Revisiting Transformer Layer Parameterization Through Causal Energy Minimization

Jin Xu[*] Microsoft

Camille Couturier Microsoft

Victor Rühle Microsoft

Saravan Rajmohan Microsoft

James Hensman[*] Microsoft Research Cambridge

Abstract

Transformer blocks typically combine multi-head attention (MHA) for token mixing with gated MLPs for token-wise feature transformation, yet many choices in their parameterization remain largely empirical. We introduce Causal Energy Minimization (CEM), a framework that recasts Transformer layers as optimization steps on conditional energy functions while explicitly accounting for layer parameterization. Extending prior energy-based interpretations of attention, CEM shows that weight-tied MHA can be derived as a gradient update on an interaction energy, and that a gated MLP with shared up/down projections can be viewed through an element-wise energy. This perspective identifies a design space for Transformer layers that includes within-layer weight sharing, diagonal-plus-low-rank interactions, lightweight preconditioners, and recursive updates. We evaluate CEM-derived layers in language-modeling experiments at the moderate hundred-million-parameter scale. Despite their constrained parameterizations, these layers train stably and can match corresponding Transformer baselines. Overall, our results suggest that CEM provides a useful lens for understanding Transformer layer parameterization, connecting Transformer architectures to energy-based models and motivating further exploration of energy-guided layer designs.

1 Introduction

Stacked sequence-to-sequence mappings underlie modern foundation models [Bommasani et al., 2021]. Early work employed recurrent [Cho et al., 2014, Hochreiter and Schmidhuber, 1997, Sutskever et al., 2014] and convolutional architectures [Gehring et al., 2017, Kalchbrenner et al., 2016], but these have been largely replaced by Transformer [Vaswani et al., 2017]. Despite alternatives such as structured state-space models [Gu and Dao, 2024, Gu et al., 2022], Transformer layers, particularly multi-head attentions (MHAs) and gated multilayer perceptrons (MLPs), remain the core building blocks of today's large language models (LLMs). Yet architectural innovations for Transformers continue to be driven mainly by empirical studies [Ainslie et al., 2023, Shazeer, 2019, 2020, Shazeer et al., 2017], motivating perspectives that connect these parameterizations to explicit computational principles.

Energy-based models (EBMs) provide one such perspective. By assigning a scalar energy $\mathcal{E}(\boldsymbol{x})$ to a configuration $\boldsymbol{x}$, EBMs interpret computation as the search for low-energy states [Ackley et al., 1985, Hopfield, 1982, Krotov and Hopfield, 2016, LeCun et al., 2006, Ramsauer et al., 2021]. This viewpoint connects layer computation to optimization, making it possible to study architecture and parameterization through energy functions, update rules, and optimization dynamics.

[*]Correspondence to jinxu2@microsoft.com and jameshensman@microsoft.com

Preprint.

We introduce Causal Energy Minimization (CEM), a framework that associates Transformer layers with conditional energy functions and interprets their residual updates as optimization steps. Concretely, to map an input sequence $\boldsymbol{h}_{1:J}$ to an output sequence $\boldsymbol{h}'_{1:J}$, Causal Energy Minimization (CEM) introduces, for each position i, an optimization variable $\boldsymbol{x}_i$ initialized at $\boldsymbol{h}_i$. The variable is updated by a procedure $\mathcal{A}$ using a conditional energy $\epsilon(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i})$ that depends on the causal history $\boldsymbol{h}_{1:i}$, and the resulting state is used as the output $\boldsymbol{h}'_i$. This formulation separates the roles of the energy and the update procedure, providing a unified way to analyze and design layer transformations.

Building on prior energy-based views of attention [Ramsauer et al., 2021, Sander et al., 2022], CEM focuses on the parameterization induced by the energy-gradient perspective. We show that weight-tied MHA arises as such a step on an interaction energy, with the key projection tied to the value projection and the query projection tied to the output projection, and that gated MLPs with shared up/down projections admit an analogous interpretation through element-wise energies (Sections 2.1 and 2.2). Our goal is not to replace Transformers with standalone energy-based sequence models, which have faced challenges on large-scale language modeling [Du et al., 2020, Qin et al., 2022], but to ask whether Transformer layers themselves can be revisited as energy-based updates and what this perspective implies for their parameterization and extension.

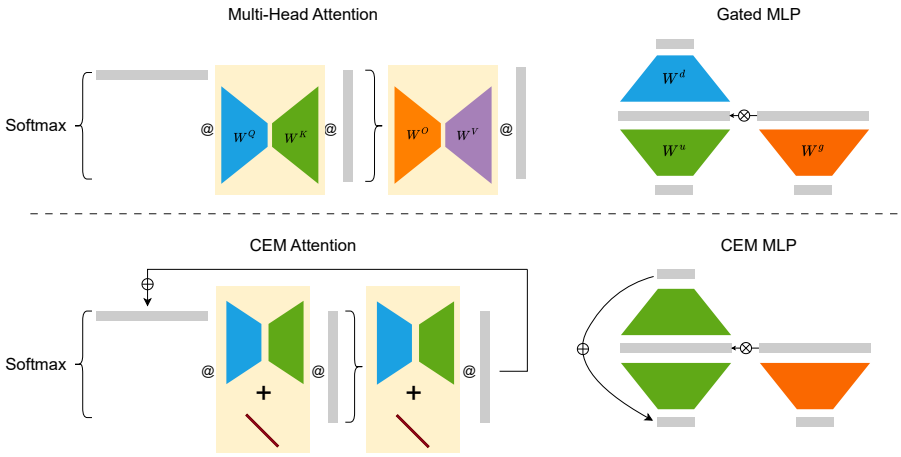

Figure 1: **Comparison of transformer layer parameterizations**. Top left: standard multi-head attention (per head). Top right: gated MLP. Bottom left: CEM-derived attention. Bottom right: CEM-derived MLP. Colors indicate shared weights within each subfigures (See Sections 2.1 and 2.2). Arrows highlight the recursive structure of CEM modules, which implement multiple gradient steps of energy minimization (Equations (16) and (17)), while brown bars denote the optional diagonal term added to the key–query projections (See details in Section 2.3). For the attention heads, W^V maps the hidden state to values, which are then projected back by W^O and scaled by the scalar Softmax weight.

Contributions. We use CEM to study Transformer layer parameterization, focusing on how energy functions and optimization updates can inform the design of layer variants. Our experiments evaluate these ideas at moderate scale and provide evidence that CEM-derived layers can match standard Transformer components while enabling new parameterization choices. In particular,

- We connect shared key/query and value/output projections in MHA, and shared up/down projections in MLP, to gradient updates on interaction and element-wise energies, respectively.
- We extend layer design beyond single gradient updates, investigating diagonal-plus-low-rank weights, preconditioned updates, and within-layer recursive steps.
- We show that CEM-derived layers match their Llama counterparts at moderate scale, with parameterization choices that improve performance without increasing parameter count.

2 Transformer layers as energy updates

We reframe Transformer layers through the lens of CEM. While prior work has explored energy-based views of attention [Ramsauer et al., 2021, Sander et al., 2022], we extend this perspective to MLPs and uncover a common weight-sharing structure across both components. We introduce two complementary energy terms: an *interaction term*, capturing dependencies across features of different tokens, and an *element-wise term*, assigning energy to each token's feature vector. Gradient-based updates on these energies naturally recover standard Transformer layers with weight sharing, shared key/value and query/output projections in MHA emerge from interaction-energy updates, while shared up/down projections in MLPs emerge from element-wise energy updates. Figure 1 illustrates the resulting scheme.

2.1 Gradient of interaction energy yields weight-tied attention

Multi-head attention (MHA). In conventional MHA, the query, key, and value projections for head k are defined as

$$\boldsymbol{q}_i^k = \boldsymbol{W}_k^Q \boldsymbol{h}_i, \quad \boldsymbol{k}_j^k = \boldsymbol{W}_k^K \boldsymbol{h}_j, \quad \boldsymbol{v}_j^k = \boldsymbol{W}_k^V \boldsymbol{h}_j,$$

where $\boldsymbol{h}_j$ denotes j-th token feature vector in the sequence. The attention update then takes the form

$$\mathrm{MHA}(\boldsymbol{h}_{1:i}) = \sum_{k=1}^{K} \boldsymbol{W}_k^{O\top} \left(\sum_{j=1}^{i} \alpha_{ij}^k \, \boldsymbol{v}_j^k \right), \text{ where } \alpha_{ij}^k = \mathrm{softmax}_j \left(\left\{ \tfrac{1}{\sqrt{D_r}} (\boldsymbol{k}_{j'}^k)^\top \boldsymbol{q}_i^k \right\}_{j'=1}^{i} \right). \quad (1)$$

Typically, the per-head outputs are concatenated and followed by a single output projection $\boldsymbol{W}^O$. Equivalently, one may view $\boldsymbol{W}^O$ as partitioned into head-specific blocks $\{\boldsymbol{W}_k^O \in \mathbb{R}^{D_r \times D_h}\}_{k=1}^K$, with contributions summed as written above, where D_h is the feature dimension for $\boldsymbol{h}_i$ and D_r is the head dimension where $\boldsymbol{q}_i^k, \boldsymbol{k}_j^k, \boldsymbol{v}_j^k \in \mathbb{R}^{D_r}$. See $Section\ A.1$ for detailed explanation.

Interaction energy. MHA can be derived by considering a gradient step on the following simple interaction energy, similar to that in modern Hopfield networks [Ramsauer et al., 2021]:

$$\epsilon(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i}) = -\tau \sum_{k=1}^{K} \log \sum_{j=1}^{i} \exp\left(\tfrac{1}{\tau} \boldsymbol{\beta}_{kj}^\top \boldsymbol{x}_i \right) \quad \text{where} \quad \boldsymbol{\beta}_{kj} = \boldsymbol{A}_k \boldsymbol{h}_j. \quad (2)$$

Here $\{\boldsymbol{A}_k \in \mathbb{R}^{D_h \times D_h}\}_{k=1}^K$ are learnable projection matrices, D_h is the feature dimension, and τ is a scalar temperature. Our formulation differs from Hopfield networks in two respects: projection weights are embedded directly in the energy and reappear as tied attention projections, and we perform gradient updates rather than Concave-Convex Procedure (CCCP) iterations (see Section C). We now derive the gradient of the interaction energy $\epsilon(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i})$ with respect to $\boldsymbol{x}_i$:

$$\nabla_{\boldsymbol{x}_i} \epsilon(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i}) = -\sum_{k=1}^{K} \sum_{j=1}^{i} \mathrm{softmax}_j \left(\left\{ \tfrac{1}{\tau} \boldsymbol{\beta}_{kj'}^\top \boldsymbol{x}_i \right\}_{j'=1}^{i} \right) \boldsymbol{\beta}_{kj} \,. \quad (3)$$

Adopting a low-rank factorization $\boldsymbol{A}_k = \boldsymbol{W}_k^{Q\top} \boldsymbol{W}_k^K$ we obtain $\boldsymbol{\beta}_{kj} = \boldsymbol{W}_k^{Q\top} (\boldsymbol{W}_k^K \boldsymbol{h}_j)$. If our chosen algorithm is to take a single gradient step, initialized at $\boldsymbol{x}_i = \boldsymbol{h}_i$, then we compute:

$$\boldsymbol{h}_i' = \boldsymbol{h}_i - \eta_\epsilon \, \nabla_{\boldsymbol{x}_i} \epsilon(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i}) \Big|_{\boldsymbol{x}_i = \boldsymbol{h}_i}, \quad (4)$$

with

$$\nabla_{\boldsymbol{x}_i} \epsilon(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i}) \Big|_{\boldsymbol{x}_i = \boldsymbol{h}_i} = -\sum_{k=1}^{K} \boldsymbol{W}_k^{Q\top} \left(\sum_{j=1}^{i} \alpha_{ij}^k \, \boldsymbol{v}_j^k \right)$$
$$\text{where} \quad \alpha_{ij}^k = \mathrm{softmax}_j\left(\left\{ \tfrac{1}{\tau} (\boldsymbol{k}_{j'}^k)^\top \boldsymbol{q}_i^k \right\}_{j'=1}^{i} \right)$$

where the value and key projections are shared: $\boldsymbol{q}_i^k = \boldsymbol{W}_k^Q \boldsymbol{h}_i \,,\ \boldsymbol{v}_j^k = \boldsymbol{k}_j^k = \boldsymbol{W}_k^K \boldsymbol{h}_j$.

It is therefore clear that the gradient of the interaction energy recovers the MHA form, with the weight-tied parameterization

$$\boldsymbol{W}_k^K = \boldsymbol{W}_k^V \qquad \boldsymbol{W}_k^Q = \boldsymbol{W}_k^O \qquad \tau = \sqrt{D_r}\,. \tag{5}$$

In this view, the residual update corresponds exactly to a single gradient descent step (with step size $\eta^\epsilon = 1$) on the defined interaction energy.

2.2 Gradient of element-wise energy yields weight-tied MLPs

Gated MLPs. A gated MLP applies an element-wise transformation to the hidden state $\boldsymbol{h}_i \in \mathbb{R}^{D_h}$:

$$\mathrm{GatedMLP}(\boldsymbol{h}_i) = \boldsymbol{W}^{\mathrm{d}}\big((\boldsymbol{W}^g \boldsymbol{h}_i) \circ \sigma(\boldsymbol{W}^u \boldsymbol{h}_i)\big). \tag{6}$$

Here, the learnable parameters are the *gate* and *up* projections $\boldsymbol{W}^g, \boldsymbol{W}^u \in \mathbb{R}^{D_m \times D_h}$ and the *down* projection $\boldsymbol{W}^{\mathrm{d}} \in \mathbb{R}^{D_h \times D_m}$. The function σ denotes a pointwise nonlinearity (e.g. GELU).

Element-wise energy term. This energy term assigns energy independently to each token feature vector, while sharing the same functional form across positions:

$$\xi(\boldsymbol{x}_i \mid \boldsymbol{h}_i) = -\boldsymbol{\gamma}_i^\top \phi(\boldsymbol{V}\boldsymbol{x}_i), \quad \text{where} \quad \boldsymbol{\gamma}_i = \boldsymbol{W}\boldsymbol{h}_i\,. \tag{7}$$

Here, the learnable parameters are the projection matrices $\boldsymbol{W}, \boldsymbol{V} \in \mathbb{R}^{D_v \times D_h}$, with projection dimension D_v not necessarily equal to the hidden dimension D_h. The function ϕ denotes a pointwise nonlinearity.

Energy-gradient formulation. For the element-wise energy ξ, the gradient with respect to $\boldsymbol{x}_i$ is

$$\nabla_{\boldsymbol{x}_i}\, \xi(\boldsymbol{x}_i \mid \boldsymbol{h}_i) = -\boldsymbol{V}^\top \big(\boldsymbol{\gamma}_i \circ \phi'(\boldsymbol{V}\boldsymbol{x}_i)\big)\,. \tag{8}$$

Taking one gradient step at $\boldsymbol{x}_i = \boldsymbol{h}_i$ yields

$$\boldsymbol{h}_i' = \boldsymbol{h}_i - \eta_\xi\, \nabla_{\boldsymbol{x}_i}\, \xi(\boldsymbol{x}_i \mid \boldsymbol{h}_i)\Big|_{\boldsymbol{x}_i = \boldsymbol{h}_i}\,, \tag{9}$$

with

$$\nabla_{\boldsymbol{x}_i}\, \xi(\boldsymbol{x}_i \mid \boldsymbol{h}_i)\Big|_{\boldsymbol{x}_i = \boldsymbol{h}_i} = -\boldsymbol{V}^\top \big((\boldsymbol{W}\boldsymbol{h}_i) \circ \phi'(\boldsymbol{V}\boldsymbol{h}_i)\big)\,. \tag{10}$$

Comparing equation 6 and equation 10, the energy-gradient update recovers the structure of gated MLPs when we identify the parameters as

$$\boldsymbol{W}^{d\top} = \boldsymbol{W}^u = \boldsymbol{V}\,, \qquad \boldsymbol{W}^g = \boldsymbol{W}\,, \tag{11}$$

and we set $\phi(x) = \int_{-\infty}^{x} \sigma(z)\mathrm{d}z$.

2.3 Enhancing transformer layers from an energy optimization perspective

Having shown that Transformer layers, both MHA and MLPs, can be interpreted as gradient updates on energy functions in Sections 2.1 and 2.2, we next explore how these layers can be enhanced from the perspective of energy optimization.

Diagonal-plus-low-rank parameterization . In Section 2.1, we introduced a low-rank parameterization of $\boldsymbol{A}_k = \boldsymbol{W}_k^{Q\top}\boldsymbol{W}_k^K$ in the interaction energy, recovering the query and key projections of standard attention. We now ask whether a purely low-rank form is sufficient, and instead propose a diagonal-plus-low-rank parameterization for the matrix $\boldsymbol{A}_k$:

$$\boldsymbol{A}_k \;=\; \mathrm{diag}(\boldsymbol{d}_k) \;+\; \boldsymbol{W}_k^{Q\top}\boldsymbol{W}_k^K, \tag{12}$$

where $\boldsymbol{d}_k \in \mathbb{R}^{D_h}$. This augmented parameterization captures key–query interactions that low-rank matrices alone cannot represent, yielding a richer structure for the interaction matrix $\boldsymbol{A}_k$. The diagonal term enriches the interaction matrix but increases computational cost, so we propose sharing it across heads. A detailed empirical analysis is provided in Figure 3b and more background on this parameterization can be found in Section A.2.

Learned lightweight preconditioners. A single gradient step may be insufficient to produce a well-optimized update. Second-order methods such as Newton's method improve optimization by rescaling gradients with curvature information, but computing such curvature is prohibitively expensive in Transformer layers. Inspired by this idea, we introduce lightweight learned preconditioner matrices with a diagonal-plus-low-rank structure,

$$\boldsymbol{P} = \mathrm{diag}\big(\,\mathrm{softplus}(\boldsymbol{d})\big) + \boldsymbol{U}\boldsymbol{V}^\top + \boldsymbol{V}\boldsymbol{U}^\top, \tag{13}$$

where $\boldsymbol{d} \in \mathbb{R}^{D_h}$ and $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{D_h \times R}$ with $R \ll D_h$. The positive diagonal term provides a stable base scaling, while the symmetric low-rank correction captures richer curvature information at negligible cost. We make no claim that $\boldsymbol{P}$ approximates the true Hessian; rather, we treat it as a learned proxy that can capture useful curvature information.

For the interaction energy, we insert per-head matrices $\boldsymbol{P}_k$, giving

$$\Delta \boldsymbol{x}_i^{\epsilon}(\boldsymbol{h}_i \mid \boldsymbol{h}_{1:i}) \coloneqq -\sum_{k=1}^{K} \boldsymbol{P}_k \, \boldsymbol{W}_k^{Q\top} \left(\sum_{j=1}^{i} \alpha_{ij}^k \boldsymbol{v}_j^k \right), \tag{14}$$

$$\text{where} \quad \alpha_{ij}^k = \mathrm{softmax}_j\Big(\big\{ \tfrac{1}{\tau} (\boldsymbol{k}_{j'}^k)^\top \boldsymbol{q}_i^k \big\}_{j'=1}^{i} \Big).$$

This denotes the update evaluated at $\boldsymbol{x}_i = \boldsymbol{h}_i$ for the interaction energy ϵ.

For the element-wise energy, the gated MLP update becomes (contrast with unpreconditioned one in Equation (10)):

$$\Delta \boldsymbol{x}_i^{\xi}(\boldsymbol{h}_i \mid \boldsymbol{h}_{1:i}) \coloneqq -\boldsymbol{P}_{\mathrm{mlp}} \boldsymbol{V}^\top \big((\boldsymbol{W}\boldsymbol{h}_i) \circ \phi'(\boldsymbol{V}\boldsymbol{h}_i) \big), \tag{15}$$

with $\boldsymbol{P}_{\mathrm{mlp}}$ denoting its preconditioner.

In both cases, the preconditioners could be trained to provide lightweight curvature information, enabling updates that converge more effectively to well-optimized states.

Multiple recursive steps So far, each Transformer layer has been interpreted as performing a single gradient step on its associated energy function. From the optimization viewpoint, however, a single step rarely reaches a well-optimized state. A natural extension is therefore to apply multiple recursive updates within the same layer, analogous to running several iterations of an optimization algorithm. For the interaction energy (attention), starting from $\boldsymbol{x}_i^{(0)} = \boldsymbol{h}_i$, we perform T updates of the form

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} - \eta_\epsilon \, \Delta \boldsymbol{x}_i^{\epsilon}(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i}), \tag{16}$$

for $t = 0, \ldots, T-1$ and set $\boldsymbol{h}_i' = \boldsymbol{x}_i^{(T)}$.

For the element-wise energy (MLP), starting from $\boldsymbol{x}_i^{(0)} = \boldsymbol{h}_i$, the recursion is

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} - \eta_\xi \, \Delta \boldsymbol{x}_i^{\xi}(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i}), \tag{17}$$

for $t = 0, \ldots, T-1$, with the output $\boldsymbol{h}_i' = \boldsymbol{x}_i^{(T)}$. This recursive scheme enables each layer to better minimize its energy function without adding parameters, as illustrated in Figure 1. Unlike blockwise recursion in looped Transformer, our approach updates only $\boldsymbol{x}_i^{(t)}$ and fix $\boldsymbol{h}_{1:i}$, with most computation performed outside the recursion. This within-layer recursion thus offers a distinct mechanism that could provide a new dimension for test-time scaling, which we leave for future work.

2.4 A construction of transformer block with energy updates

We now present the full Transformer block from the CEM perspective, where both attention and MLP components arise as recursive gradient updates on their respective energy functions. Residual connections are absorbed into the recursion, while $\mathrm{RMSNorm}(\cdot)$ are applied. Standard Transformer with weight sharing, as detailed in Equations (5) and (11), appears as the special case $T_\epsilon = T_\xi = 1$, using identity preconditioners $(\boldsymbol{P}_k) = (\boldsymbol{P}_{\mathrm{mlp}}) = \boldsymbol{I}$ and vanishing diagonal terms $\boldsymbol{d}_k = \boldsymbol{0}$. The complete CEM block is summarized in Algorithm 1.

A subtlety arises when incorporating positional encodings: rotary embeddings (RoPE) in particular complicate the energy-gradient view by making the projection weights depend on both query and key indices. To avoid this overhead, we instead adopt relative-position biases such as Alibi, as discussed in Section B.

Algorithm 1: Transformer Block as Energy Updates (Orange parts highlight CEM specifics)

Input: Sequence $\boldsymbol{h}_{1:J}$;
Output: Sequence $\boldsymbol{h}'_{1:J}$.
Hyperparameters: Recursive steps T_ϵ, T_ξ, step sizes η_ϵ, η_ξ, heads K, $\phi(x) = \int_{-\infty}^{x} \mathrm{SiLU}(z)\,\mathrm{d}z$.
Trainable parameters: $\{\boldsymbol{W}_k^Q, \boldsymbol{W}_k^K, \boldsymbol{D}_k = \mathrm{diag}(\boldsymbol{d}_k)\}_{k=1}^K$, $\boldsymbol{W}$, $\boldsymbol{V}$, $\{\boldsymbol{P}_k\}_{k=1}^K$, $\boldsymbol{P}_{\mathrm{mlp}}$.

Main Block

for $i = 1 : J$ **do**
  $\boldsymbol{h}_{1:i} \leftarrow \mathrm{RMSNorm}(\boldsymbol{h}_{1:i})$
  for $k = 1 : K$ **do**
    $\boldsymbol{k}_{1:i}^k \leftarrow \boldsymbol{W}_k^K \boldsymbol{h}_{1:i}$
    $\boldsymbol{v}_{1:i}^k \leftarrow \boldsymbol{W}_k^K \boldsymbol{h}_{1:i}$
  $\boldsymbol{h}_i \leftarrow \mathrm{MHA}(\boldsymbol{h}_{1:i}^{1:K}, \boldsymbol{k}_{1:i}^{1:K}, \boldsymbol{v}_{1:i}^{1:K})$
for $i = 1 : J$ **do**
  $\boldsymbol{h}_i \leftarrow \mathrm{RMSNorm}(\boldsymbol{h}_i)$
  $\boldsymbol{h}'_i \leftarrow \mathrm{MLP}(\boldsymbol{h}_i)$
return $\boldsymbol{h}'_{1:J}$

Subroutine: MHA

MHA$(\boldsymbol{h}_{1:i}, \boldsymbol{k}_{1:i}^k, \boldsymbol{v}_{1:i}^k)$:
$\boldsymbol{x}_i \leftarrow \boldsymbol{h}_i$
for $t = 0 : T_\epsilon - 1$ **do**
  $\boldsymbol{u}_i \leftarrow \mathrm{RMSNorm}(\boldsymbol{x}_i)$
  for $k = 1 : K$ **do**
    $\boldsymbol{q}_i^k \leftarrow \boldsymbol{W}_k^Q \boldsymbol{u}_i$
    $\boldsymbol{a}_{ijk} \leftarrow D_h^{-1/2}\big(\boldsymbol{k}_j^{k\top} \boldsymbol{q}_i^k + \boldsymbol{h}_j^\top \boldsymbol{D}_k \boldsymbol{u}_i\big)$
    $\mathbf{o}_i^k \leftarrow \sum_{j=1:i} \mathrm{softmax}_j(\{\boldsymbol{a}_{ijk}\}_{j=1}^i)\, \boldsymbol{v}_j^k$
  $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \eta_\epsilon \sum_k \boldsymbol{P}_k \boldsymbol{W}_k^{Q\top} \mathbf{o}_i^k$
return $\boldsymbol{x}_i$

Subroutine: MLP

MLP$(\boldsymbol{h}_i)$:
$\boldsymbol{x}_i \leftarrow \boldsymbol{h}_i$
$\boldsymbol{\gamma} = \boldsymbol{W}\boldsymbol{h}_i$
for $t = 0 : T_\xi - 1$ **do**
  $\boldsymbol{u}_i \leftarrow \mathrm{RMSNorm}(\boldsymbol{x}_i)$
  $\boldsymbol{g}_i \leftarrow \boldsymbol{V}^\top\big(\boldsymbol{\gamma} \circ \phi'(\boldsymbol{V}\boldsymbol{u}_i)\big)$
  $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \eta_\xi \boldsymbol{P}_{\mathrm{mlp}}\, \boldsymbol{g}_i$
return $\boldsymbol{x}_i$

3 Related Work

Energy based sequence models. EBMs assign low energies to preferred configurations [Hopfield, 1982, LeCun et al., 2006]. Modern extensions to Hopfield networks [Krotov and Hopfield, 2016, Ramsauer et al., 2021] with continuous patterns and log-sum-exp energy demonstrate how attention-like updates can arise from their updating iterations. Subsequent work on sequence-level EBMs extends these ideas to language modeling and text generation [Du et al., 2020, Liu et al., 2023, Qin et al., 2022]. More recently, Gladstone et al. [2025], Hoover et al. [2023] explore using energy minimization as building blocks for large language models, providing a new paradigm for scaling learning and thinking capacity. In contrast, our work does not treat energy minimization procedures as layers. Instead, we show that existing Transformer layers, including both MHA and MLP, with weight-sharing, can already be reframed as energy-based updates. This perspective enables principled layer redesigns and extensions, and leads to improved parameter efficiency in Transformers.

Optimization and probabilistic-inference views of Transformers. Several works have connected Transformers with optimization or probabilistic inference. Yang et al. [2022] view Transformers as unfolded optimization procedures; Wu and Tu [2023] derive attention-like updates from mean-field inference; and Ravuri and Lawrence [2025] interprets each Transformer block as performing gradient descent on a variational lower bound of the probabilistic Laplacian Eigenmaps model. Closest to our work, Dehmamy et al. [2025] derive attention and feed-forward from gradients of an unconditional energy. In comparisons, CEM studies causal, conditional energy updates, with particular emphasis on the parameterizations they induce and the layer extensions they suggest.

Alternative transformer blocks. Transformer models have largely converged on Llama-style backbones with multi-head attention [Vaswani et al., 2017] and gated MLPs [Shazeer, 2020]. Many efficiency-oriented variants reduce the cost of attention via multi-query, group-query, or multi-head latent attention [Ainslie et al., 2023, Liu et al., 2024, Shazeer, 2019], while others target the feedforward block [Liu et al., 2021, Shazeer, 2020, So et al., 2021]. Sparsely activated mixture-of-experts (MoE) layers scale capacity [Fedus et al., 2022, Shazeer et al., 2017], and other work simplifies skip connections, projections, or normalization with little performance loss [He and Hofmann, 2024, He et al., 2023]. State-space models (SSMs) [Gu and Dao, 2024, Gu et al., 2022, Yang et al., 2024] have also shown strong results, with connections to attention noted by Dao and Gu [2024], though hybrid designs remain necessary for state-of-the-art performance [Kimi Team et al., 2025].

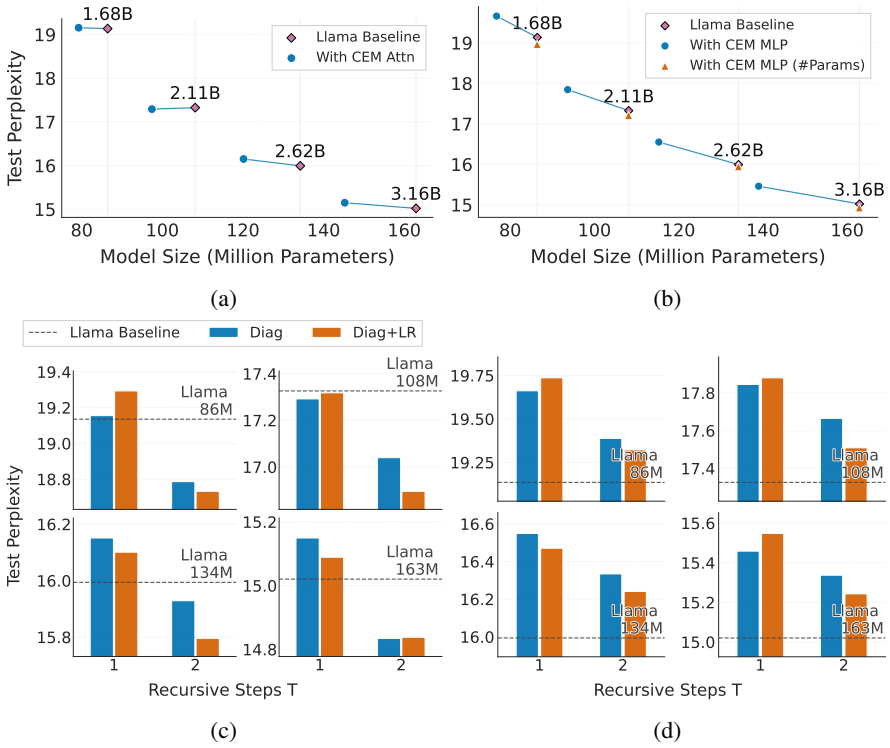

Figure 2: (a) Llama Transformer with attention replaced by CEM attention ($T = 1$). CEM variants (blue dots) are linked to their Llama baselines (pink diamonds) with matching dimensions but fewer parameters, trained on the same number of tokens (shown above markers). (b) Llama Transformer with gated MLPs replaced by CEM MLPs ($T = 1$). Orange triangles additionally show parameter-matched variants obtained by increasing the hidden dimension. (c) Effects of recursion steps (x-axis) and preconditioners (colors) for CEM attention, with all models dimension-matched to Llama baselines (dashed lines). (d) Effects of recursion steps and preconditioners for CEM MLPs.

4 Experiments

Our experiments address four questions: (i) Do the weight-sharing schemes induced by CEM lead to significant performance degradation? (ii) do within-layer recursion and lightweight preconditioners improve performance; (iii) can a Transformer composed entirely of CEM layers be trained end-to-end stably; and (iv) how do design choices such as KQ diagonal terms and recursion affect performance. All models are trained on SlimPajama for the compute-optimal number of tokens of the corresponding Llama baselines [Hoffmann et al., 2022], and we report test perplexity as the main evaluation metric. Experimental details can be found in Section D.

4.1 Replace Transformer layers with single-step CEM layers

To evaluate the effectiveness of CEM layers, we train Transformer models with CEM components in either the MLP or attention blocks, and compare against Llama baselines. We focus on the weight-tying formulation (see Equations 5 and 11), but without recursions or preconditioners here.

Figure 2a compares CEM attention with standard Llama MHAs, while Figure 2b compares CEM MLPs with Llama-style gated MLPs. Blue dots denote dimension-matched CEM models, where CEM attention uses about half the parameters and CEM MLPs about two-thirds of their Llama counterparts. Some degradation is expected, but the goal is to assess how closely CEM models approach baseline performance with fewer parameters. For CEM MLPs, we also report results with increased intermediate dimension to restore the baseline parameter count (orange triangles).

Replacing attention with the CEM variant has only a small effect on test perplexity despite halving the parameter count, with no natural parameter-matching scheme available since the model dimension

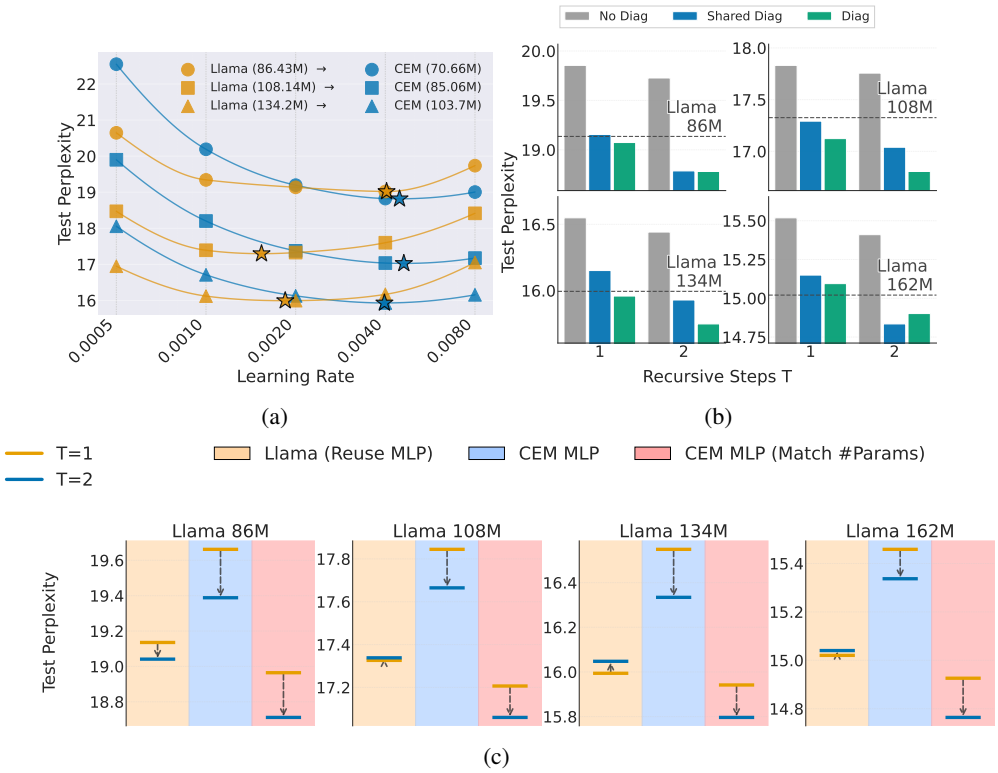

Figure 3: (a) Optimal learning rate estimated via Akima interpolation. Orange denotes baseline Llama models and blue denotes CEM models ($T = 2$) with both MHA and MLP replaced. Marker shapes indicate model size; stars mark interpolated optima from five data points. Matched Llama and CEM models (with roughly half the MHA and two-third the MLP parameters) are trained with the same token budget (Chinchilla-optimal for Llama). For smaller models, parameter reduction is less pronounced due to embedding and head parameters. (b) KQ diagonal strategies in CEM attention: no diagonal in $\boldsymbol{A}_k$, a shared diagonal across heads, and per-head diagonals. All CEM models match the dimensionality of the Llama baselines (dashed line). (c) Within-layer recursion vs. plain layer reuse in MLPs. We compare the performance gains of increasing recursion from $T = 1$ (orange) to $T = 2$ (blue), under three settings: Plain layer reuse (light orange area), dimension-matched CEM MLP (blue area) and parameter-matched MLPs (pink area). An equivalent figure comparing within-layer recursion vs. layer reuse for MHA can be found in Figure 4.

must remain fixed for controlled comparison. For CEM-MLPs, perplexity is higher due to parameter sharing, but increasing the hidden dimension to match parameter count yields consistent, albeit modest, improvements in perplexity — though at the cost of additional FLOPs. Unless otherwise noted, we adopt the optimal Llama hyperparameters from grid search (see Section D) to make consistent comparisons and avoid tuning each CEM configuration individually, even though these settings may be suboptimal for CEM models (see Figure 3a). These results indicate that single-step CEM layers define constrained, parameter-efficient variants of standard Transformer components that can retain competitive perplexity in this controlled setting. The results are strongest for CEM attention, suggesting that the corresponding weight-sharing parameterization merits further investigation.

4.2 Recursive steps and preconditioners in CEM layers

We test whether within-layer recursion and lightweight preconditioners improve performance. Transformer variants are trained on SlimPajama to the compute-optimal token budget of their Llama baselines and evaluated by test perplexity. Figure 2c and Figure 2d replace the standard Llama MHAs and MLPs with their CEM counterparts, respectively. We compare diagonal and diagonal-plus-low-rank preconditioners for $T = 1$ and $T = 2$. All models are dimension-matched to their Llama baselines, so CEM components use fewer parameters, and preconditioners add negligible overhead.

As shown in Figure 2, the two ingredients contribute unequally. Increasing recursion from $T = 1$ to $T = 2$ consistently improves CEM layers, and for attention, recursive CEM variants can outperform the Llama baseline while using fewer parameters. For MLPs, CEM variants remain below the baseline, but the gap narrows at $T = 2$. Learned preconditioners, by contrast, contribute only marginally: they cannot provide consistent gains especially for $T = 1$, suggesting that the recursive structure, rather than the preconditioner, drives most of the improvement.

We did not observe consistent gains for $T \geq 3$ under our standard training setup, likely due to the difficulty of optimizing through additional unrolled iterations. However, controlled experiments up to $T = 8$ on a synthetic problem (Table 3) show that deeper recursion can yield further improvements. Thus, scaling beyond $T = 2$ in full Transformer settings remains promising but will likely require advances in optimization for unrolled architectures, which we leave to future work.

4.3 Training Transformers with CEM-derived layers

We next test whether a Transformer composed entirely of CEM-derived attention and MLP layers can be trained end-to-end. We use $T = 2$ with diagonal-plus-low-rank preconditioners for both components, which keeps the constrained CEM parameterization: about half the attention parameters and two-thirds the MLP parameters of the standard counterparts. Due to memory constraints, we omit the largest model with diagonal-plus-low-rank preconditioners.

We sweep five learning rates from 0.0005 to 0.008 and use Akima interpolation [Akima, 1970] to estimate the optimum. Figure 3a reports the interpolated test perplexity. We additionally report a 256M-parameter setting in Figure 5 of the Appendix, where the same observation holds.

End-to-end CEM models train stably and achieve perplexity comparable to the corresponding Llama baselines while using fewer parameters. The interpolated optima suggest that CEM models may prefer higher learning rates, though this trend requires further study.

4.4 Ablation study

We first study the role of diagonal terms in inter-token distances (Figure 3b) in attention, comparing three settings: no diagonal, a shared diagonal across heads, and per-head diagonals. All other components are fixed (Llama MLPs, CEM attention with one recursion step, and a simple diagonal preconditioner). Including a diagonal term proves essential for good performance, and our shared-diagonal strategy provides performance close to per-head diagonals while reducing parameters and compute, making it a more efficient alternative.

Second, we test whether within-layer recursion is necessary or if naive layer reuse suffices (Figure 3c). Simply reapplying the same residual block yields little or no perplexity gain. Note that this reuse differs from recursive Transformer, where entire blocks (attention and MLP) are reused. In contrast, CEM-based within-layer recursion produces consistent improvements in both dimension- and parameter-matched settings. A similar trend holds for attention (Figure 4).

5 Discussion and Conclusion

5.1 Limitations

This work studies Transformer layer parameterization through the lens of energy updates, but developing scalable and performant new architectures will require further investigation. Our experiments therefore provide a controlled evaluation at the hundred-million-parameter scale, using test perplexity as a standard proxy for language-model quality. Larger-scale scaling studies and downstream evaluations, which become most meaningful at larger model sizes, are natural next steps for assessing how the benefits of CEM-derived layers transfer to larger models and practical tasks. In addition, improving the efficiency of these layers will likely require custom kernels and systems-level optimization. Our implementation focuses on validating the proposed parameterizations rather than optimizing runtime performance. Because CEM-derived layers introduce structured weight sharing and recursive updates, custom kernels, fused operations, and hardware-aware implementations will be needed to improve their practical runtime behavior.

5.2 Conclusion and future directions

We introduced CEM, a framework that recasts Transformer layers as causal energy minimization. From this view, weight-tied attention and gated MLPs emerge as energy-gradient updates, motivating optimization-inspired extensions such as diagonal-plus-low-rank parameterization, lightweight preconditioners, and recursive updates. These CEM-derived layers approach or match Transformer baselines in moderate-scale language modeling, with recursion and diagonal-plus-low-rank parameterizations yielding the most consistent gains and preconditioners providing more marginal improvements. Overall, CEM offers a new lens for rethinking Transformer parameterizations.

We believe the following directions are particularly promising: (1) **Custom kernels**, where FlashAttention-style IO-aware implementations [Dao et al., 2022] could fuse tied projections, diagonal–plus-low-rank terms, and recursive updates to improve throughput; (2) **Test-time scaling**, where CEM-style within-layer recursion may provide an additional axis for scaling test-time compute [Muennighoff et al., 2025, Snell et al., 2024] and for supporting latent reasoning [Hao et al., 2024, Tan et al., 2025, Zhang and Viteri, 2024], especially when combined with blockwise recursion as in looped or recursive Transformers [Bae et al., 2024, Dehghani et al., 2019, Yang et al., 2023]; and (3) **Architecture–hardware co-design**, where alternative optimization procedures for the same underlying energy could yield layer parameterizations that are jointly designed with new hardware specifics and kernel implementation.

Acknowledgments and Disclosure of Funding

We would like to thank Riccardo Grazzi, Elon Portugaly, Babak Rahmani, Jannes Gladrow, Taketomo Isazawa, and many others at Microsoft Research Cambridge for their valuable discussions and early feedback on this work. We also thank the anonymous reviewers for their constructive suggestions, which helped improve the clarity and presentation of the paper.

References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, 2023.

Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM*, 17:589–602, 1970.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. *ArXiv*, abs/2410.20672, 2024.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Koulako Bala Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel J. Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan,

Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Silvère Bonnabel, Marc Lambert, and Francis Bach. Low-rank plus diagonal approximations for riccati-like matrix differential equations. *SIAM Journal on Matrix Analysis and Applications*, 45(3): 1669–1688, 2024. doi: 10.1137/23M1587610. URL https://doi.org/10.1137/23M1587610.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*, 2024.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Nima Dehmamy, Benjamin Hoover, Bishwajit Saha, Leo Kozachkov, Jean-Jacques Slotine, and Dmitry Krotov. Nrgpt: An energy-based alternative for gpt. *arXiv preprint arXiv:2512.16762*, 2025.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR, 2017.

Alexi Gladstone, Ganesh Nanduru, Md Mofijul Islam, Peixuan Han, Hyeonjeong Ha, Aman Chadha, Yilun Du, Heng Ji, Jundong Li, and Tariq Iqbal. Energy-based transformers are scalable learners and thinkers. *arXiv preprint arXiv:2507.02092*, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *ICLR*, 2022.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Bobby He and Thomas Hofmann. Simplifying transformer blocks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RtDok9eS3s.

Bobby He, James Martens, Guodong Zhang, Aleksandar Botev, Andrew Brock, Samuel L Smith, and Yee Whye Teh. Deep transformers without shortcuts: Modifying self-attention for faithful signal propagation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NPrsUQgMjKK.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural Computation*, volume 9, pages 1735–1780. MIT Press, 1997.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=iBBcRUlOAPR.

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed Zaki, and Dmitry Krotov. Energy transformer. *Advances in neural information processing systems*, 36:27532–27559, 2023.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

Yu Kimi Team, Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in Neural Information Processing Systems*, 29, 2016.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *CoRR*, 2024.

Hanxiao Liu, Zihang Dai, David R. So, and Quoc V. Le. Pay attention to mlps. In *Neural Information Processing Systems*, 2021.

Xin Liu, Muhammad Khalifa, and Lu Wang. Bolt: Fast energy-based controlled text generation with tunable biases. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=EknvgeZ4Jwq.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. COLD decoding: Energy-based constrained text generation with langevin dynamics. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TiZYrQ-mPup.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, et al. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021.

Aditya Ravuri and Neil D Lawrence. Transformers as unrolled inference in probabilistic laplacian eigenmaps: An interpretation and potential improvements. *arXiv preprint arXiv:2507.21040*, 2025.

Michael E Sander, Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics*, pages 3515–3530. PMLR, 2022.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. In *arXiv preprint arXiv:1911.02150*, 2019.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=B1ckMDqlg`.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. URL `https://arxiv.org/abs/2408.03314`. Preprint; ICLR 2025 version on OpenReview.

David R. So, Wojciech Ma'nke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V. Le. Primer: Searching for efficient transformers for language modeling. *ArXiv*, abs/2109.08668, 2021.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, 2023. URL `https://huggingface.co/datasets/cerebras/SlimPajama-627B`.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *ArXiv*, abs/2505.16552, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Haoyi Wu and Kewei Tu. Probabilistic transformer: A probabilistic dependency model for contextual word representation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7613–7636, 2023.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. *ArXiv*, abs/2311.12424, 2023.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37: 115491–115522, 2024.

Yongyi Yang, David P Wipf, et al. Transformers from an optimization perspective. *Advances in Neural Information Processing Systems*, 35:36958–36971, 2022.

Alan Loddon Yuille and Anand Rangarajan. The concave-convex procedure (cccp). In *Neural Information Processing Systems*, 2001.

Jason Zhang and Scott Viteri. Uncovering latent chain of thought vectors in language models. *arXiv preprint arXiv:2409.14026*, 2024. URL `https://arxiv.org/abs/2409.14026`.

A Background

A.1 Equivalence between concatenation and summation in attention

In Section 2.1, we write the multi-head attention update in the form

$$\mathrm{MHA}(\boldsymbol{h}_{1:i}) = \sum_{k=1}^{K} \boldsymbol{W}_k^{O\top} \left(\sum_{j=1}^{i} \mathrm{softmax}_j \Big(\big\{ \tfrac{1}{\sqrt{D_h}} (\boldsymbol{k}_{j'}^k)^\top \boldsymbol{q}_i^k \big\}_{j'=1}^{i} \Big) \boldsymbol{v}_j^k \right),$$

where each head k contributes an output vector that is multiplied by a head-specific block $\boldsymbol{W}_k^O \in \mathbb{R}^{D_r \times D_h}$.

This notation differs slightly from the conventional implementation of multi-head attention, where per-head outputs are concatenated and processed by a single output projection. To make the equivalence explicit, let

$$O_k \;\coloneqq\; \sum_{j=1}^{i} \mathrm{softmax}_j \Big(\big\{ \tfrac{1}{\sqrt{D_h}} (\boldsymbol{k}_{j'}^k)^\top \boldsymbol{q}_i^k \big\}_{j'=1}^{i} \Big) \boldsymbol{v}_j^k \in \mathbb{R}^{D_r}$$

denote the output of head k. The standard formulation concatenates these outputs,

$$O = [\, O_1;\, O_2;\, \ldots;\, O_K \,] \in \mathbb{R}^{KD_r},$$

where ; denotes vertical stacking of matrices, and applies a single output projection $\boldsymbol{W}^O \in \mathbb{R}^{KD_r \times D_h}$.

If we partition $\boldsymbol{W}^O$ into head-aligned blocks,

$$\boldsymbol{W}^O = [\, \boldsymbol{W}_1^O;\, \boldsymbol{W}_2^O;\, \ldots;\, \boldsymbol{W}_K^O \,], \qquad \boldsymbol{W}_k^O \in \mathbb{R}^{D_r \times D_h},$$

then multiplying out gives

$$\boldsymbol{W}^{O\top} O = [\, \boldsymbol{W}_1^{O\top}, \ldots, \boldsymbol{W}_K^{O\top} \,][\, O_1;\, O_2;\, \ldots;\, O_K \,] = \sum_{k=1}^{K} \boldsymbol{W}_k^{O\top} O_k.$$

Thus, the conventional *concatenation followed by a single projection* is algebraically equivalent to the *sum over head-specific projections* used in our presentation. We adopt the latter form for notational clarity in the CEM formulation.

A.2 Diagonal-plus-low-rank parameterization

We use the diagonal-plus-low-rank (D+LR) parameterization for both the attention-score computation (in Equation (12)) and the preconditioners (in Equation (13). Here we provide a brief background on this parameterization.

The basic form of a D+LR matrix is often written as

$$W = \mathrm{diag}(d) + UV^\top, \qquad U, V \in \mathbb{R}^{d \times r}, \quad r \ll d.$$

Compared with a pure diagonal parameterization, which cannot express cross-feature interactions, and a pure low-rank parameterization, which only captures interactions within a rank-r subspace, the D+LR form models both aspects. The memory footprint and computational cost are still much smaller compared to the full matrix: applying W requires one diagonal pass and two thin matrix multiplications, with complexity $O(d) + O(dr)$, far smaller than the $O(d^2)$ cost of a dense matrix. D+LR parameterizations have already been widely used in deep learning such as [Bonnabel et al., 2024, Gu et al., 2022].

B Incorporating positional encoding into CEM attention

Positional encoding. Standard Transformer architectures such as Llama employ Rotary Position Embeddings (RoPE) [Su et al., 2024] to encode relative position information. Recall from Section 2.1 that in our energy-based formulation, each head k is parameterized by a matrix $\boldsymbol{A}_k$:

$$\boldsymbol{\beta}_{kj} = \boldsymbol{A}_k \boldsymbol{h}_j, \qquad \text{with } \boldsymbol{A}_k \in \mathbb{R}^{D_h \times D_h}.$$

In the simplest case, we adopt a low-rank factorization $\boldsymbol{A}_k = \boldsymbol{W}_k^{Q\top} \boldsymbol{W}_k^K$, so that queries, keys, and values arise as

$$\boldsymbol{q}_i^k = \boldsymbol{W}_k^Q \boldsymbol{h}_i, \quad \boldsymbol{k}_j^k = \boldsymbol{W}_k^K \boldsymbol{h}_j, \quad \boldsymbol{v}_j^k = \boldsymbol{W}_k^V \boldsymbol{h}_j,$$

under the weight-tying constraints $\boldsymbol{W}_k^K = \boldsymbol{W}_k^V$ and $\boldsymbol{W}_k^Q = \boldsymbol{W}_k^O$ (see equation 5). The interaction energy is then defined as

$$\epsilon(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i}) = -\tau \sum_{k=1}^{K} \log \sum_{j=1}^{i} \exp\Big(\tfrac{1}{\tau}\, \boldsymbol{\beta}_{kj}^\top \boldsymbol{x}_i\Big),$$

and its gradient update recovers the standard multi-head attention form with weight sharing.

When incorporating RoPE, however, $\boldsymbol{A}_k$ must depend explicitly on both indices i and j through rotation matrices $\boldsymbol{R}(i)$ and $\boldsymbol{R}(j)$:

$$\boldsymbol{A}_k \;=\; \boldsymbol{W}_k^{Q\top}\, \boldsymbol{R}(i)^\top \boldsymbol{R}(j)\, \boldsymbol{W}_k^K.$$

This makes $\boldsymbol{\beta}_{kj}$ dependent on the query index i as well as j, which substantially increases memory costs: the value projection effectively becomes query-dependent.

Alibi positional encodings. To mitigate this overhead, we instead adopt **Alibi positional encodings** [Press et al., 2022], which introduce a head-specific bias

$$b_{ijk} = -m_k |i - j|$$

directly into the attention scores before the softmax. Concretely, in the unbiased case the score is

$$s_{ijk} = \tfrac{1}{\tau}\, \boldsymbol{\beta}_{kj}^\top \boldsymbol{x}_i,$$

so with Alibi it becomes

$$s_{ijk} \;=\; \tfrac{1}{\tau}\, \boldsymbol{\beta}_{kj}^\top \boldsymbol{x}_i \;+\; b_{ijk},$$

and the normalized weights are

$$\alpha_{ij}^k = \mathrm{softmax}_j\Big(\{s_{ij'k}\}_{j'=1}^{i}\Big).$$

The slopes m_k are typically chosen as a geometric sequence, e.g. $m_k = 2^{-k}$. This adds negligible overhead compared to RoPE while still encoding relative bias. In practice, we further include a learnable bias distinguishing self- vs. cross-token attention:

$$b_{ijk} = -m_k |i - j| + b_{i=j} + b_{i\neq j}.$$

Interaction energy with bias. In the energy formulation, this simply shifts the logits inside the log-sum-exp:

$$\epsilon(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i}) = -\tau \sum_{k=1}^{K} \log \sum_{j=1}^{i} \exp\Big(\tfrac{1}{\tau}\, \boldsymbol{\beta}_{kj}^\top \boldsymbol{x}_i + b_{ijk}\Big).$$

The corresponding gradient update is

$$\nabla_{\boldsymbol{x}_i}\, \epsilon(\boldsymbol{x}_i \mid \boldsymbol{h}_{1:i}) = -\sum_{k=1}^{K} \sum_{j=1}^{i} \mathrm{softmax}_j\Big(\tfrac{1}{\tau}\, \boldsymbol{\beta}_{kj}^\top \boldsymbol{x}_i + b_{ijk}\Big)\, \boldsymbol{\beta}_{kj},$$

so b_{ijk} modifies the logits before normalization but leaves the overall gradient structure unchanged.

C Relation to Hopfield networks

Hopfield networks are classical models of associative memory, where stored patterns correspond to attractors of an energy landscape, and the dynamics converge to the attractor most consistent with the initial state. This viewpoint aligns with our interpretation of Transformer layers as energy-minimizing updates: both attention and MLP sublayers can be seen as iterative steps that decrease a suitably defined energy function. We next detail these connections.

Interaction energy. Classical Hopfield networks [Hopfield, 1982] store a finite set of patterns $\{\boldsymbol{h}_j\}$ in an energy function of the form

$$\epsilon(\boldsymbol{x}) = -\tfrac{1}{2}\sum_j (\boldsymbol{h}_j^\top \boldsymbol{x})^2,$$

More recent extensions reinterpret Hopfield networks as continuous attractor models, greatly expanding their representational capacity. For instance, dense associative memories [Krotov and Hopfield, 2016] and modern Hopfield networks [Ramsauer et al., 2021] introduce an energy of the log-sum-exp form,

$$\epsilon(\boldsymbol{x}) = -\tau \log \sum_j \exp\Big(\tfrac{1}{\tau}\boldsymbol{h}_j^\top \boldsymbol{x}\Big),$$

which is convex in $\boldsymbol{x}$ and whose fixed-point updates under the concave-convex procedure (CCCP) [Yuille and Rangarajan, 2001] yield

$$\boldsymbol{x}' = \sum_j \mathrm{softmax}_j\Big(\tfrac{1}{\tau}\boldsymbol{h}_j^\top \boldsymbol{x}\Big)\,\boldsymbol{h}_j,$$

exactly the update rule underlying the attention mechanism. This connection underlies the interpretation of attention as a form of fast Hopfield retrieval.

Our perspective. We depart from the setup of modern Hopfield networks in three important ways. First, instead of computing fixed points via iterative CCCP updates [Yuille and Rangarajan, 2001], we interpret each Transformer sublayer as performing a *single gradient step* on an energy function. Second, in our formulation the query and key projection matrices are embedded directly in the energy, which causes them to reappear as the output–value projections in the gradient update—naturally yielding the tied $\boldsymbol{W}_Q, \boldsymbol{W}_K$ and $\boldsymbol{W}_O, \boldsymbol{W}_V$ structure of attention. Finally, while Ramsauer et al. [2021] introduce novel Hopfield layers and evaluate them on associative-memory benchmarks, our framework treats standard Transformer layers themselves as energy-based updates, and we demonstrate that this perspective leads to principled extensions and improvements for text modeling tasks.

Element-wise energy. The element-wise energy leading to gated MLPs has a less direct connection. Optimization via CCCP is possible only when using a convex form. We briefly experimented with models using energies of the form

$$\xi(\boldsymbol{x}_i \mid \boldsymbol{h}_i) = -|\boldsymbol{\gamma}_i|^\top \phi\big(\,\mathrm{diag}(\mathrm{sign}(\boldsymbol{\gamma}_i))\,\boldsymbol{V}\boldsymbol{x}_i\big), \qquad \boldsymbol{\gamma}_i = \boldsymbol{W}\boldsymbol{h}_i,$$

with ϕ a convex nonlinearity, so that the energy is convex in $\boldsymbol{x}$. The gradient of this energy form is

$$-\boldsymbol{V}^\top\left(\boldsymbol{\gamma}_i \circ \phi'(\mathrm{diag}(\mathrm{sign}(\boldsymbol{\gamma}_i))\boldsymbol{V}\boldsymbol{x}_i)\right)$$

We used a straight-through estimator to deal with the sign nonlinearity. We found that these models successfully trained, but with worse performance than ignoring the sign. Unlike the interaction energy, the link to memory association here is unclear, as are the corresponding convergence guarantees and capacity limits.

D Experimental details

D.1 Model architectures

We evaluate CEM-based architectures across multiple model scales ranging from 86M to 162M parameters. All models follow the Llama architecture as baseline with modifications for CEM components. Table 1 summarizes the architectural details for each model size.

D.2 Dataset and preprocessing

Dataset. We use a subset of SlimPajama-627B [Soboleva et al., 2023], a cleaned and deduplicated variant of RedPajama comprising approximately 627 billion tokens drawn from CommonCrawl, C4, GitHub, books, arXiv, Wikipedia, and StackExchange. The dataset is accessed via `gmongaras/SlimPajama-627B_Reupload` on Hugging Face and is released under the Apache 2.0 license.

Table 1: Model architecture configurations for different parameter counts. All models use a vocabulary size of 32,000 tokens.

Configuration	86M	108M	134M	162M
Model dimension (d_h)	672	672	768	864
Number of layers	8	12	12	12
Number of heads	8	12	12	12
MLP intermediate dimension	1792	1792	2048	2304
Context length	2048	2048	2048	2048

Tokenization. We employ the `LlamaTokenizerFast` with a vocabulary size of 32,000 tokens.

Data processing. Documents are concatenated and split into fixed-length sequences of 2048 tokens, with no padding applied.

D.3 Training configuration

Training hyperparameters are summarized in Table 2. We follow Chinchilla-optimal compute allocation [Hoffmann et al., 2022] for determining the number of training tokens for each model size.

Table 2: Training hyperparameters for CEM models and Llama baselines.

Hyperparameter	CEM models	Llama baseline
Optimizer	AdamW	
Learning rate	0.002	
β_1	0.9	
β_2	0.95	
ϵ	1e-9	
Weight decay	0.1	
Gradient clipping	1.0	
LR schedule	Cosine	
Warmup steps	5% of total	
Final LR factor	0.1	
Batch size (per GPU)	8	
Gradient accumulation	4	
Effective batch size	128	
Precision	bf16-mixed	

D.4 Initialisation of preconditioners

In Section 2.3, we introduce a trainable diagonal-plus-low-rank preconditioner of the form

$$\boldsymbol{P} = \mathrm{diag}\big(\mathrm{softplus}(\boldsymbol{d})\big) + \boldsymbol{U}\boldsymbol{V}^\top + \boldsymbol{V}\boldsymbol{U}^\top.$$

with $\boldsymbol{d} \in \mathbb{R}^{D_h}$ and $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{D_h \times R}$, where $R \ll D_h$. Following Hu et al. [2021], we initialize $\boldsymbol{U}$ from a normal distribution ($\sigma = 0.02$) and set $\boldsymbol{V}$ to zeros. For the diagonal term, we parameterize

$$\boldsymbol{d} = \sqrt{D_h}\,\boldsymbol{p},$$

where $\boldsymbol{p}$ is initialized to $1/\sqrt{D_h}$. This ensures that $\boldsymbol{d}$ starts at 1, but still yielding an appropriate effective gradient step size.

To keep the preconditioners lightweight, we set $R = 4$ for attention modules and $R = 16$ for MLPs. In the diagonal-only case, the preconditioner reduces to

$$\boldsymbol{P} = \mathrm{diag}\big(\mathrm{softplus}(\boldsymbol{d})\big).$$

D.5 Compute resources

All experiments were conducted on a cluster of $8\times$ NVIDIA A100 GPUs (40GB memory each). Training time per model scales with size: the smallest models ($\sim$86M parameters) require about 8×2 GPU-hours, while the largest models we tested ($\sim$162M parameters) require about 8×18 GPU-hours. End-to-end reproduction of all results in this paper would therefore require on the order of $10{,}000$ GPU-hours.

E Additional results

Recursive updates in MHA Similar to our analysis of MLP recursion Figure 3c, we examine recursive updates in attention layers (Figure 4). As with MLPs, naive reuse of the same MHA block offers no benefit and can even degrade performance in the case of MHA. In contrast, within-layer recursion in CEM attention yields clear and consistent perplexity improvements.

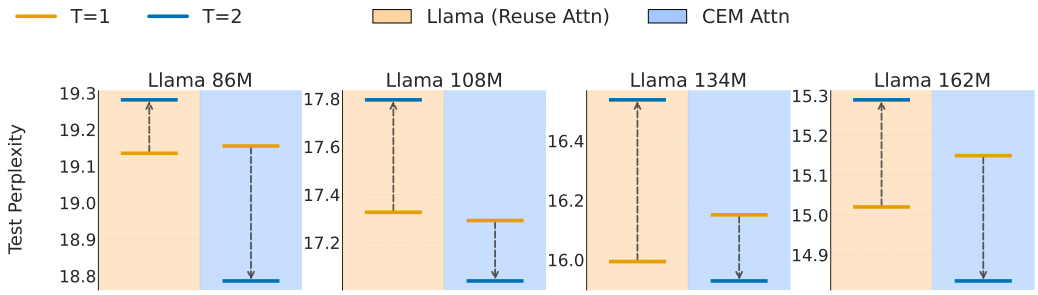

Figure 4: Within-layer recursion vs. plain layer reuse in MHAs. We compare the performance gains of increasing recursion from $T = 1$ (orange) to $T = 2$ (blue), under two settings: Plain layer reuse (light orange area), dimension-matched CEM MHA (blue area).

Study recursion on synthetic data To better isolate and understand the intrinsic behaviour of the recursion, we also examine it in a controlled and computationally lightweight setting using Gaussian-process generated data and fit with our recursive CEM MLP. The results in Table 3, illustrate that additional recursive steps generally improve performance, though the gains are not strictly monotonic.

Akima interpolation with larger models We scale our models to 256M parameters, and results analogous to Figure 3a are shown in Figure 5. We find that CEM models, despite having fewer parameters, continue to outperform Llama models at this scale. Scaling to substantially larger sizes would require significantly more computational resources, and we leave this for future work.

F LLM Usage Statement

We used ChatGPT-5 to assist with paraphrasing, text editing, and proofreading. For most paragraphs, we first wrote a draft and then asked ChatGPT to paraphrase it without changing the original meaning. We checked that the paraphrased text did not alter the meaning. We also used ChatGPT to help search for and discover relevant related work, but all bibliographic entries were manually verified for correctness. All conceptual development, technical contributions, experiments, and analysis were carried out by the authors.

G Reproducibility

We provide experimental details in Section D. Model architectures are given in Table 1, and training configurations in Table 2. All experiments use the SlimPajama dataset (Section D.2) and were conducted on 8× NVIDIA A100 GPUs. The code is not yet publicly available but will be released upon publication.

Table 3: Model size, compute, and RMSE (mean $\pm$ std) on synthetic data sampled from Gaussian processes with 10 input dimensions. Here the latent state dimension is set equal to the model dimension. Values are averaged over repeated runs, and the lowest (best) train and test RMSE for each kernel are highlighted in **bold**. FLOPs are reported per forward pass. (Abbreviations: K $= 10^3$, M $= 10^6$.)

MODEL	PARAMETERS	FLOPS	KERNEL	TRAIN RMSE	TEST RMSE
Plain	33.98K	6.76M	RBF	0.0139 ± 0.0022	0.0158 ± 0.0024
Gated	33.94K	6.73M	RBF	0.0109 ± 0.0014	0.0122 ± 0.0015
CEM-T1	22.66K	6.73M	RBF	0.0102 ± 0.0010	0.0119 ± 0.0011
CEM-T2	22.66K	11.08M	RBF	0.0074 ± 0.0008	0.0092 ± 0.0008
CEM-T4	22.66K	19.79M	RBF	0.0068 ± 0.0006	$\mathbf{0.0086 \pm 0.0007}$
CEM-T8	22.66K	37.20M	RBF	$\mathbf{0.0066 \pm 0.0008}$	0.0088 ± 0.0012
Plain	33.98K	6.76M	Matern	0.2129 ± 0.0205	0.2308 ± 0.0237
Gated	33.94K	6.73M	Matern	0.1903 ± 0.0205	0.2162 ± 0.0253
CEM-T1	22.66K	6.73M	Matern	0.1836 ± 0.0199	0.2194 ± 0.0267
CEM-T2	22.66K	11.08M	Matern	$\mathbf{0.1623 \pm 0.0170}$	0.2166 ± 0.0274
CEM-T4	22.66K	19.79M	Matern	0.1742 ± 0.0206	0.2168 ± 0.0271
CEM-T8	22.66K	37.20M	Matern	0.1676 ± 0.0174	$\mathbf{0.2161 \pm 0.0248}$
Plain	33.98K	6.76M	Periodic	0.0557 ± 0.0056	0.0614 ± 0.0058
Gated	33.94K	6.73M	Periodic	0.0386 ± 0.0036	0.0445 ± 0.0043
CEM-T1	22.66K	6.73M	Periodic	0.0352 ± 0.0036	0.0421 ± 0.0044
CEM-T2	22.66K	11.08M	Periodic	0.0240 ± 0.0022	0.0342 ± 0.0033
CEM-T4	22.66K	19.79M	Periodic	$\mathbf{0.0238 \pm 0.0029}$	$\mathbf{0.0340 \pm 0.0042}$
CEM-T8	22.66K	37.20M	Periodic	0.0242 ± 0.0037	0.0344 ± 0.0053
Plain	33.98K	6.76M	Rational Quadratic	0.0791 ± 0.0109	0.0885 ± 0.0108
Gated	33.94K	6.73M	Rational Quadratic	0.0599 ± 0.0070	0.0715 ± 0.0075
CEM-T1	22.66K	6.73M	Rational Quadratic	0.0578 ± 0.0072	0.0709 ± 0.0082
CEM-T2	22.66K	11.08M	Rational Quadratic	0.0409 ± 0.0039	$\mathbf{0.0596 \pm 0.0068}$
CEM-T4	22.66K	19.79M	Rational Quadratic	0.0417 ± 0.0033	0.0606 ± 0.0067
CEM-T8	22.66K	37.20M	Rational Quadratic	$\mathbf{0.0406 \pm 0.0035}$	0.0606 ± 0.0060
Plain	33.98K	6.76M	Non-Stationary	0.0385 ± 0.0052	0.0426 ± 0.0062
Gated	33.94K	6.73M	Non-Stationary	0.0315 ± 0.0030	0.0358 ± 0.0039
CEM-T1	22.66K	6.73M	Non-Stationary	0.0305 ± 0.0044	0.0356 ± 0.0051
CEM-T2	22.66K	11.08M	Non-Stationary	0.0257 ± 0.0027	0.0330 ± 0.0042
CEM-T4	22.66K	19.79M	Non-Stationary	0.0267 ± 0.0030	0.0330 ± 0.0043
CEM-T8	22.66K	37.20M	Non-Stationary	$\mathbf{0.0243 \pm 0.0030}$	$\mathbf{0.0317 \pm 0.0043}$

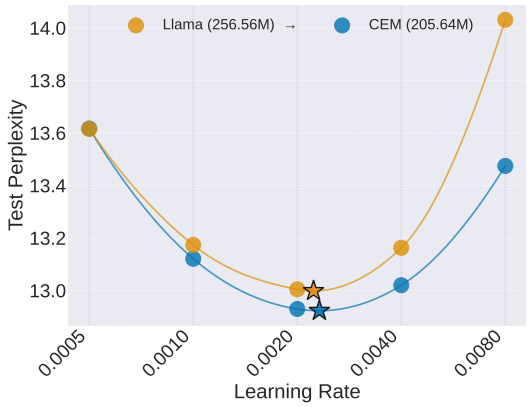

Figure 5: Additional results for optimal learning-rate estimation via Akima interpolation. Orange curves denote baseline Llama models; blue curves denote CEM models ($T = 2$) with both MHA and MLP replaced. Star markers denote interpolated optima from five data points. CEM models use roughly half the MHA parameters and two-third the MLP parameters per layer, with two additional layers to offset the larger parameter reduction for this size, and are trained under the same token budget (Chinchilla-optimal for Llama).