Efficient Linear System Solver with Transformers

Max Vladymyrov *mxv@google.com*
Johannes von Oswald *jvoswald@google.com*
Nolan Miller *namiller@google.com*
Mark Sandler *sandler@google.com*

Key Findings

- Linear Transformers can efficiently solve small positive definite symmetric linear systems.
- Effective reparametrization allows for solving problems with different lineary system sizes.
- Competitive performance with classical methods for small systems.

Problem formulation

Find vector $x \in \mathbb{R}^N$ that solves the system of *N* linear equations: $\langle a_i, x\rangle = b_i$

With $a_i \in \mathbb{R}^N$ and $b_i \in \mathbb{R}$

Training data: positive definite symmetric matrices *A* with a fixed condition number.

Linear Transformer

Linear Transformer updates each layer using

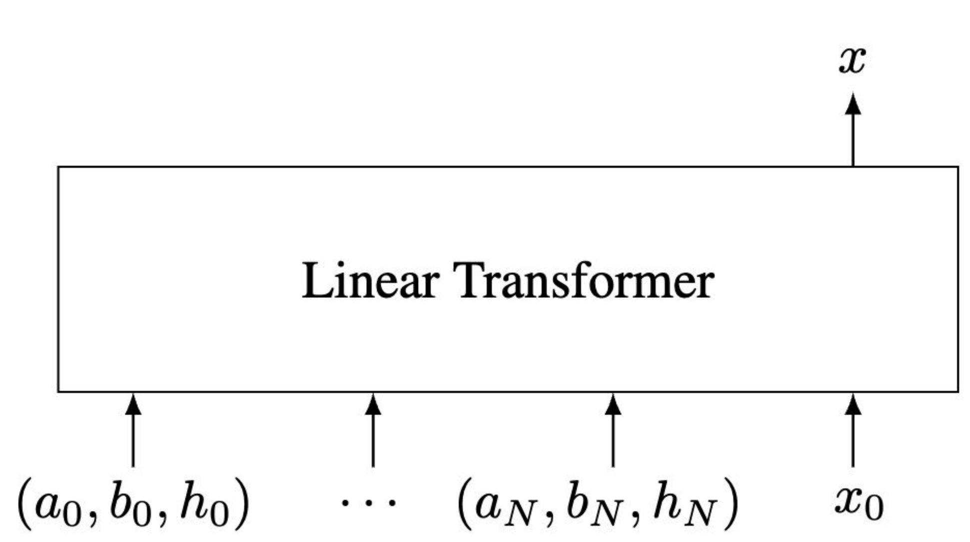
$$\Delta e_i = \sum_{j=1}^{N}(e_j^\top Q e_i) P e_j.$$

With weights $P = W_P W_V$ and $Q = W_K^\top W_Q$.

Objective function to minimize:

$$L(\theta) = \mathop{\mathbb{E}}_{A,b}\left[(f_\theta(\{e_1, ..., e_N\}, e_{N+1}) - x)^2\right].$$

Data Encoding

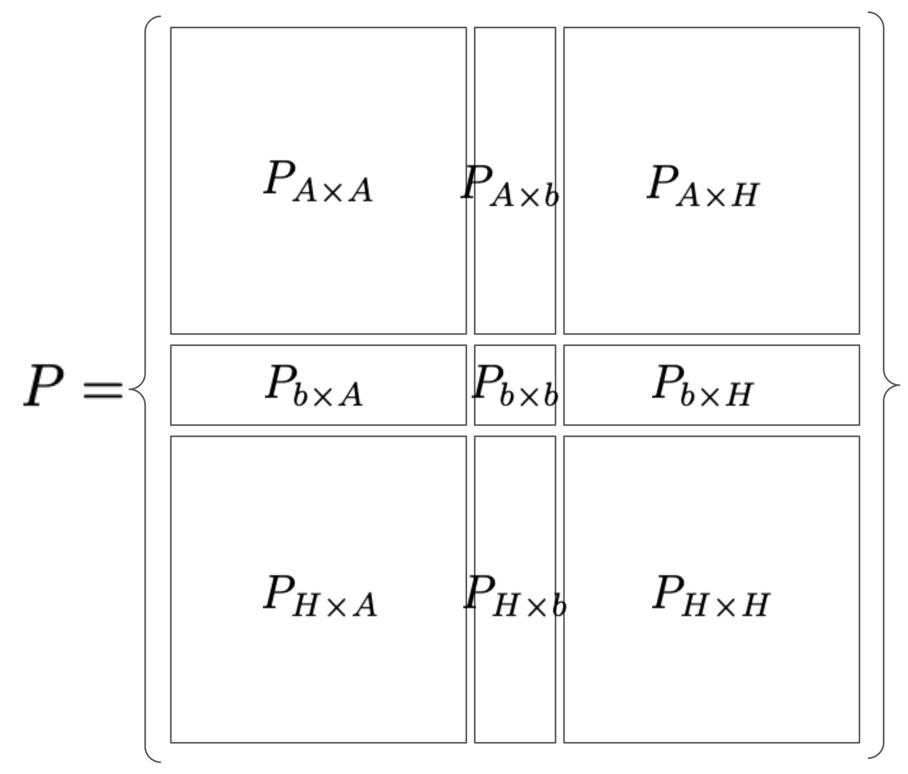


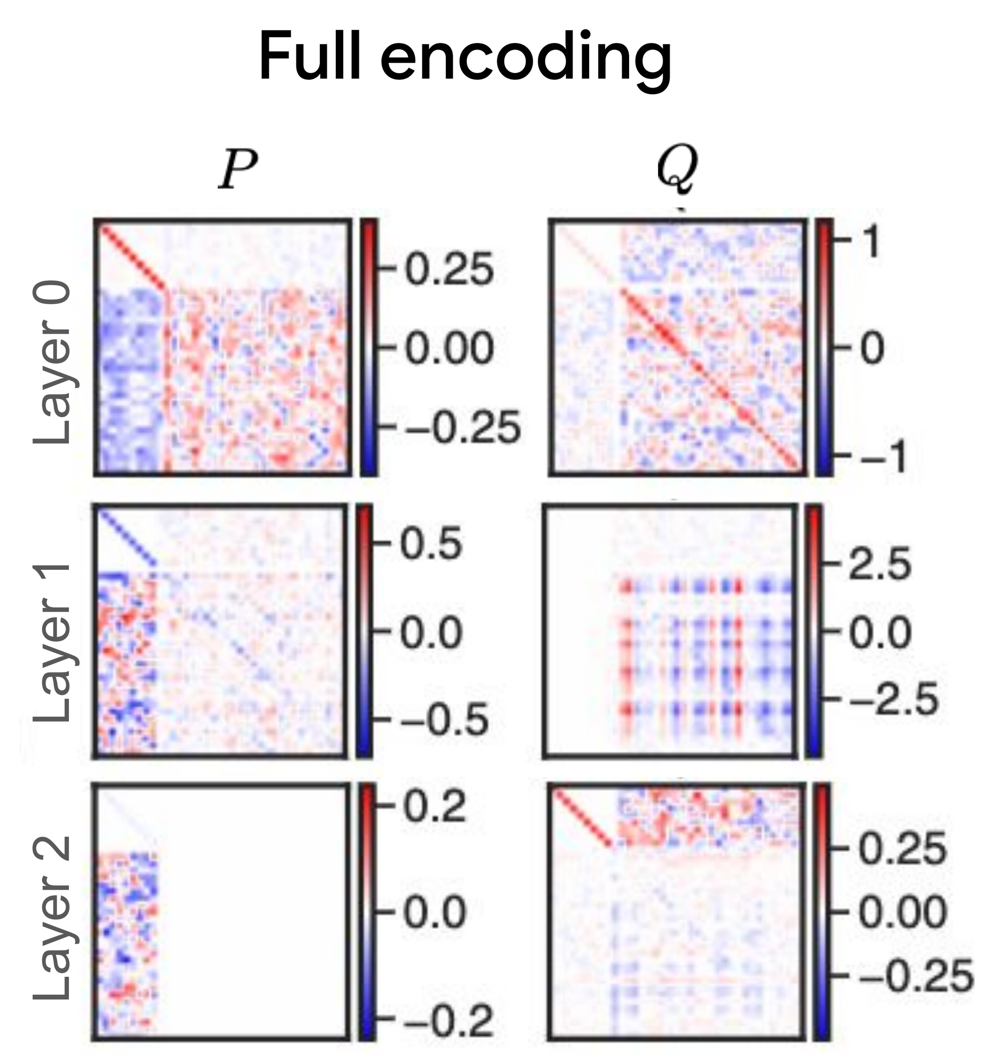
Each equation is encoded as a token $e_i = (a_i, b_i, h_i)$, where $H = (h_0, h_1, ...h_N)$ is an optional embedding matrix (either learned or predefined).

We append a query token $e_{N+1} = (x_0, 1_{1+K})$ to the sequence, where x_0 represents test data.

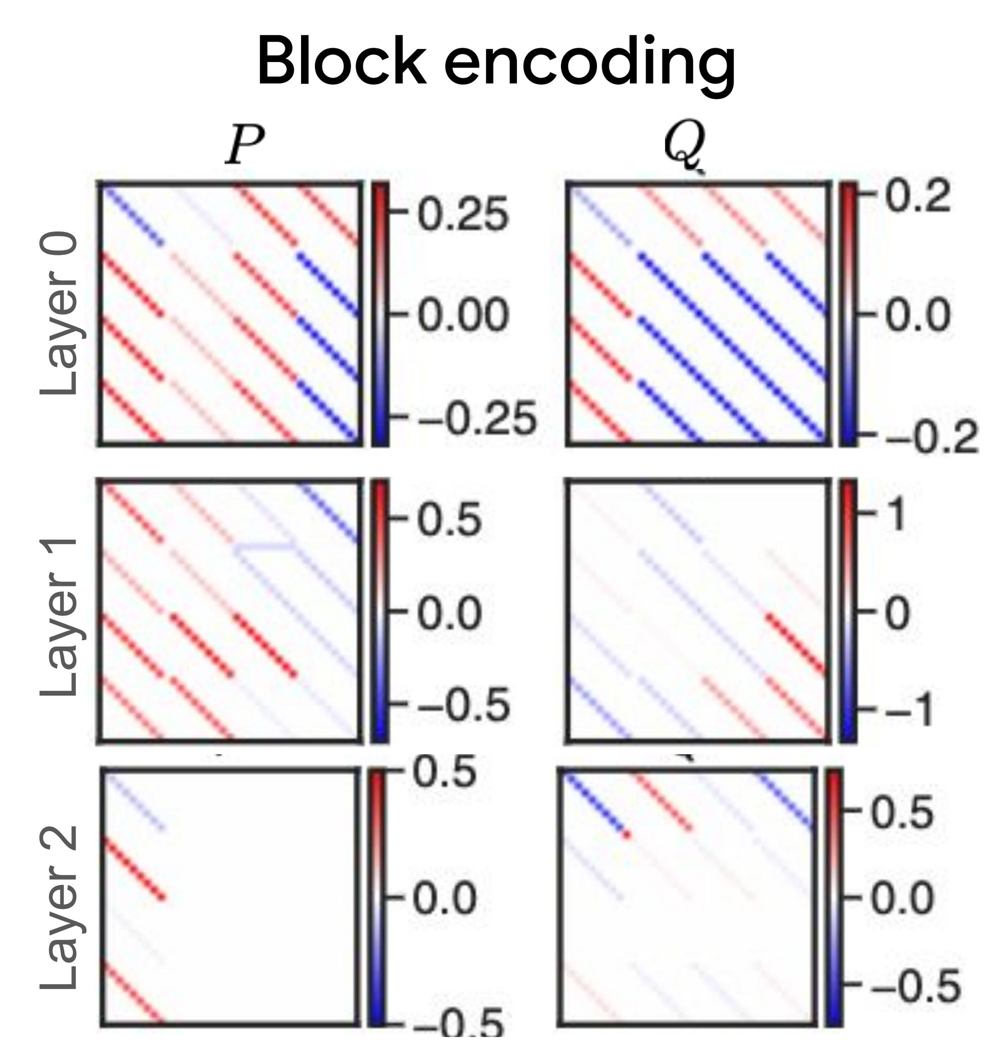
Re-parameterization of weight matrix

Consider the following block re-parametrization of weight matrix P (same for matrix Q):





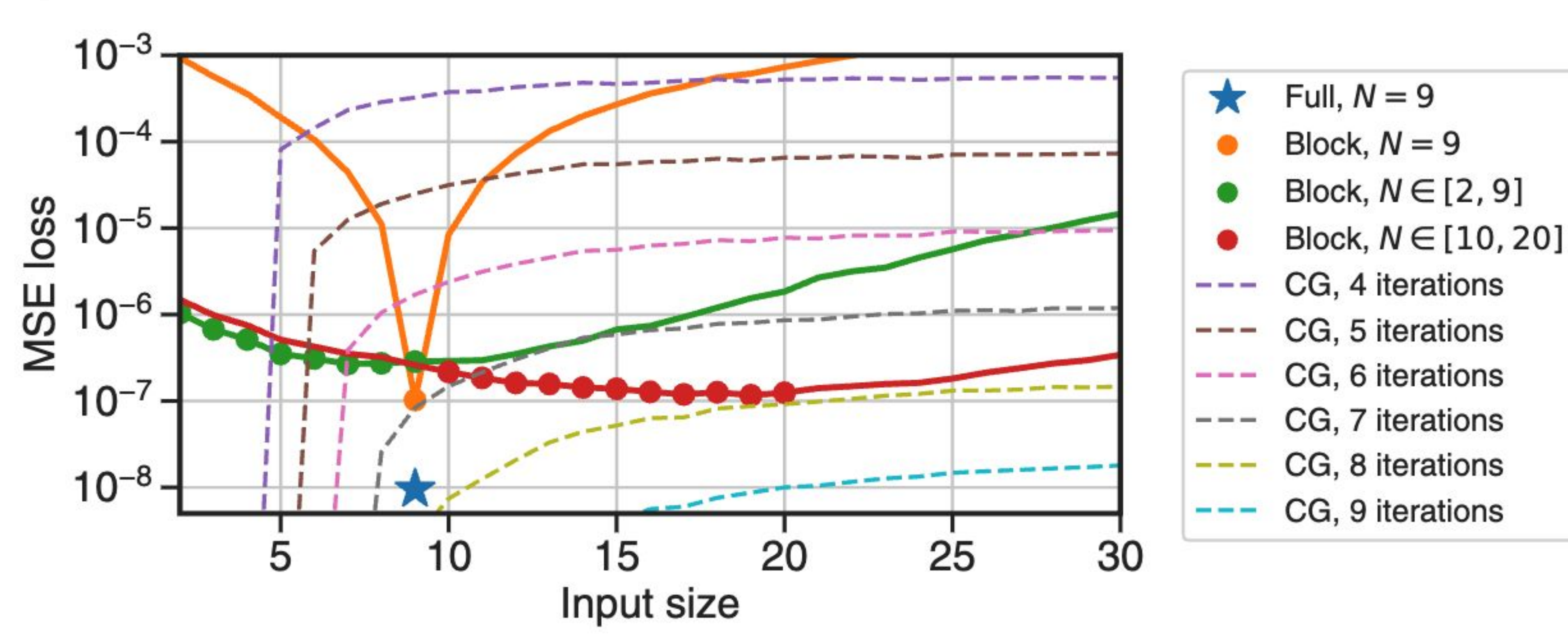
- Square matrices represented as scalar times identity, e.g. $P_{A\times A} = p_{A\times A} I$.
- Rectangular matrices represented as scalar times a vector of ones, e.g. $P_{A\times b} = p_{A\times b} 1$



Motivation:

- Elements of A are sampled independently, => there is no bias for any dimension.
- **Efficiency:** identity matrices speed up computation
- **Performance:** comparable loss to training with full matrices.
- **Generalization:** Decouples problem dimension (N) from model parameters, enabling application and fine-tuning to different input sizes.

Experiments



- **Full, *N=9***. Full encoding, trained on *N=9* problems only + 27-dim learned embedding *H*. This model cannot generalize to matrices of other sizes, but it achieves the best performance for problems of size *N=9*.
- **Block, *N=9***. Block encoding, trained on *N=9* problems only + three *NxN* fixed identity matrices H. The generalization quality is limited.
- **Block, *N* ∈ *[2,9]***. Block encoding, trained on sizes *N* ∈ *[2,9]* + three *NxN* fixed identity matrices H. Generalizes well beyond its training sizes.
- **Block, *N* ∈ *[10,20]***. Block encoding, fine-tuned from model *N* ∈ *[2,9]* above on sizes *N* ∈ *[10,20]*. Generalizes well beyond its training sizes.