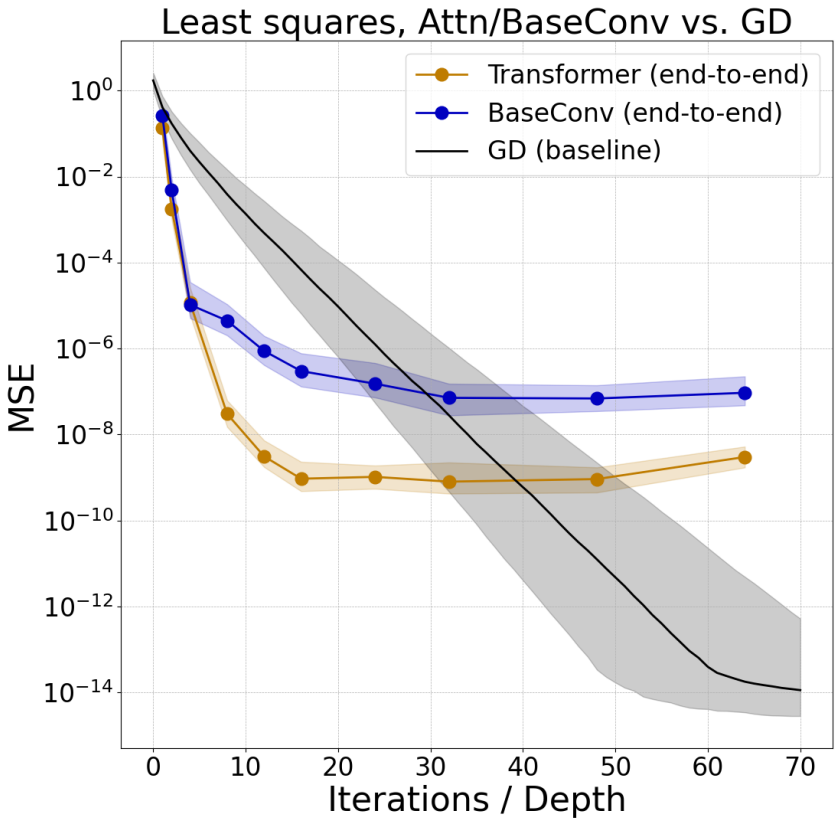
$$\boldsymbol{x}_1 := \boldsymbol{x}_0 - \eta \nabla \mathcal{L}(\boldsymbol{x}_0), \quad \text{where } \nabla_{\boldsymbol{x}} \mathcal{L} = \boldsymbol{A}^T(\boldsymbol{Ax} - \boldsymbol{b}).$$

We implement a weight construction with BaseConv for GD and show empirically that gated convolutions can express a **high-precision algorithm**.



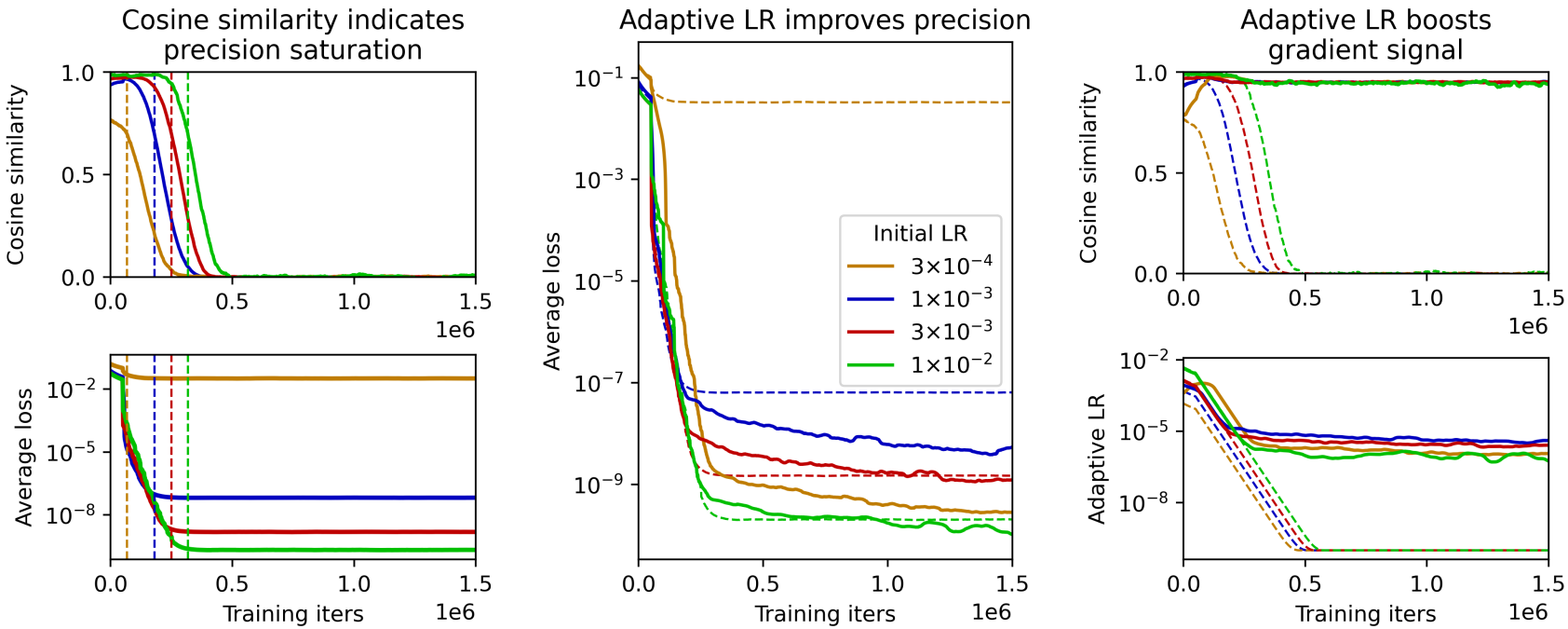
# ...but it's hard to train end-to-end precisely

Simply swapping out softmax attention for BaseConv and training end-to-end is insufficient – **optimization challenges**!




We look at a simpler task: **one step of gradient descent**.

# Adaptive learning rate scheduler for high precision




Gradient metric is predictive of precision saturation (left). We propose a simple adaptive LR scheduler that alleviates precision saturation (middle). Adaptive LR effectively boosts gradient signal during training (right).

# Application: in-context differential equations

## Parameterized ODEs

We consider a simple distribution of 1D linear ordinary differential equations (ODEs) defined as follows:

$$\mathcal{L}_\alpha(c, u_0) = u \,\Big|\, \begin{cases} u'(t) = \alpha_1 c(t) + \alpha_2 u(t) + \alpha_3 \\ u(0) = u_0 \end{cases} \quad \text{on } \Omega = [-1, 1]$$

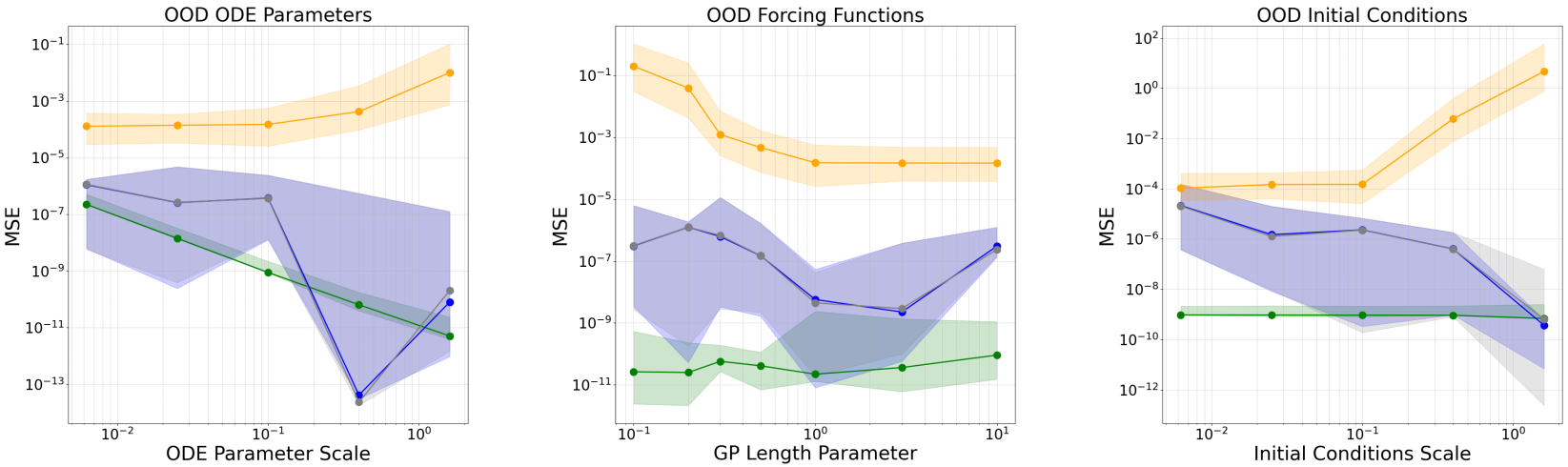
where:

- $\alpha$: Parameters influencing the behavior of the solution.
- $c(t)$: Forcing function dependent on time.
- $u_0$: Initial condition of the solution $u$.

Problem setup is **in-context operator learning**: we're given forcing functions and corresponding solutions, and we need to infer $\alpha$ to make predictions for new forcing functions.

# Application: in-context differential equations

Applied iteratively during inference, BaseConv trained on iterations of gradient descent learns a **precise and general numerical algorithm**.

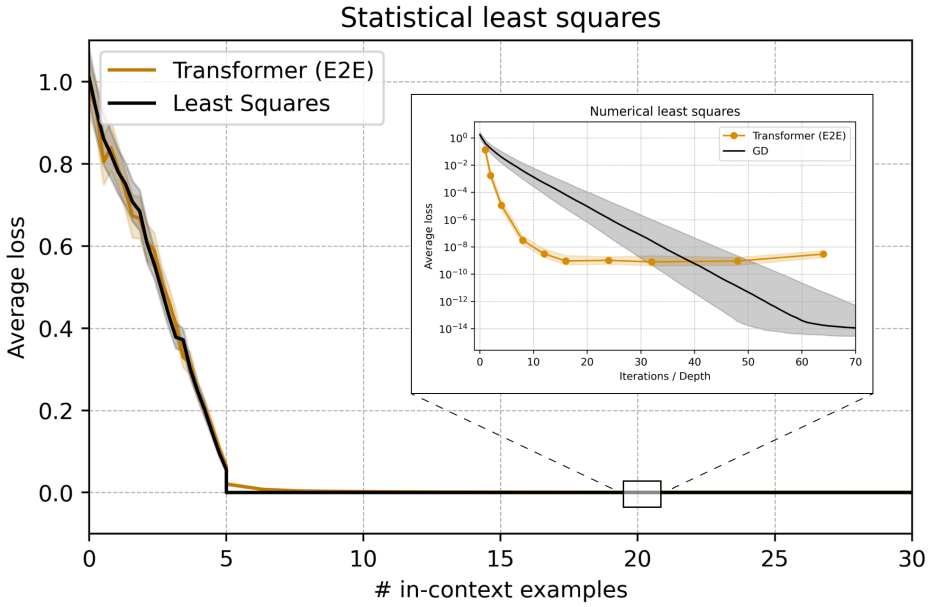


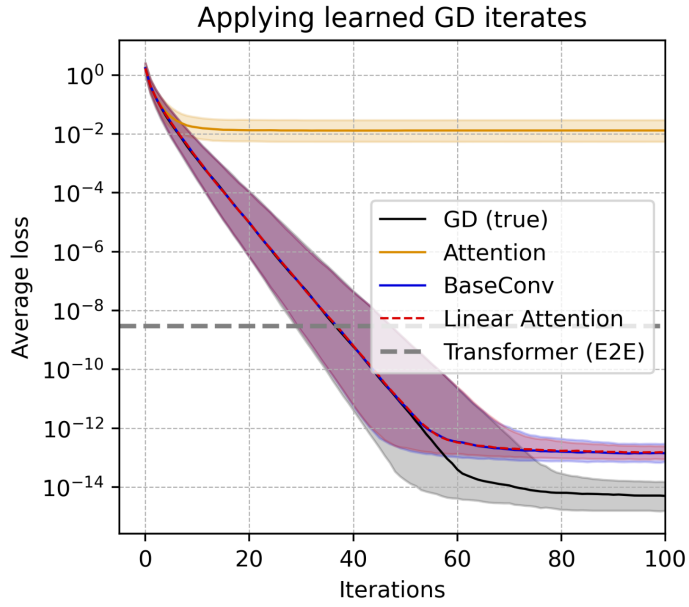

Out-of-distribution error comparison between Transformer (orange), BaseConv (blue), gradient descent (gray), and least squares (green). We evaluate out-of-distribution ODE parameters (left), forcing function smoothness (middle), and initial condition distribution (right).

# Conclusion

- Current ML training recipes are built for classification tasks (e.g. language modeling). We observe fundamental shortcomings when applying them to even basic regression tasks.
  - **Architecture:** softmax attention struggles to implement high-precision floating-point multiplication.
  - **Optimizer:** precision with Adam and standard learning rate schedulers saturates.
- We investigate alternatives and show they fix the precision bottlenecks:
  - **Architecture:** polynomial architectures (BaseConv, linear attention) to represent general arithmetic circuits.
  - **Optimizer:** adaptive learning rate scheduler using cosine similarity metric over gradients.

# Thank you!




Jerry Liu*, Jessica Grogan*, Owen Dugan, Ashish Rao, Simran Arora, Atri Rudra, Chris Ré. *Towards Learning High-Precision Least Squares Algorithms with Sequence Models.* ICLR 2025.

**ArXiv:** `https://arxiv.org/abs/2503.12295`

**Code:** `https://github.com/HazyResearch/precision-ls`