Linear Transformers are Versatile In-Context Learners

Google Research
Duke University

Max Vladymyrov mxv@google.com
Johannes von Oswald jvoswald@google.com
Mark Sandler sandler@google.com
Rong Ge rongge@cs.duke.edu

Neural Information Processing Systems

Key Findings

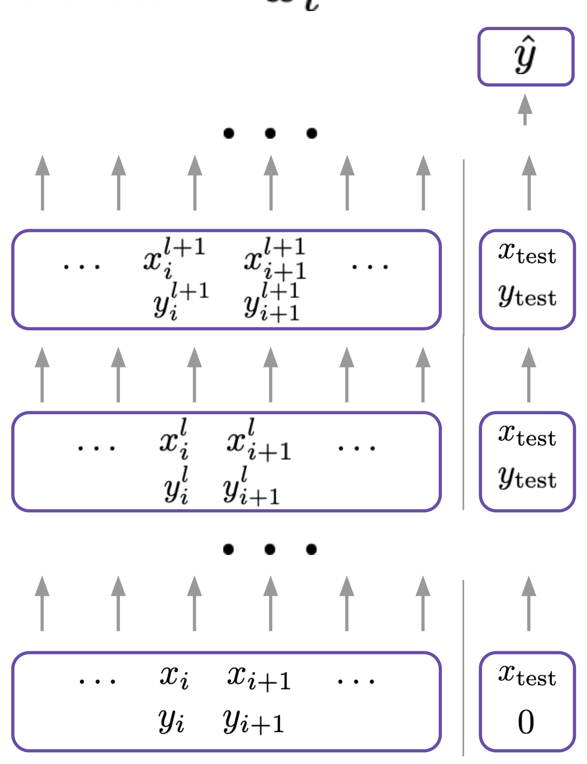
- 🧩 Each layer of a linear transformer acts like a step in a complex optimization algorithm, similar to gradient descent.
- 🧠 Linear transformers can learn to solve challenging problems, like linear regression with varying levels of noise.
- 🚀 They discover effective optimization strategies that outperform standard methods.
- 🔍 These strategies include adjusting step sizes based on noise levels and rescaling the solution.

Model: *Linear Transformer*

- Linear Transformer updates each layer using

$$\begin{pmatrix} x_i^{l+1} \\ y_i^{l+1} \end{pmatrix} := \sum_{k=1}^{h} \left[P_k^l \sum_{j=1}^{n} \left(\begin{pmatrix} x_j^l \\ y_j^l \end{pmatrix} ((x_j^l)^\top, y_j^l) \right) Q_k^l \right]$$

- Each token $e_i = (x_i, y_i) \in \mathbb{R}^{d+1}$ consists of a feature vector $x_i \in \mathbb{R}^d$, and its corresponding output $y_i \in \mathbb{R}$.
- We append a query token $e_{n+1} = (x_t, 0)$ to the sequence, where x_t represents test data.
- The goal of in-context learning is to predict y_t for the test data x_t.



Problem: *Noisy regression*

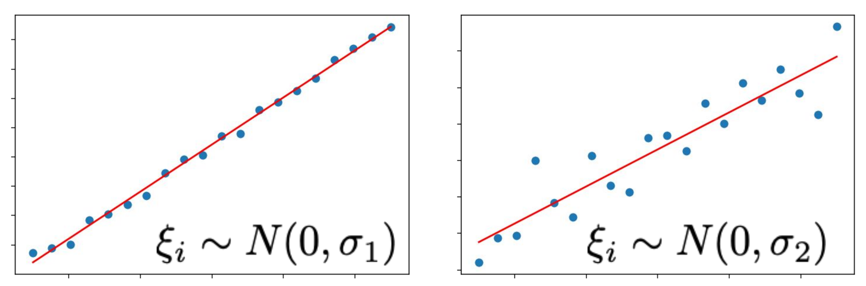
For each input sequence τ the input is given by:

- A ground-truth weight vector $w_\tau \sim N(0, I)$.
- n input data points $x_i \sim N(0, I)$.
- Noise $\xi_i \sim N(0, \sigma_\tau^2)$ sampled with variance $\sigma_\tau \sim p(\sigma_\tau)$.
- Labels $y_i = \langle w_\tau, x_i \rangle + \xi_i$.

For a *known* noise level σ_τ, the best estimator for w_τ is provided by ridge regression:

$$L_{\mathrm{RR}}(w) = \sum_{i=1}^{n} (y_i - \langle w, x_i \rangle)^2 + \sigma_\tau^2 \|w\|^2,$$

We also consider problems where the noise variance σ_τ is sampled from a given distribution $p(\sigma_\tau)$.



$$y_i = \langle w_\tau, x_i \rangle + \xi_i$$

$$L(\theta) = \mathop{\mathbb{E}}_{\substack{w_\tau \sim N(0,I) \\ x_i \sim N(0,I) \\ \xi_i \sim N(0,\sigma_\tau^2)}} \left[(\hat{y}_\theta(\{e_1, \dots, e_n\}, e_{n+1}) - y_t)^2 \right],$$

Main Result: Linear transformers maintain linear regression model at every layer.

Linear transformers are restricted to maintaining a linear regression model based on the input:

Theorem 4.1. *Suppose the output of a linear transformer at l-th layer is $(x_1^l, y_1^l), (x_2^l, y_2^l), \dots, (x_n^l, y_n^l), (x_t^l, y_t^l)$, then there exists matrices M^l, vectors u^l, w^l and scalars a^l such that*

$$x_i^{l+1} = M^l x_i + y_i u^l, \qquad x_t^{l+1} = M^l x_t,$$
$$y_i^{l+1} = a^l y_i - \langle w^l, x_i \rangle, \qquad y_t^{l+1} = -\langle w^l, x_t \rangle.$$

Diagonal attention matrices

We also analysed even simpler variant of linear transformer with diagonal attention matrices. Since the elements x are permutation invariant, a diagonal parameterization reduces each attention heads to just four parameters:

$$P_k^l = \begin{pmatrix} p_{x,k}^l I & 0 \\ 0 & p_{y,k}^l \end{pmatrix}; \quad Q_k^l = \begin{pmatrix} q_{x,k}^l I & 0 \\ 0 & q_{y,k}^l \end{pmatrix}.$$

Using reparametrization

$$w_{xx}^l = \sum_{k=1}^H p_{x,k}^l q_{x,k}^l, \quad w_{xy}^l = \sum_{k=1}^H p_{x,k}^l q_{y,k}^l,$$
$$w_{yx}^l = \sum_{k=1}^H p_{y,k}^l q_{x,k}^l, \quad w_{yy}^l = \sum_{k=1}^H p_{y,k}^l q_{y,k}^l.$$

leads to the following diagonal layer updates:

$$x_i^{l+1} = x_i^l + w_{xx}^l \Sigma^l x_i^l + w_{xy}^l y_i^l \alpha^l$$
$$y_i^{l+1} = y_i^l + w_{yx}^l \langle \alpha^l, x_i^l \rangle + w_{yy}^l y_i^l \lambda^l,$$

Each term controls a specific aspect of the updates:

- w_{yx}^l: how much x_i^l influences y_i^{l+1}.
  - Controls the *gradient descent*.
- w_{xx}^l: how much x_i^l influences x_i^{l+1}.
  - Controls the *preconditioner* strength.
- w_{xy}^l: how much y_i^l influences x_i^{l+1}.
  - *Adapting the step-sizes* based on the noise.
- w_{yy}^l: how much y_i^l influences y_i^{l+1}.
  - *Adaptive rescaling* based on the noise.

Experiments

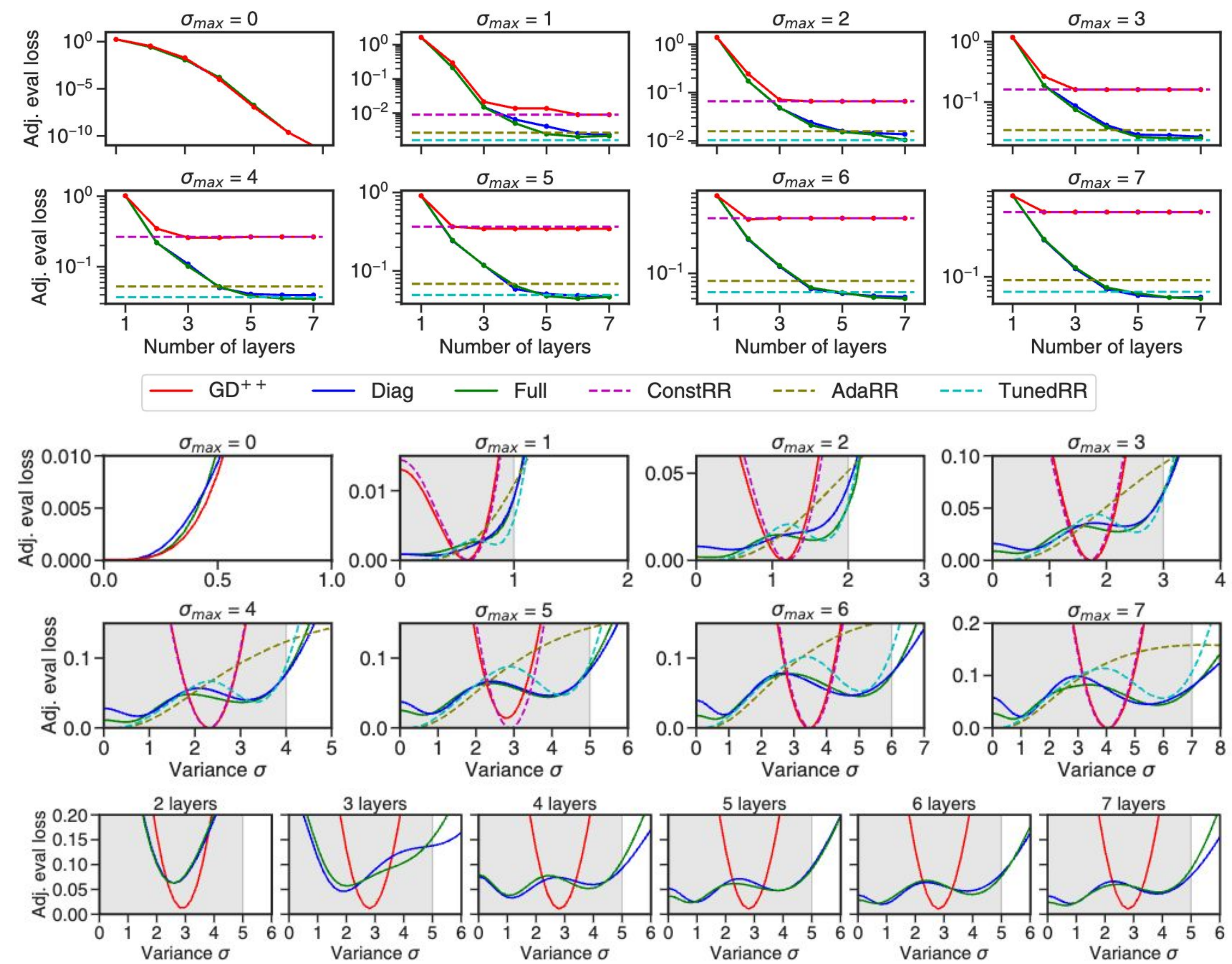
Linear Transformer-based methods:

- **Full**. Trains full parameter matrices.
- **Diag**. Trains diagonal parameter matrices
- **GD++**. An even more restricted diagonal variant that uses only w_{yx}^l and w_{xx}^l terms.

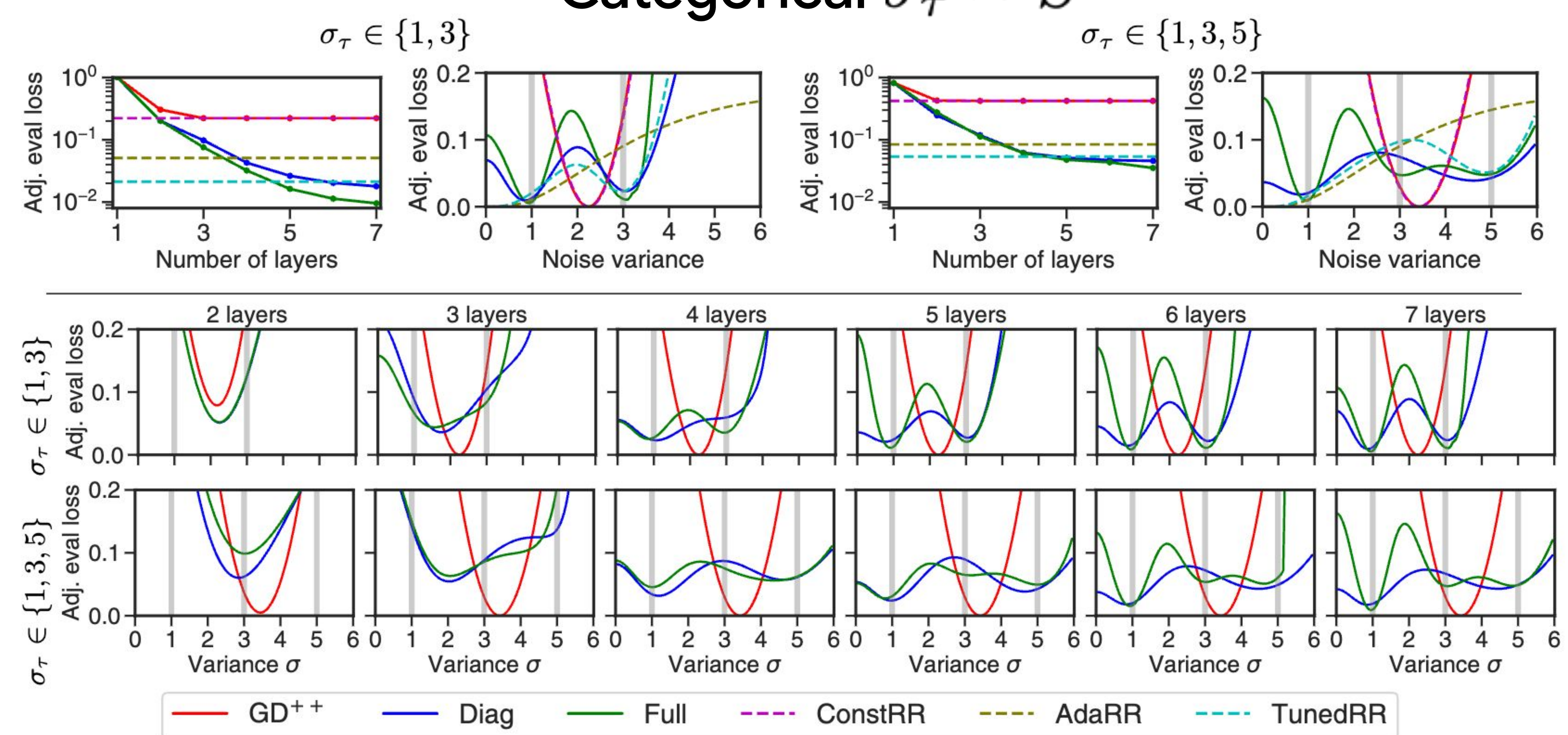
Baselines:

- **Constant Ridge Regression (ConstRR)**. The noise variance is estimated using a single scalar value for all the sequences.
- **Adaptive Ridge Regression (AdaRR)**. Estimate the noise variance via unbiased estimator: $\sigma_{\mathrm{est}}^2 = \frac{1}{n-d} \sum_{j=1}^n (y_j - \hat{y}_j)^2$, where $\hat{y}_j$ represents the solution to the ordinary least squares.
- **Tuned Adaptive Ridge Regression (TunedRR)**. Same as above, but after the noise is estimated, we tuned two parameters:
  - a max. threshold for the estimated variance,
  - a multiplicative adj. to the noise estimator.

Uniform $\sigma_\tau \sim U(0, \sigma_{max})$



Categorical $\sigma_\tau \sim S$



Conclusions

- 💡 Linear transformers, even though they are simple, can be a surprisingly versatile in-context learners.
- 🔥 They can discover effective optimization strategies that outperform standard methods.
- 🔮 Transformers have the potential to automatically discover new and effective algorithms for various machine learning tasks.