BOLTZMANN-MOE: ENERGY-BASED MIXTURE-OF-EXPERTS WITH NATIVE ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

In spite of the dominance of transformers in advanced AI models, our theoretical understanding of their inner workings from a statistical modeling perspective is still incomplete. In this work we show that transformers may be related to a very fundamental class of statistical models. Taking a first principles approach, we introduce the "Boltzmann-MoE", an energy-based model to learn both the overall data distribution, and which adapts itself to the context tokens. Boltzmann-MoE is a generalization of the Energy Transformer (ET) – a variant of looped transformers with some weight-tying – and naturally includes a mixture-of-experts with a native router. The Boltzmann-MoE energy is surprisingly simple and intuitive, consisting of two "free energies" (negative log-partition functions): the ET attention energy, which is a KDE on the context tokens, and the MoE energy is the log sum of Boltzmann weights measuring alignment of the token with memorized patterns. The forward pass of Boltzmann-MoE is a preconditioned gradient update, yielding softmax attention, softmax gating of the experts, and experts that combine a two-layer MLP with a gated linear unit. With a preconditioner split into dissipative and rotational parts, this update is a port-Hamiltonian flow whose learned rotational part adds deterministic exploration of the energy landscape while keeping the forward pass end-to-end differentiable. Our model is an alternative to standard transformers and MoE grounded in statistical physics. At 130M–315M active parameters on Nemotron-CC, hybrids with a standard transformer encoder followed by Boltzmann-MoE blocks, which clearly outperform pure energy models, trail an iso-active Switch-MoE by 0.1–1.3 points of average zero-shot accuracy and need about twice its tokens to reach the same loss. In deep models, distinct energy blocks learn update fields 3–4$\times$ more aligned than those of standard transformer layers.

1 INTRODUCTION

A classical view in generative modeling is that a model should learn an explicit probability density P over the data, and that generation is sampling from it. Yet, most modern approaches do not do this: diffusion and flow matching learn only the *score* $\nabla_{\boldsymbol{x}} \log P$, from which the density is recoverable only up to each component's normalization factor[1] (Song & Ermon, 2019; Wenliang & Kanagawa, 2020). Energy-based models keep the density, but at a price: one parametrises an energy E_θ and obtains $P_\theta(\boldsymbol{x}) = e^{-E_\theta(\boldsymbol{x})}/\mathcal{Z}_\theta$ with $\mathcal{Z}_\theta = \int e^{-E_\theta(\boldsymbol{x})} d\boldsymbol{x}$. The energy is a negative log-density only *up to* the additive $\log \mathcal{Z}_\theta$, which is independent of $\boldsymbol{x}$, dropping from gradients and making EBM hard to fit by maximum likelihood.

In this work we build an EBM whose energy is not the output of a network but a closed-form object (in contrast to Gladstone et al., 2026), required to 1) learn a generic data distribution and 2) adapt at inference time to the context. Such a model turns out to be closely related to the Energy Transformer (ET) (Hoover et al., 2023), whose energy is a sum of an attention term E^{AT} and a feedforward term E^{FF}. In ET, $E^{AT} = -\log \sum_B e^{\boldsymbol{q}^\top \boldsymbol{k}_B/\sqrt{d_h}}$ is already a **free energy** $F = -\log Z^{AT}$, whose partition function sums Boltzmann weights over a *finite* set of states, one per context token, and is

[1] one constant per connected component of the support, so the relative mass of well-separated modes is not identifiable from what they learn.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

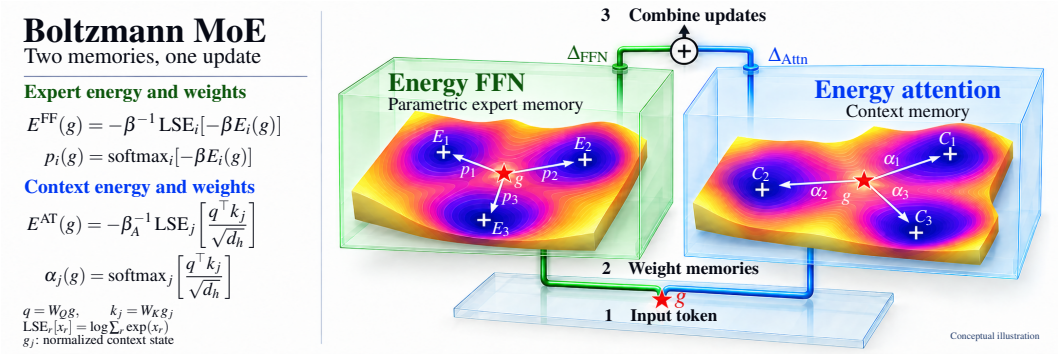

Figure 1: The Boltzmann-MoE is a type of parallel transformer, where the FF and attention outputs are added to each other. Similar to ET, the total energy is the sum $E = E^{FF} + E^{AT}$, and the forward pass is $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \Pi\nabla_{\boldsymbol{x}_i} E$. The key feature of Boltzmann-MoE is that both E^{FF} and E^{AT} are both free energies, estimating the log-density of the full data and of the context tokens, respectively.

therefore evaluated exactly. This is equivalent to a Kernel Density Estimation of the context token density.

In ET, E^{FF} is a Hopfield energy chosen so that its gradient is an MLP, and it carries no particular statistical meaning. We ask instead what energy one *should* write down to represent a generic distribution over the training data. While a single MLP-based energy should be able to approximate any function, we can take a free energy approach to estimate a distribution with many basins. So we let each cluster contribute its own Boltzmann weight and sum them. The result is again a partition function, $Z^{FF} = \sum_{i=1}^{K} e^{-\beta E_i(\boldsymbol{x})}$ with $E^{FF} = -\beta^{-1}\log Z^{FF}$, whose **states are the K experts of a mixture**. It is the same object as E^{AT}, differing only in what it sums over.

The goal of inference is to sample points where the probability density is high, conversely where the energy is low. Therefore, in Boltzmann-MoE we define the inference to be a series of gradient-based updates $\boldsymbol{x} \leftarrow \boldsymbol{x} - \nabla_{\boldsymbol{x}} E$. It is this gradient of E which yields the attention and the MoE modules; in practice the step is preconditioned by a learned Π with a positive semi-definite symmetric part. This makes inference a port-Hamiltonian flow: the symmetric part descends and the antisymmetric part circulates on level sets, the same non-reversible drift that accelerates Langevin samplers (Lelièvre et al., 2013), but without noise. The forward pass stays differentiable, so the model is trained end to end through what would otherwise be a non-differentiable sampling step (§3.7). The router then costs nothing to invent. Differentiating a log-sum-exp produces a softmax-weighted sum, so the descent direction of E^{FF} is $-\nabla_{\boldsymbol{x}} E^{FF} = -\sum_i p_i(\boldsymbol{x})\,\nabla_{\boldsymbol{x}} E_i(\boldsymbol{x})$ with $p_i = \mathrm{softmax}(-\beta E_i)$: a single derivative supplies both the mixture weights and the experts, p_i acting as the router and $\nabla_{\boldsymbol{x}} E_i$ as expert i's MLP. We call this the **Boltzmann-MoE**. And p_i is more than softmax-shaped: it is the *exact* posterior over the expert index, so the update is the posterior average of the per-expert gradients (§3.3). **A learned gate is a parametric approximation to a quantity this construction evaluates exactly.**

Energy blocks need not act on raw token embeddings. In our models a stack of standard transformer layers serves as an encoder that maps tokens into the representation space on which the energy is defined, and the energy blocks follow it; the statistical reading applies to the latter.

Contributions.

1. **A first-principles transformer model:** we build an EBM whose forward pass is a new energy transformer, and a port-Hamiltonian flow that serves as a differentiable stand-in for sampling.

2. **Boltzmann-MoE with native routing:** Our free energy approach naturally yields an FF module that is an MoE. Moreover, this MoE's router is native: it directly weights experts by the overlap between input and expert patterns instead of a learned gate.

3. **Cheap proxy routing:** Exact Boltzmann routing requires the costly step of computing all expert energies. We distill the router into a cheap proxy that nominates the top-k experts, so that only those are evaluated, recovering approximate FLOP parity with a Switch-MoE.
4. **Aligned updates:** distinct energy blocks learn 3–4$\times$ more aligned updates than transformer layers.
5. **Empirical validation:** We train models of 130M–315M active parameters on language modeling. Hybrid Boltzmann-MoE models trail an iso-active Switch-MoE by 0.1–1.3pp of zero-shot accuracy, and need roughly twice its tokens to reach the same loss.

2 Related work

Energy-based transformers. Energy-based learning replaces a normalized likelihood with a scalar energy whose minima are the model's preferred configurations (LeCun et al., 2006; Hopfield, 1982; Ackley et al., 1985). Modern Hopfield networks made the link to attention explicit(Krotov & Hopfield, 2016; Demircigil et al., 2017), with Ramsauer et al. (2021) showing that a particular energy over stored patterns yields softmax attention in its gradient update step. But perhaps the closest works to casting transformers as EBM is the Energy Transformer (Hoover et al., 2023) builds a layer whose forward pass is gradient descent on such an energy, and NRGPT (Dehmamy et al., 2025) extends it to autoregressive language modeling. Recent work pushes energies into inference-time computation (Gladstone et al., 2026; Wang & Du, 2026), but there the energy is the output of a deep network: every refinement step costs a full backward pass, and neither E nor ∇E has a closed form one can inspect. Our energy is a simple closed-form equation like ET, which makes the forward pass and the routing much simpler than one computed by differentiating a network. The price is a slight drop in expressiveness, and we claim none of the reasoning behaviour those works demonstrate.

Mixture-of-experts routing. Sparse mixtures route each token to a few of many experts through a learned gate (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Dai et al., 2024). Nothing in the task loss prefers balanced experts, so the gate needs an explicit remedy: an auxiliary load-balancing loss together with a z-loss (Fedus et al., 2022), or a per-expert bias nudged under `no_grad` toward under-loaded experts (DeepSeek-AI, 2024). Our routing weights are instead the Boltzmann posterior over per-expert energies, so no gate is learned, and balance is imposed by the so called "chemical potential" μ_k, solved by a short Sinkhorn iteration under `no_grad`. That places us closer to DeepSeek-V3's loss-free bias than to Switch's auxiliary objective, with μ solved rather than controlled, so it needs no gain and no clamp.

See Appendix C for extended related works, including the prior work that already contains parts of the reading in section 3: hyperspherical energy transformers with recurrent depth, the kernel-smoothing view of attention, and Sinkhorn-balanced routing.

3 Two Free Energies

As stated above, we want our EBM to contain two free energies, one modeling the density of the learned data (MoE energy), and one forming an adaptable density, estimating density of context tokens (attention energy). To generate high-likelihood samples, we make the inference a gradient-based update. If the block's output is $-\nabla_{\boldsymbol{h}} E$, then stacking blocks performs optimisation. So what should E look like? We build both terms as free energies, i.e. negative logs of unnormalised densities (log-partition functions).

3.1 Free energy

Suppose a system can occupy states $s = 1, \ldots, M$. In statistical mechanics, we parametrize the probability weight of the occurrence of state s using the Boltzmann weight, $e^{-\beta E_s(\boldsymbol{g})}$, when the representation is $\boldsymbol{g}$, where $E_s(\boldsymbol{g})$ is called the energy of the state and $\beta \equiv 1/\tau$ is called the inverse temperature. An example is a Gaussian mixture model, with $E_s(\boldsymbol{g}) = \|\boldsymbol{g} - \mu_s\|^2$ and $\beta^{-1} = 2\sigma_s^2$. The total weight is called *partition function*, $Z(\boldsymbol{g}) = \sum_s e^{-\beta E_s(\boldsymbol{g})}$, and the log-likelihood of $\boldsymbol{g}$ is

$\log p(\boldsymbol{g}) = -\beta F(\boldsymbol{g}) + \text{const}$ with

$$F(\boldsymbol{g}) \;=\; -\beta^{-1} \log \sum_{s=1}^{M} e^{-\beta E_s(\boldsymbol{g})}. \tag{1}$$

F is the *free energy*. The signs are the ones physics uses and we keep them throughout: an energy is a quantity to be *descended*, low energy carries large Boltzmann weight, and every exponent below is $-\beta$ times the energy.

Two properties will matter repeatedly. First, **a free energy interpolates between voting and winner-takes-all, and the interpolation is the entropy.** With $p_s \propto e^{-\beta E_s}$ the Boltzmann weights of equation 3 below, $\bar{E} = \sum_s p_s E_s$ the mean state energy and $\mathbb{S} = -\sum_s p_s \log p_s$ their entropy, equation 1 is exactly the thermodynamic decomposition

$$F = \bar{E} - \beta^{-1}\mathbb{S}, \qquad \text{hence} \qquad \min_s E_s - \beta^{-1} \log M \;\le\; F \;\le\; \min_s E_s, \tag{2}$$

since $0 \le \mathbb{S} \le \log M$ for M states. As $\beta \to \infty$, $F \to \min_s E_s$ (winner-takes-all); as $\beta \to 0$ the weights become uniform and the update equation 3 tends to the plain average of the per-state gradients. The temperature is a dial between consensus and specialisation.

Second, and this is the fact that generates a router out of nothing,

$$\nabla_{\boldsymbol{g}} F(\boldsymbol{g}) \;=\; \sum_{s=1}^{M} \underbrace{\frac{e^{-\beta E_s(\boldsymbol{g})}}{\sum_{s'} e^{-\beta E_{s'}(\boldsymbol{g})}}}_{p_s(\boldsymbol{g})} \nabla_{\boldsymbol{g}} E_s(\boldsymbol{g}), \qquad p_s = \mathrm{softmax}\big(-\beta E_s(\boldsymbol{g})\big). \tag{3}$$

The gradient of a free energy is automatically a softmax-weighted mixture of the individual gradients. We did not *choose* the softmax; it arose from differentiating a logsumexp.

3.2 Modeling distribution over the training data

We want our model to encode something about the data it was trained on. Concretely, we need a density $p(\boldsymbol{g})$ on representation space that is high where the training distribution puts mass. Since we do not know the shape of that distribution, we should pick the form that assumes as little as possible. This is the classical maximum-entropy (MAXENT) argument (Jaynes, 1957): among all densities matching a set of expected features, the one that adds no further structure is exponential in those features. Requiring a distribution and asking for minimal commitment beyond matched features *forces* the Boltzmann form. So this is not simply an analogy borrowed from physics; it is what "generic distribution" means once you insist on a density.

That still leaves the features. A single set of expected features only ever gives an exponential of a *sum* (a product of experts, one basin). A mixture needs a latent cluster label that is summed out. Let $z \in \{1, \dots, K\}$ be that label and $E_i(\boldsymbol{g})$ the energy of the state in which cluster i accounts for $\boldsymbol{g}$: minus the overlap between $\boldsymbol{g}$ and the patterns the cluster stores (§3.3).

Proposition 1 (Maximum-entropy mixture)**.** *Among all joint densities $p(\boldsymbol{g}, z)$ with prescribed $\mathbb{E}\big[\mathbf{1}[z{=}i]\, E_i(\boldsymbol{g})\big]$ and $\mathbb{E}\big[\mathbf{1}[z{=}i]\big]$, $i = 1, \dots, K$, the maximum-entropy one is*

$$p(\boldsymbol{g}, z) \;\propto\; e^{-\beta_z E_z(\boldsymbol{g}) - \beta_z \mu_z}. \tag{4}$$

If the clusters are exchangeable, $\beta_i \equiv \beta$, then the marginal and the posterior are

$$p(\boldsymbol{g}) \propto \sum_{z=1}^{K} e^{-\beta(E_z(\boldsymbol{g}) + \mu_z)} = Z^{\mathrm{FF}}(\boldsymbol{g}), \qquad p(z{=}i \mid \boldsymbol{g}) = \mathrm{softmax}_i\big(-\beta(E_i(\boldsymbol{g}) + \mu_i)\big). \tag{5}$$

Proof. Maximum entropy under linear moment constraints gives the exponential family in those features (Jaynes, 1957); the features are indicator-gated, so the exponent is $-\sum_i \mathbf{1}[z{=}i](\beta_i E_i + \beta_i \mu_i)$. Summing over z gives the marginal, and the ratio of joint to marginal the posterior, in which the normaliser cancels. □

The multipliers $\mu_i = -\beta^{-1} \log \pi_i$ encode the prior over clusters. We set $\mu_i = 0$ until §3.4, where they return as the chemical potential that balances the mixture.

Comparing with equation 1, this is exactly a partition function **whose states are the experts**: expert i is one configuration the system can be in, and Z sums over K of them. Its free energy is

$$\boxed{\textbf{MoE energy:} \quad E^{\mathrm{FF}}(\boldsymbol{g}) \;=\; -\beta^{-1}\log\sum_{i=1}^{K} e^{-\beta E_i(\boldsymbol{g})}.} \tag{6}$$

Each expert holds a set of patterns and reports, through minus its energy E_i, how well the current representation matches them. The free energy is then the soft minimum over "how badly does any of my memories explain this?".

3.3 Expert energies and their gradients

A Boltzmann-MoE feedforward block holds K energy experts. We use the W_1W_2 feedforward energy introduced by NRGPT (Dehmamy et al., 2025), one per expert,

$$E_k(\boldsymbol{g}) \;=\; -\tfrac{1}{\sqrt{I_e}}\,\varphi(\boldsymbol{W}_1^{(k)}\boldsymbol{g})^{\top}(\boldsymbol{W}_2^{(k)}\boldsymbol{g}), \qquad \boldsymbol{W}_1^{(k)},\boldsymbol{W}_2^{(k)} \in \mathbb{R}^{I_e\times d}, \tag{7}$$

with φ a GELU and I_e the expert width. Its gradient has two terms,

$$-\nabla_{\boldsymbol{g}}E_k = \tfrac{1}{\sqrt{I_e}}\Big[\underbrace{\boldsymbol{W}_2^{(k)\top}\varphi(\boldsymbol{W}_1^{(k)}\boldsymbol{g})}_{\text{2-layer MLP}} + \underbrace{\boldsymbol{W}_1^{(k)\top}\big(\varphi'(\boldsymbol{W}_1^{(k)}\boldsymbol{g})\odot \boldsymbol{W}_2^{(k)}\boldsymbol{g}\big)}_{\text{gated linear unit}}\Big], \tag{8}$$

and both of the feedforward layers in common use appear: the first is a standard two-layer MLP, and the second is a gated unit of the GLU family used in SwiGLU/GeGLU transformers (Shazeer, 2020), with gate $\varphi'(\boldsymbol{W}_1\boldsymbol{g})$ and value $\boldsymbol{W}_2\boldsymbol{g}$. Neither was put in by hand; both are the gradient of one scalar energy. A single-matrix Hopfield variant, $E_k = -\|\varphi(\boldsymbol{W}^{(k)}\boldsymbol{g})\|^2/I_e$ (Hoover et al., 2023), halves the parameters at equal width. The choice matters little: at 134M on 32B tokens of a math-bearing mixture, W_1W_2 experts reach 45.01 `Avg11` and loss 2.644 against 44.82 and 2.654 for Hopfield experts (the two arms also differ in router; table 10). Both forms, the activation convention and the gradient scaling are given in §A.1.

Sign convention. The state energy is **minus** the overlap, so the Boltzmann weight $e^{-\beta E_i}$ is large exactly when expert i matches $\boldsymbol{g}$ well, and $p(\boldsymbol{g}) \propto e^{-\beta E^{\mathrm{FF}}(\boldsymbol{g})} = Z^{\mathrm{FF}}(\boldsymbol{g})$ as in equation 5. E_i need not be bounded below; it stays finite because it is evaluated on a normalised representation. Flipping this sign inverts the router, which then favours the *worst*-matching experts.

The router was already there. Apply equation 3 to equation 6:

$$\begin{aligned} -\nabla_{\boldsymbol{g}}E^{\mathrm{FF}}(\boldsymbol{g}) &= -\sum_{i=1}^{K} p_i(\boldsymbol{g})\,\nabla_{\boldsymbol{g}}E_i(\boldsymbol{g}), \\ p_i(\boldsymbol{g}) &= \mathrm{softmax}\big(-\beta E_i(\boldsymbol{g})\big). \end{aligned} \tag{9}$$

The descent step is a *mixture of per-expert updates, weighted by how well each expert explains the current token*. This is a mixture-of-experts layer, and we did not design it: we asked for a generic distribution over the training data, and the gradient of the resulting free energy is an MoE with a built-in router. There is no gate to learn and no reason for the weights to be balanced by an auxiliary penalty – p_i is a posterior, not a free parameter. Indeed, by equation 4 and equation 5, $p_i(\boldsymbol{g}) = p(\boldsymbol{g},i)/\sum_z p(\boldsymbol{g},z) = p(z{=}i \mid \boldsymbol{g})$ **exactly**: the intractable normaliser of the joint is shared by the marginal and cancels in the ratio, so the posterior over the latent is available in closed form although the density is not. Equation equation 9 then reads $\nabla_{\boldsymbol{g}}\log p(\boldsymbol{g}) = \mathbb{E}_{p(z|\boldsymbol{g})}\big[\nabla_{\boldsymbol{g}}\log p(\boldsymbol{g},z)\big]$, which is Fisher's identity (Fisher, 1925) taken with respect to the data rather than the parameters: **the update is the posterior average of the per-expert updates**, and a learned gate is a parametric approximation to a quantity this construction evaluates exactly. What is more, $E_i(\boldsymbol{g})$ shares its first matrix product with $\nabla_{\boldsymbol{g}}E_i(\boldsymbol{g})$, so the routing signal costs no arithmetic beyond what the selected experts already need.

3.4 BALANCE IS A CHEMICAL POTENTIAL, NOT A PENALTY

The softmax of equation 9 is the unique minimiser of $\sum_i p_i E_i - \beta^{-1}\mathbb{H}[\boldsymbol{p}]$ over the simplex, and E^{FF} is its optimal value (the Gibbs variational principle). Nothing in it stops a few experts from absorbing most tokens. Adding the batch constraint $\frac{1}{N}\sum_n p_i(\boldsymbol{g}_n) = \frac{1}{K}$ introduces one Lagrange multiplier per expert, and the solution is the same softmax over shifted energies,

$$p_i(\boldsymbol{g}) = \mathrm{softmax}_i\big(-\beta\,(E_i(\boldsymbol{g}) + \mu_i)\big), \tag{10}$$

with μ_i the multiplier of Proposition 1, i.e. a **chemical potential**: a property of the expert (how crowded it is), where E_i is a property of the token (how well it fits). μ is a dual variable, not a parameter. It adds no gate, no loss term and, by the envelope theorem, no gradient pathway, so the construction needs neither a learned router nor an auxiliary balancing loss (cf. Fedus et al., 2022). Jointly over the batch, this is entropy-regularised optimal transport between tokens and experts, and we solve for μ with a few Sinkhorn iterations (Sinkhorn & Knopp, 1967; Cuturi, 2013) (§A.3, §A.4).

3.5 SPARSE BOLTZMANN ROUTING: PROXY AND SURROGATE

The posterior equation 10 needs all K energies, and since an energy shares most of its arithmetic with its expert's update (§4), exact top-k saves at most $2\times$. We therefore separate *selection* from *weighting*: a cheap score $\hat{E}_k$ nominates $p \geq k$ candidates, their exact energies pick the top k, $\mathcal{S}(\boldsymbol{g})$, and the weights are the exact posterior renormalised on them,

$$\hat{p}_i(\boldsymbol{g}) = e^{-\beta(E_i+\mu_i)} \Big/ \textstyle\sum_{j\in\mathcal{S}(\boldsymbol{g})} e^{-\beta(E_j+\mu_j)}, \qquad i \in \mathcal{S}(\boldsymbol{g}). \tag{11}$$

The weights remain energies, so the only approximation is the selection. The scorer is distilled with $\lambda\,\mathrm{KL}(p \,\|\, \mathrm{softmax}(-\hat{E}))$ on a detached input, and takes one of two forms. The **proxy** is a factorised, shrunken copy of each expert, a rank-r factorisation of its weights initialised by SVD and evaluated on m of its I_e units, costing $Kr(d{+}m)$ instead of KdI_e multiply-accumulates ($r{=}16$, $m{=}512$ here); it matches the exact top-1 choice 97.8% of the time at 0.21% of the cost. The **surrogate** is a linear or MLP head, architecturally a Switch gate, but trained only to imitate the posterior and used only to nominate; it costs 0.08pp `Avg11` against exact routing. Details in §H.

3.6 MODELING A DISTRIBUTION THAT RESHAPES ITSELF TO THE CONTEXT

The experts of §3.2 are fixed once training ends. They encode what is typical of the corpus, and nothing about the particular problem in front of the model. But the interesting behaviour of a transformer is that the *same* weights solve a task specified only in the prompt. So we need a second density – one that is rebuilt for every input.

What would such a thing look like? It cannot be a fixed parametric family, because its shape has to depend on the context, which is not known at training time. The standard answer in statistics for "I need a density and I refuse to commit to a shape" is to build it out of the samples themselves: kernel density estimation (Rosenblatt, 1956; Parzen, 1962), in which every observed point contributes a bump and the density is their sum.

Here the observations are the context tokens. Put one kernel at each of them, and the density of a candidate representation is how well it is explained by *some* token already in the context. There is one wrinkle: hidden states are RMSNormed or LayerNormed, so $\boldsymbol{g}$ lives on the unit sphere S^{d-1}, not in $\mathbb{R}^d$, and a Gaussian bump is the wrong object. The sphere's natural exponential-family analogue of the Gaussian is the von Mises–Fisher distribution (Mardia & Jupp, 2000; Banerjee et al., 2005), $p(\boldsymbol{g} \mid \boldsymbol{\mu}) \propto e^{\kappa\,\boldsymbol{\mu}^\top \boldsymbol{g}}$, whose log-density is *linear* in the inner product rather than quadratic in a distance.

Two roles now need distinguishing. A context token acts as a mode centre, while the token being refined acts as the test point, and there is no reason to compare them in the same basis. Project each into its own: keys $\boldsymbol{k}_B = \boldsymbol{W}_K \boldsymbol{g}_B$ for context tokens, queries $\boldsymbol{q}_A = \boldsymbol{W}_Q \boldsymbol{g}_A$ for the target. Placing a vMF kernel of concentration $\kappa = 1/\sqrt{d_h}$ at each key gives

$$p\big(\boldsymbol{q}_A \mid \mathcal{C}_A\big) \;\propto\; \sum_{B\prec A} \exp\Big(\frac{\boldsymbol{q}_A^\top \boldsymbol{k}_B}{\sqrt{d_h}}\Big), \qquad \mathcal{C}_A = \{x_B : B \prec A\}, \tag{12}$$

and its negative log-likelihood is the *attention free energy*

$$\textbf{Attention energy:} \quad E_A^{\mathrm{AT}}(\boldsymbol{h}) \;=\; -\log \sum_{B \prec A} \exp\Big(\frac{\boldsymbol{q}_A^\top \boldsymbol{k}_B}{\sqrt{d_h}}\Big) \tag{13}$$

at unit temperature, the scale being carried by $1/\sqrt{d_h}$ as in standard attention. On the sphere this is exact; for unnormalised queries a Gaussian kernel gives the same weights up to a per-query term $\|\boldsymbol{q}_A\|^2/2$, which is constant over B, leaves the softmax unchanged, and can be absorbed into the feedforward channel. Compare equation 13 with equation 6. **The same free energy, with the sum running over context tokens instead of experts.** Each context token is a state in the partition function, exactly as each expert was, and the sign convention carries over unchanged: the state energy of context token B is the negative overlap $E_B = -\boldsymbol{q}_A^\top \boldsymbol{k}_B/\sqrt{d_h}$, its Boltzmann weight e^{-E_B} is largest for the key that best matches the query, and $E_A^{\mathrm{AT}} = -\log Z_A^{\mathrm{AT}}$ exactly as E^{FF} is $-\beta^{-1}\log Z^{\mathrm{FF}}$. And by equation 3, the descent direction $-\nabla E_A^{\mathrm{AT}}$ is a mixture of the per-token gradients weighted by $p_B = \mathrm{softmax}_B(-E_B)$ – which is to say, *attention is the router of the context free energy*, in precisely the sense that Boltzmann routing is the router of the expert free energy. The softmax in attention has the same origin as the softmax in equation 9: both are $\partial(\text{log-sum-exp})$.

A context-built density has no fixed modes: a different prompt gives a different landscape, which is how a task never seen in training can carve out its own basins (§A.7).

3.7 Putting them together

The block's energy is the sum of the two free energies,

$$E(\boldsymbol{h}) \;=\; \underbrace{\sum_A E_A^{\mathrm{AT}}(\boldsymbol{h})}_{\text{adaptive, built from the context}} \;+\; \underbrace{\sum_A E^{\mathrm{FF}}(\boldsymbol{g}_A)}_{\text{fixed, learned from the corpus}}, \tag{14}$$

and the forward pass is $\boldsymbol{h} \leftarrow \boldsymbol{h} - \Pi\,\nabla_{\boldsymbol{h}} E$ with a learned $\Pi = \boldsymbol{S}\boldsymbol{S}^\top + (\boldsymbol{A} - \boldsymbol{A}^\top)$. The symmetric part is positive semi-definite by construction, so to first order the step never increases the energy, $\delta E = -\eta\,\nabla E^\top \Pi\,\nabla E = -\eta\,\|\boldsymbol{S}^\top \nabla E\|^2 \le 0$, while the antisymmetric part contributes nothing to δE and moves the state *along* level sets. With the exact gradient $\nabla_{\boldsymbol{x}} E$ this is a **port-Hamiltonian** flow (van der Schaft, 2006): a dissipative part $\boldsymbol{S}\boldsymbol{S}^\top$ that descends and a conservative part $\boldsymbol{A} - \boldsymbol{A}^\top$ that circulates on level sets. In stochastic dynamics, adding exactly such an antisymmetric drift to Langevin dynamics, $d\boldsymbol{x} = -(\boldsymbol{I} + \boldsymbol{J})\nabla U\,dt + \sqrt{2/\beta}\,d\mathbf{W}$ with $\boldsymbol{J}^\top = -\boldsymbol{J}$, leaves $e^{-\beta U}$ invariant while accelerating convergence and the visiting of modes (Hwang et al., 1993; 2005; Lelièvre et al., 2013; Rey-Bellet & Spiliopoulos, 2015; Ma et al., 2015). Our block keeps the circulation and drops the noise, and needs no auxiliary momentum as in Hamiltonian Monte Carlo (Neal, 2011; Robnik et al., 2023). Because Π is learned, it is fit to the few steps of the network's depth, and the whole T-step trajectory stays a smooth function of the parameters: a differentiable stand-in for sampling, trained end to end through the very refinement that would otherwise require a non-differentiable sampler. This is consistent with the finding of Dehmamy et al. (2025) that exploration beats pure descent; we do not claim the flow samples p. A learned Π beats $\Pi = I$ by 0.05 nats of loss in a small pure-energy model (§E.3, one seed). An unconstrained Π trains about as well (§E.2), but it carries no descent guarantee, so all models reported here use the constrained form above. The guarantee is exact for the gradient with respect to the residual state $\boldsymbol{x}$. Energy Transformer and NRGPT instead differentiate with respect to $\boldsymbol{g} = \mathrm{RMSNorm}(\boldsymbol{x})$; the chain rule adds the Jacobian $\nabla_{\boldsymbol{x}} E = \frac{\sqrt{d}}{\|\boldsymbol{x}\|}\big(\boldsymbol{I} - \boldsymbol{g}\boldsymbol{g}^\top/d\big)\nabla_{\boldsymbol{g}} E$, a Rayleigh (tangent) projection onto the sphere (§A.8). Our code supports both; we saw no significant difference in quality and report the $\nabla_{\boldsymbol{g}}$ form. The two terms are *independent* free energies: neither appears inside the other's partition function, so each contributes its own additive term and its own softmax. Their division of labour is the architecture's content. The expert term anchors representations to the statistics of the training data – what is generally true. The attention term deforms the landscape to the case at hand – what is true right now. Iterating the block lets the two negotiate: context reshapes the basins, memory keeps the trajectory on the data manifold.

Table 1: **Development sweep on nemotron-cc** (single Nemotron-CC web shard, cosine LR $= 10^{-3}$; token budget per tier in the tier headers). Energy rows use exact all-K routing at evaluation. `Avg11` is the eleven-task zero-shot accuracy mean in percentage points (§F.2). PPL is wikitext word-level perplexity (lower better). † marks baselines. Param counts from `audit_config`; Boltzmann routing shares 75% of its GEMMs with the expert forward (§B).

Scale	Arm	Arch	K/k	Total	Active	FLOPwt	lm_loss	Avg11	PPL
d=768, ∼130M active, 4.0B tokens									
Energy	Boltz-MoE rec	6G1x6E	8/2	163M	132M	194M	3.283	43.96	45.4
	Boltz-MoE deep	6S6E	4/2	166M	132M	132M	3.287	43.47	46.7
Baseline	Switch MoE †	6G1x6S	8/2	163M	132M	196M	3.216	**44.72**	42.1
	12-layer dense †	12G	–	162M	162M	162M	**3.164**	43.95	**39.2**
d=1024, ∼225M active, 7.6B tokens									
Energy	Boltz-MoE rec	6G1x6E	8/2	352M	225M	454M	3.101	44.73	36.2
	Boltz-MoE deep	6S6E	4/2	309M	225M	225M	3.151	45.07	35.1
	12-NRGPT (no MoE)	12N	–	318M	318M	318M	3.288	42.77	47.5
Baseline	12S all-Switch †	12S	4/2	297M	225M	225M	3.048	45.51	33.3
	Switch MoE †	6G1x6S	8/2	352M	225M	458M	**2.988**	**46.01**	**31.4**
	12-layer dense †	12G	–	255M	255M	255M	3.057	45.56	34.4
d=1280, ∼315M active, 5.6B tokens									
Energy	Boltz-MoE deep	6S6E	4/2	442M	315M	315M	3.057	46.17	34.3
	Boltz-MoE rec	6G1x6E	8/2	502M	314M	655M	3.098	45.45	34.3
Baseline	Switch MoE †	6G1x6S	8/2	503M	315M	661M	**2.954**	**46.89**	**30.0**
	12-layer dense †	12G	–	363M	363M	363M	3.016	46.16	32.9
d=1280, ∼315M active, 9.5B tokens (1.5× Chinchilla)									
Energy	Boltz-MoE deep	6S6E	4/2	442M	315M	315M	2.981	46.71	33.1
Baseline	12S all-Switch †	12S	4/2	423M	315M	315M	**2.948**	**46.81**	**29.7**

4 EXPERIMENTS

Setup. We train from scratch on a single 10.4B-token shard of the high-quality web subset of Nemotron-CC (Su et al., 2025), packed to sequence length 4096 with the Granite-4.0 tokeniser. All arms use AdamW ($\beta = (0.9, 0.95)$, weight decay 0.1) at peak learning rate 10^{-3} with 1000 warmup steps and cosine decay, bf16 and FSDP-2 (§F.1). We compare at three widths, $d \in \{768, 1024, 1280\}$, with 4.0B, 7.6B and 5.6B tokens respectively, plus a 9.5B-token (1.5× Chinchilla) run at d=1280. Within a width every arm is matched on *active* parameters and sees the same tokens.

Architectures. All models have twelve layers. `G` is a dense transformer layer, `S` a transformer layer with a Switch-style top-k MoE (Fedus et al., 2022), `E` a Boltzmann-MoE block (energy attention plus Boltzmann-MoE, W_1W_2 experts, K=4, top-2, Sinkhorn balance, cheap proxy router), and `N` an energy block without a mixture. `6S6E` is six `S` layers acting as the encoder followed by six distinct `E` blocks; `6G1x6E` is six `G` layers followed by *one* `E` block applied six times. Baselines are the dense `12G`, the all-Switch `12S`, and `6G1x6S`, the recurrent Switch counterpart of `6G1x6E`. TOTAL and ACTIVE count all and per-token parameters; **FLOPwt** $= \sum_b N_b \,\mathrm{ACTIVE}_b$ weights each block by its applications per token N_b (6 for a recurrent block), so it tracks matrix-multiply FLOPs per token (excluding attention's quadratic term).

Metrics. `Avg11` is the unweighted mean zero-shot accuracy over eleven tasks, in percentage points (§F.2); we also report `lm_loss` (nats) and WikiText word perplexity, both lower-is-better. Energy rows in table 1 use exact routing at evaluation; replicates in table 7 use the deployed proxy router. Except for table 7, **every arm is a single seed**, so sub-1pp `Avg11` margins should be read with that in mind.

Boltzmann-MoE trails Switch at matched active parameters, and the gap narrows with scale. Against the recurrent Switch baseline `6G1x6S`, the deep `6S6E` arm is behind by 1.25, 0.94 and 0.72pp `Avg11` at d=768, 1024 and 1280, while spending 1.5–2.1× *fewer* FLOPs per token (FLOPwt 132M vs. 196M up to 315M vs. 661M). The fairest comparison is `12S`, which matches `6S6E` on depth, tokens, active parameters *and* FLOPs: the gap is 0.44pp at d=1024 and 0.10pp at d=1280 with 1.5× Chinchilla tokens (46.71 vs. 46.81). The continuous metrics still favour Switch there (0.033 nats of loss, 3.4 PPL), so we read this as *competitive*, not as parity.

Against a dense transformer. At d=1280 `6S6E` ties `12G` on `Avg11` (46.17 vs. 46.16, 13% fewer active parameters, higher loss), and dropping both encoder and mixture (`12N`) costs 2.3pp (§E.7).

Depth over recurrence, and token efficiency. Six distinct energy blocks (`6S6E`) beat one block applied six times (`6G1x6E`) at d=1024 and 1280 (+0.34, +0.72pp) at about half the FLOPs, though not at d=768 (−0.49pp; §E.5). On the loss curves Switch reaches the energy model's final loss with about 53% of its tokens, a roughly constant factor over training (§D.1).

Seeds. Two more `6S6E` seeds at d=768 agree within 0.48pp, against a 1.99pp gap to Switch (§E.8).

Ablations. Code lineage, the constraint on Π, expert width at fixed active parameters and a 200M learning-rate and router sweep are in §E and table 8; only expert width (too-narrow experts cost 1.3pp) and the distillation weight move the results.

Sparse routing is nearly free for hybrids, not for pure energy stacks. Evaluating the same checkpoint with deployed sparse routing (proxy or surrogate nominates, exact energies weight) instead of exact all-K routing costs −0.13pp `Avg11` on average over eight hybrid arms from 134M to 1B, within the 0.37pp spread of repeated evaluations; a pure recurrent energy stack, with no layers outside the mixture, instead loses 7.96pp (table 10; these arms trained on the earlier math-bearing mixture).

Do successive energy blocks follow related updates? A deep energy model has a different energy in every block, so the descent picture holds globally only if consecutive blocks produce related updates. Because the dynamics live on a sphere, where one fixed flow gives successive steps that rotate relative to each other, we compare blocks *functionally*: feed the same states to blocks l and $l{+}1$ and take the cosine of their updates, in excess of a token-shuffled baseline. Given the same encoded input (the output of the six-layer encoder), energy blocks are 3–4× more mutually aligned than softmax blocks in both deep hybrids we measured (excess cosine 0.16 vs. 0.04 at 4B tokens, 0.10 vs. 0.04 at 32B; §E.6). The alignment is learned: at random initialisation every excess cosine is below 0.002. Different energies, then, but related update fields.

Boltzmann routing has zero marginal cost. For a selected W_1W_2 expert the routing energy reuses three of the four GEMMs that produce the expert's update, so routing adds one dot product and a softmax, whereas a Switch gate is a separate $d \to K$ map (§B).

5 Conclusion

We built a transformer block from two free energies: a kernel density estimate over the context, whose gradient is attention, and a maximum-entropy mixture over the training data, whose gradient is a mixture-of-experts layer routed by an exact posterior. Load balance is a chemical potential, solved rather than penalised, and sparse routing changes which experts are evaluated, not how they are weighted. Empirically the construction is competitive but not superior: at 130M–315M active parameters it trails an iso-active Switch-MoE by 0.1–1.3pp, needs about twice the tokens for the same loss, and works best behind a standard transformer encoder (single seeds, under 10B tokens).

Limitations and outlook. Each deep energy block has its own energy; their learned alignment suggests related energies, but this remains to be made precise. The explicit energy also permits annealing, posterior sampling of experts or Langevin dynamics, none of which we have tested.

AI USE STATEMENT

In this work, we used generative AI tools (large language model coding assistants) to write and debug training, evaluation and analysis code, to manage and monitor cluster jobs, to audit configurations and results, and to draft and edit parts of the text. We have not used generative AI tools to generate research ideas or to fabricate or alter experimental results; every number in the paper is produced by our training and evaluation pipeline and traced to stored logs. We have reviewed all AI-assisted work: code was tested and its outputs checked against independent measurements, and AI-edited text was read and revised by the authors. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies the architecture of language models at small scale (130M–500M parameters) trained on publicly available web text (Nemotron-CC) and mathematics corpora. It involves no human subjects, no personal data collection and no new dataset release. Models of this size have limited capabilities, and we do not release them as products; they inherit the biases of their web-scale training data, which we did not audit. We are not aware of conflicts of interest beyond the authors' affiliations.

REPRODUCIBILITY STATEMENT

The model is defined in §3, with the expert forms, routing, chemical potential and gradient scaling in §A and the proof of Proposition 1 in the text. Architectures, parameter counts and token budgets for every arm are given in table 1 and §4; the data, optimiser and schedule in §F.1; and the evaluation protocol, including the `Avg11` task list and scoring, in §F.2. Parameter counts are computed from the built models rather than by hand. We will release the training code, configuration files and evaluation scripts.

REFERENCES

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

Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, 2005.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew Johnson, Katie Millican, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, pp. 4057–4086. PMLR, 2022.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, 2013.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.70. arXiv:2401.06066.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

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

Mete Demircigil, Judith Heusel, Matthias Löwe, Sven Upgang, and Franck Vermet. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168(2):288–299, 2017.

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.

Ronald A. Fisher. Theory of statistical estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 22(5):700–725, 1925.

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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ZBj3Qp1bYg.

Geoffrey E Hinton and Terrence J Sejnowski. Learning and relearning in Boltzmann machines. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pp. 282–317. MIT Press, 1986.

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed J Zaki, and Dmitry Krotov. Energy Transformer. In *Advances in Neural Information Processing Systems*, volume 36, 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. doi: 10.1073/pnas.79.8.2554.

Yunzhe Hu, Difan Zou, and Dong Xu. Hyper-set: Designing transformers via hyperspherical energy minimization. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. URL https://openreview.net/forum?id=FinhjyDgYA.

Chii-Ruey Hwang, Shu-Yin Hwang-Ma, and Shuenn-Jyi Sheu. Accelerating Gaussian diffusions. *The Annals of Applied Probability*, 3(3):897–913, 1993.

Chii-Ruey Hwang, Shu-Yin Hwang-Ma, and Shuenn-Jyi Sheu. Accelerating diffusions. *The Annals of Applied Probability*, 15(2):1433–1444, 2005.

Edwin T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, 1957.

Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. In *Advances in Neural Information Processing Systems*, pp. 1172–1180, 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.

Tony Lelièvre, Francis Nier, and Grigorios A. Pavliotis. Optimal non-reversible linear drift for the convergence to equilibrium of a diffusion. *Journal of Statistical Physics*, 152(2):237–274, 2013.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6265–6274. PMLR, 2021.

Yi-An Ma, Tianqi Chen, and Emily B Fox. A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/9a4400501febb2a95e79248486a5f6d3-Abstract.html.

Kanti V. Mardia and Peter E. Jupp. *Directional Statistics*. John Wiley & Sons, Chichester, 2000.

Radford M Neal. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011. URL https://arxiv.org/abs/1206.1901.

Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *International Conference on Learning Representations (ICLR)*, 2021.

Luc Rey-Bellet and Konstantinos Spiliopoulos. Irreversible Langevin samplers and variance reduction: a large deviations approach. *Nonlinearity*, 28(7):2081–2103, 2015.

Jakob Robnik, G Bruno De Luca, Eva Silverstein, and Uroš Seljak. Microcanonical Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 24(311):1–34, 2023. URL https://jmlr.org/papers/v24/22-1450.html.

Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

Tim Salimans and Jonathan Ho. Should ebms model the energy or the score? In *ICLR Workshop on Energy Based Models*, 2021.

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.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC: Transforming Common Crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2459–2475, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.123. URL https://aclanthology.org/2025.acl-long.123/.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4344–4353, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1443.

Arjan van der Schaft. Port-Hamiltonian systems: an introductory survey. In *Proceedings of the International Congress of Mathematicians*, volume 3, pp. 1339–1365, 2006.

Runqian Wang and Yilun Du. Equilibrium matching: Generative modeling with implicit energy-based models. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.02300.

Li K Wenliang and Heishiro Kanagawa. Blindness of score-based methods to isolated components and mixing proportions. *arXiv preprint arXiv:2008.10087*, 2020.

Man Yung Wong. Affinity is not enough: Recovering the free energy principle in mixture-of-experts. *arXiv preprint arXiv:2605.00604*, 2026. Applies Friston's free-energy principle as motivation for three additive gate modifications (LIF membrane potential, precision weighting, anticipatory routing); retains the learned gate, no partition function, experiments at 4 experts.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 7103–7114, 2022.

A THEORY: EXPERT ENERGIES, ROUTING, AND THE CHEMICAL POTENTIAL

A.1 EXPERT ENERGIES AND THEIR GRADIENTS

A Boltzmann-MoE feedforward block holds K energy experts. Each expert reports an *overlap* $a_k(h)$ – how well h matches the patterns it stores – and its state energy is the negative overlap, $E_k = -a_k$, exactly as in §3.2. The main text uses only E_k; this appendix also names a_k, because the non-negative overlap is the quantity the implementation actually stores. Both expert forms below are quoted as overlaps, so that one routing rule covers them. The block's feedforward branch returns the *descent direction* of the mixture free energy, ffwd_out $= -\nabla_h E^{\mathrm{FF}} = \sum_k p_k \nabla_h a_k$, and the attention branch is returned in the same convention, both entering the residual update $h \leftarrow h - \Pi\big(\text{attn_out} + s_{\text{ff}}\,\text{ffwd_out}\big)$ through the projection Π, which is a learned unconstrained map in the arms reported here and therefore carries the overall sign of the step as well as its geometry. We use two expert parametrisations.

W_1W_2 experts (overlap of either sign).

$$a_k(h) \;=\; \tfrac{1}{\sqrt{I_e}}\,\varphi(W_1^{(k)}h)^\top(W_2^{(k)}h), \qquad W_1^{(k)}, W_2^{(k)} \in \mathbb{R}^{I_e\times d}, \tag{15}$$

so that $E_k = -a_k$ is unbounded in both directions, with gradient

$$\nabla_h a_k \;=\; \tfrac{1}{\sqrt{I_e}}\Big[\varphi(W_1^{(k)}h)^\top W_2^{(k)} + \big(\varphi'(W_1^{(k)}h)\odot(W_2^{(k)}h)\big)^\top W_1^{(k)}\Big] \;\equiv\; \mathrm{t}_1^{(k)} + \mathrm{t}_2^{(k)}. \tag{16}$$

Hopfield experts (non-negative overlap).

$$a_k(h) \;=\; \tfrac{1}{I_e}\left\|\varphi(W^{(k)}h)\right\|^2 \;\geq\; 0, \qquad \nabla_h a_k \;=\; \tfrac{c}{I_e}\,W^{(k)\top}\big(\varphi(W^{(k)}h)\odot\varphi'(W^{(k)}h)\big). \tag{17}$$

A single weight matrix per expert, so half the parameters of a W_1W_2 expert at equal I_e. Here $a_k \geq 0$, so $E_k = -a_k \leq 0$: the state energy is bounded above and, as in §3.2, not below. The constant c is discussed in §A.6.

Activation convention. φ is the exact (erf-based) GELU, but φ' is by default the *half-magnitude* sigmoid surrogate $\varphi'(z) = \frac{1}{2}\sigma(\sqrt{2/\pi}\,z)$, which is uniformly half the true GELU derivative. This is inherited from the pre-existing Energy-MLP code path and is retained because it is what the hyperparameters were tuned against. Replacing it with the self-consistent tanh-GELU pair *hurt*: -2.20pp average accuracy and $+3.42$ WikiText perplexity at 145M over 7.86B tokens. We therefore keep the surrogate and note that any factor absorbed into φ' is compensated by the constant c.

The projection is two maps, and it is unconstrained. §3.7 writes a single projection Π applied to $-\nabla_{\boldsymbol{h}}E$. The implementation applies *two*: separate learned maps for the attention and mixture channels, with the mixture channel's scalar gain frozen at 1. Both are ordinary unconstrained linear maps (`energy_proj_type: dual_unconstrained`) in every arm reported here. Two consequences worth stating plainly. First, the block realises the descent *direction* composed with a learned transform, not a descent step: only a positive-definite constraint would give the latter, and the codebase's `psd_anti` variant supplies exactly that. Second, with two independent maps the two branches are no longer required to contribute in a common metric, so equation 14 should be read as the energy whose gradient *supplies* the two update directions rather than as a potential the block provably descends. We report the unconstrained form because that is what was run, and because a matched comparison finds nothing to choose between the two. Three otherwise identical 134M arms, differing only in `energy_proj_type` and trained for 2500 steps, give a median `lm_loss` over their final 500 steps of 3.6081 (`unconstrained`), 3.6096 (`dual_unconstrained`) and 3.6177 (`psd_anti`). Bootstrap 95% intervals on those medians are ± 0.010 and overlap throughout, against a spread across arms of 0.0096, so no difference is resolvable; step times agreed to within 0.3%, so the positive-definite constraint is computationally free. At this scale it therefore costs nothing and buys nothing measurable. We state the obvious limitation: this is a 134M probe at a short budget, so it bounds the size of any effect rather than excluding one, and it does not license the reverse claim that the constrained form is preferable.

A.2 ROUTING

Routing weights are the Boltzmann probabilities of the expert state energies,

$$p_k(h) \;=\; \mathrm{softmax}_k\big(\beta\, a_k(h)\big), \qquad -\nabla_h E^{\mathrm{FF}} \;=\; \textstyle\sum_k p_k(h)\, \nabla_h a_k(h), \tag{18}$$

with the overlap $a_k = -E_k$ for *both* expert forms, which is equation 9 verbatim. Here β is the inverse temperature of §3.1; the configuration field is the temperature itself, `temperature` $= \tau = 1/\beta$, so every numerical value quoted in this appendix ($\tau = 1$, $\tau = 0.35$, and so on) is a temperature and not a β. The Boltzmann weight $e^{-\beta E_k}$ is largest for the expert of *highest overlap* with the input, so that is the expert the router selects. There is **no learned gate and no load-balancing loss**: the energy landscape itself ranks the experts.[2]

The router is arithmetically free. This is the central efficiency property and it holds for both expert forms. For Hopfield experts, a_k depends on h only through $W^{(k)}h$, which is the first of the two matrix products the gradient needs anyway. For W_1W_2 experts, $a_k = h^\top \mathrm{t}_1^{(k)}$, and $\mathrm{t}_1^{(k)}$ is itself one of the two terms of $\nabla_h a_k$. In both cases the router consumes exactly half of the block's matrix-multiply work, and that half is fully reused by the experts it selects. The only waste is router work spent on the $K - k$ *rejected* experts. Consequently both forms have identical relative cost under truncation:

$$\frac{\text{cost(top-}k\text{)}}{\text{cost(dense)}} = \begin{cases} \frac{1}{2} + \frac{1}{2}\,\frac{k}{K}, & \text{exact energy router,} \\ \frac{k}{K}, & \text{free proxy router (§H.1).} \end{cases} \tag{19}$$

Truncation does not renormalise. With `top_k` set, we compute the full softmax and zero the non-selected entries, so $\sum_k p_k < 1$ by design; this avoids abrupt weight redistribution at routing boundaries. It also means a soft-trained model loses output magnitude when truncated post hoc.

Routing-scale normalisation, and an ablation that we lost. The $1/\sqrt{I_e}$ factor in the W_1W_2 energy is required. Without it $|E_k|$ grows with I_e, the softmax saturates, and training shows loss spikes at steps ∼1000–4000. The Hopfield form instead divides by I_e, which is the correct $O(1)$ scale for a sum of I_e squares, but which leaves the energies numerically tiny relative to $\tau = 1$.

We first addressed this with per-token standardisation of the routing logits across experts, $E_k \mapsto (E_k - \bar{E})/\mathrm{std}(E)$, on the reasoning that it makes routing invariant to drift in the energy magnitude as the weights shrink under weight decay. **We no longer recommend it, on two independent grounds.**

Empirically we have two matched pairs, each differing only in this field, and **they split by metric**. Below, `Avg11` is the eleven-task accuracy mean in percentage points and MMLU an accuracy percentage, both *higher*-better; WikiPPL is word-level perplexity, *lower*-better. *orc* evaluates all K expert energies and *sps* is the deployed proxy-sparse path. All four arms are 134M at 32.0B tokens on the mixture of §F.1, so the rows are directly comparable, and no arm has a repeated seed.

arm	`routing_norm`	Avg11^{orc}	Avg11^{sps}	ppl^{orc}	ppl^{sps}	MMLU
hybrid `6G1x6E`	`zscore`	44.82	44.67	**41.06**	**41.82**	**25.57**
hybrid `6G1x6E`	`none`	**44.92**	**45.43**	42.36	42.79	24.74
block-pos `5G1x6E1G`	`zscore`	43.45	44.34	**40.98**	**41.79**	25.20
block-pos `5G1x6E1G`	`none`	**44.94**	**45.12**	41.99	42.59	**25.63**

`none` wins `Avg11` in both pairs and by the larger margin on the deployed path (+0.76 and +0.78pp), and it loses perplexity in both pairs (+1.30 and +1.01 oracle). **We therefore do not**

[2] The two forms differ in which member of the pair (a_k, E_k) the implementation stores – the Hopfield expert returns $+a_k$, the composable W_1W_2 expert returns $-a_k = E_k$ – so the sign flag that turns a stored value into a logit has to be set per expert kind. The kind-based defaults it shipped with are inverted for both kinds: they give a logit of $+E_k = -a_k$, i.e. the *worst*-matching experts, a defect we measured and corrected. Every arm reported here routes on the overlap a_k: the Hopfield arms because the flag is overridden in their configurations, the W_1W_2 arms because they use the legacy mixture class, which scores $+a_k$ directly.

claim a quality win for either setting: the accuracy mean and the perplexity disagree, perplexity is the lower-variance measurement, and both `Avg11` margins are within what one seed can resolve. What the pairs do establish is that the standardisation is *not* required, and an earlier reading of it as a fragility of the Hopfield form is withdrawn.

Theoretically, standardisation costs the free-energy reading, and the cost is quantifiable. Because $\mathrm{std}_k E_k$ is itself a function of the token, standardising is a per-token *adaptive temperature* $\tilde{\beta}(\boldsymbol{g}) = \beta/\mathrm{std}_k E_k(\boldsymbol{g})$. The routing weights remain an exact Boltzmann posterior at that temperature, but the returned vector is no longer the gradient of the corresponding free energy:

$$-\sum_k p_k \nabla_{\boldsymbol{g}} E_k \;=\; -\nabla_{\boldsymbol{g}} \tilde{F} \;+\; \tilde{\beta}^{-2}\,\mathbb{H}[\boldsymbol{p}]\;\nabla_{\boldsymbol{g}}\tilde{\beta}, \qquad \tilde{F} = -\tilde{\beta}^{-1}\log\textstyle\sum_k e^{-\tilde{\beta}E_k}, \tag{20}$$

an identity, not an approximation (verified in float64). The implementation computes the first term and drops the second, which is the routing entropy times the gradient of the standardising scale – exactly zero under hard routing or a locally flat spread of energies, and *largest* in the high-entropy regime. So the arms with the healthiest load balance are the ones for which the dropped term is biggest. Setting `routing_norm: none` removes the term, and the temperature τ then has to be set against the energy scale directly.

A.3 Balance as a chemical potential: full derivation

Equation equation 9 hands us routing weights but does not say why *those* weights, nor how to stop a few experts absorbing most of the traffic. Both points have one variational resolution.

The softmax is the entropy-regularised argmax. Ask for the weights that put as much mass as possible on the experts that explain the token, while committing as little as possible to any one of them:

$$\min_{\boldsymbol{p}} \sum_{i=1}^{K} p_i\, E_i(\boldsymbol{g}) \;-\; \beta^{-1}\mathbb{H}[\boldsymbol{p}] \quad \text{s.t.} \quad \textstyle\sum_i p_i = 1,\;\; \boldsymbol{p} \geq 0, \qquad \mathbb{H}[\boldsymbol{p}] = -\textstyle\sum_i p_i \log p_i. \tag{21}$$

With a multiplier λ for the normalisation, $\Lambda = \sum_i p_i E_i + \beta^{-1}\sum_i p_i \log p_i - \lambda\big(\sum_i p_i - 1\big)$, and $\partial\Lambda/\partial p_i = 0$ gives

$$E_i + \beta^{-1}\log p_i + \beta^{-1} - \lambda \;=\; 0 \quad\Longrightarrow\quad \log p_i \;=\; -\beta E_i - 1 + \beta\lambda \quad\Longrightarrow\quad p_i \;\propto\; e^{-\beta E_i}, \tag{22}$$

the non-negativity constraints inactive because $\partial\mathbb{H}/\partial p_i \to +\infty$ as $p_i \to 0$. So the softmax of equation 9 is not *a* routing rule: it is the unique solution of equation 21, whose optimal value is exactly $E^{\mathrm{FF}} = F$ of equation 1. This is the Gibbs variational principle, and equation 2 is its statement at the optimum: the objective is $U - \beta^{-1}\mathbb{S}$ and its minimum is the free energy. Free energy and softmax are one optimisation seen twice, and β^{-1} is the price of entropy. Equivalently $-\beta F$ is the convex conjugate of the negative entropy on the simplex, with $\boldsymbol{p}^\star$ its gradient – so equation 3 is the envelope theorem for equation 21, the same theorem that lets us differentiate through the balanced solution below.

Load balance is a second constraint on that problem. Now ask for what a mixture actually needs. Over a batch of N tokens, require that the batch-marginal occupancy of every expert be its fair share,

$$\frac{1}{N}\sum_{n=1}^{N} p_i(\boldsymbol{g}_n) \;=\; \frac{1}{K}, \qquad i = 1, \ldots, K. \tag{23}$$

The constraint couples the tokens, so equation 21 must be solved jointly over the batch, subject to per-token normalisation *and* equation 23. A multiplier μ_i on the i-th constraint subtracts μ_i from the i-th stationarity condition of equation 22, so the solution is the same softmax over shifted energies,

$$\boxed{\;p_i(\boldsymbol{g}) \;=\; \mathrm{softmax}_i\Big(-\beta\,\big(E_i(\boldsymbol{g}) + \mu_i\big)\Big)\;} \tag{24}$$

One scalar per expert, shared by every token in the batch, and nothing else changes. Two statements about μ need separating. *Given* μ, it is constant across tokens, so equation 3 applies to equation 10

verbatim and the block still descends a free energy, the *tilted* $E_\mu^{\mathrm{FF}} = -\beta^{-1}\log\sum_i e^{-\beta(E_i+\mu_i)}$. But μ is *solved* from the batch, so it is a function of every $\boldsymbol{g}_n$ in it, and what licenses dropping its implicit derivative is not independence but the **envelope theorem**: at a feasible point $\partial\Lambda/\partial\mu_i = -\big(\frac{1}{N}\sum_n p_i(\boldsymbol{g}_n) - \frac{1}{K}\big) = 0$, so the total derivative of the optimal value equals the partial derivative at fixed μ. The omitted term is therefore first order in the residual of equation 23 – which an exact solve leaves at zero and a truncated one does not (§A.4).

μ_i is a chemical potential. The variable conjugate to a particle number is a chemical potential, and equation 10 is the grand-canonical weight $e^{-\beta(E-\mu N)}$ at one particle per token – here $e^{-\beta(E_i+\mu_i)}$, so **our μ_i is minus the chemical potential of the physics convention**, signed instead so that a crowded expert is the costly one; we keep that sign because it is the one in which μ is subtracted from a logit, and the one the solver of §A.4 produces. It is also, by equation 4, the maximum-entropy multiplier of the expert-occupancy feature – minus the temperature times the log prior over the latent – so fixing the batch marginal is empirical Bayes on that prior. The ensembles differ only in which member of the conjugate pair is held fixed: grand-canonically one fixes μ and lets occupancy fluctuate, whereas here we fix occupancy at $1/K$ and solve for μ. The two descriptions are Legendre conjugates, $\Omega(\mu) = F - \sum_i \mu_i n_i$ with $n_i = -\partial\Omega/\partial\mu_i$, exactly as free energy and softmax were conjugate above. Balance is a change of ensemble, not of model – exact as a duality, and only $O(N^{-1/2})$ in the occupancy fluctuations of a finite batch. And because μ is a *dual variable* and not a parameter, it (i) carries no gradient pathway – a multiplier enters the stationarity condition as a constant; (ii) adds no learned parameters, in particular no gate; and (iii) is not a term in the loss. So the two properties we started with, no learned router and no auxiliary balancing penalty, survive exact balancing: a penalty with a coefficient to tune has become a constraint with a multiplier to solve.

The energy decides which expert fits; the chemical potential decides how crowded it is. $E_i(\boldsymbol{g})$ is a property of the *token*, varying from token to token; μ_i is a property of the *expert*, the same for every token. A mixture needs both – an expert can be the right one for a token and still be full – and one set of logits cannot separate the roles. One scalar per expert buys the separation, out of the same variational problem that produced the softmax. So the argument of §3.1 runs a second time: nobody chose the softmax, and nobody chose the load-balancing loss either. A penalty of the familiar form $\alpha K\sum_i f_i P_i$ (Fedus et al., 2022) is a soft, gradient-carrying surrogate for equation 23; μ is that constraint's exact dual. In the assignment $P_{ni} = p_i(\boldsymbol{g}_n)/N$ the two constraints are fixed uniform row and column sums, making this an entropy-regularised optimal transport between tokens and experts with μ its dual potentials. §A.4 solves for μ, measures its cost, and says what we do at inference, where there is no batch marginal to enforce.

A.4 Solving for the chemical potential

§3.4 writes the balanced routing weights as $p_k \propto \exp\big(\beta(a_k - \mu_k)\big)$, with μ_k the multiplier of the batch-occupancy constraint equation 23. This section is the method: how μ is obtained, what it costs, what it approximates, and what it measurably changes.

The solver. The μ satisfying equation 23 is the fixed point of a log-domain iteration, started from $\mu = 0$:

$$\mu \;\leftarrow\; \mu + \log\big(K\cdot \mathrm{load}(\mu)\big), \qquad \mathrm{load}_k(\mu) \;=\; \frac{1}{N}\sum_{n=1}^{N} \mathrm{softmax}_k\big(\ell(h_n) - \mu\big), \tag{25}$$

where $\ell_k = \beta\, a_k$ are the routing logits of §A.2, so μ comes out in logit units and is subtracted from ℓ directly. An expert carrying more than its share has $\mathrm{load}_k > 1/K$, hence μ_k rises and its logits fall. This is Sinkhorn's iteration on the two marginals; we run it under `no_grad`, so μ is a constant of the differentiation as a multiplier should be, and it is deterministic, so activation-checkpoint recomputation reproduces the same μ.

It converges in **three iterations**: on a probe batch $\max_k|\mu_k|$ moves $0.1653 \to 0.1790 \to 0.1802$ over the first three sweeps and is then flat out to twenty (that batch needs a small correction; the magnitudes reached in a trained run are below). We use three. There is no clamp and no gain – the one hyperparameter is that integer.

Why this replaces a control rule. The mechanism we implemented first is proportional control: keep a per-expert bias and, each step, nudge it toward whatever would equalise the observed load. It works – it is the middle row below – but in *every* arm we ran it finished pinned at its stability bound, $|\mathrm{bias}| = 1.0$ exactly, so it delivered that result while saturated and wanted to push harder. The bound cannot simply be relaxed: an earlier unbounded, sign-based version of the same update reached $|\mathrm{bias}| = 1482$ and destabilised a run (training loss $4.05 \to 5.18$). Solving the dual removes the multiplier's dynamics altogether, and the solution needs $|\mu|$ up to ≈ 3.4, more than $3\times$ past that bound – which is the quantitative reason the control rule under-balanced. Both target equation 23; one is exact, the other lagged and clamped.

Cost. $+0.46\%$ of the optimizer step, measured on two arms submitted together to the *same host* and identical apart from the balancing mechanism: 1.2303 against 1.2359 s/step. (Same host, same submission, is not a nicety – s/step at this scale is sensitive enough to host placement that a cross-host comparison could not resolve an effect this small.) The figure is consistent with the arithmetic share: the three softmaxes over an $N \times K$ logit matrix that equation 25 performs, set against the expert projections, are a multiply–accumulate ratio of 7.6×10^{-6}. Essentially none of the 0.46% is arithmetic; it is kernel launches and one reduction over the token dimension.

Two approximations, stated plainly. Train-only. Occupancy is a property of a batch, so enforcing equation 23 at inference would make one token's routing depend on which other tokens happened to share its batch. We solve for μ during training and route with the plain Boltzmann weights, $\mu = 0$, at inference. This is the standard choice for balanced-assignment routers, and it does introduce a train/inference mismatch: the weights the experts were trained under are not the weights they are evaluated under. **Local marginal.** The constraint is enforced on the per-device batch, not globally across data-parallel ranks. The global marginal is what the derivation asks for; the local one avoids a collective in the forward pass, and the two agree to the extent that ranks see exchangeable data.

Measured effect. A 134M model, $K = 32$ experts, top-2; three arms with identical seed, data and schedule, compared at a matched training step. Uniform load would give $\max \mathrm{share} = 1/32 = 0.031$ and $\mathrm{eff}\text{-}K = 32$:

balancing	lm loss	eff-K (of 32)	max share	mean $\lvert E\rvert$
none (as previously trained)	3.7215	7.80	0.3066	0.3975
proportional-control bias	3.7393	16.31	0.0805	0.2716
Sinkhorn dual (this section)	3.7234	**30.80**	**0.0573**	0.2286

Balance improves $3.9\times$ in eff-K – from a quarter of the experts to nearly all of them – and the busiest expert's share falls $5.4\times$, to within a factor of 1.8 of uniform. **We claim no quality gain from this.** The three losses span 0.018 and the run-to-run spread we measure in lm loss at this scale is 0.038, so the defensible reading is that exact balancing is available at no measurable cost in loss and $+0.46\%$ in compute – not that it helps. Whether it helps is a question for a full-length run we have not done. The required $|\mu|$ *decreases* over training, $3.44 \to 1.97$: the model needs less corrective pressure as it settles, not more.

Balanced routing also lowers the energy magnitude. The last column earns its own sentence. Mean $|E|$ falls from 0.3975 unbalanced to 0.2286 under the dual, which bounds a positive-feedback path rather than merely looking tidier: the block moves along its own energy gradient, so the expert selected for a token has its match to that token reinforced at the next step, and a router free to concentrate load compounds that reinforcement on the same expert every step – which is the same event as collapse. Fixing the occupancy caps how much any one expert can compound. The energy is in any case evaluated on the RMSNormed representation and not on the residual stream, so the only remaining growth path is $\|W\|$, which weight decay opposes; the measurement says balancing contributes to the same bound.

A.5 EXPERT-DIVERSITY REGULARISER

To discourage redundant experts we penalise the cosine similarity of expert *outputs* on the current batch (a functional, not weight-space, criterion), sampling K_r expert pairs per step so the cost is K_r dot products of size d:

$$\mathcal{L}_{\text{rep}} \;=\; \lambda\, \mathbb{E}_{(i,j)}\big[\,\rho\big(\cos(g_i, g_j)\big)\,\big], \qquad g_k = \nabla_h a_k. \tag{26}$$

The choice of ρ matters, and the obvious choice is wrong. Taking $\rho(u) = u$ (signed cosine) is minimised at $u = -1$, so it rewards *anti-alignment* rather than diversity. Empirically it does exactly that: across two independently trained models the mean pairwise cosine of expert outputs lands within 3% of the geometric floor $-1/(K-1)$ attainable by K unit vectors (measured -0.1394 against a floor of -0.1429 at $K = 8$; -0.3207 against -0.3333 at $K = 4$). Anti-aligned experts cancel when summed, and we measure a surviving-magnitude ratio $\|\sum_k p_k g_k\| / \sum_k p_k \|g_k\|$ of only 0.19–0.24. We therefore use $\rho(u) = |u|$ or u^2, both minimised at orthogonality. Note u^2 is roughly $3\times$ weaker than $|u|$ at $|u| \approx 0.3$, so λ must be recalibrated when switching between them.

A.6 GRADIENT SCALING AND STABILITY

For W with i.i.d. entries of scale σ and $\|h\| = \sqrt{d}$, the Hopfield gated activation has $\|\varphi \odot \varphi'\| \sim \sigma\sqrt{I_e d}$ and hence $\|W^\top(\varphi \odot \varphi')\| \sim \sigma^2 d\sqrt{I_e}$. The descent step therefore scales as $c \cdot \sqrt{I_e}$:

c	step vs. I_e	outcome
$2/I_e$ (mean)	$\sim I_e^{-1/2}$	decays with width
$1/\sqrt{I_e}$	$\sim$ const	width-invariant
2 (sum)	$\sim I_e^{1/2}$	**diverged** (NaN at step 3710, $I_e = 8192$, lr 7.5×10^{-4})

The mean form makes the *energy* width-invariant, which is correct, but over-corrects the *gradient*. The $1/\sqrt{I_e}$ choice is the width-invariant one for the gradient and sits at least $32\times$ below the configuration that diverged. We expose c as a configuration option and report both settings, because this constant turns out to control whether the energy-FF branch is load-bearing at all.

A.7 WHY A NON-PARAMETRIC CONTEXT DENSITY

The partition function in equation 13 has as many terms as there are context tokens, and their positions are the context itself. So this density has no fixed modes: give the model a different prompt and the landscape it descends is a different landscape. This is a *malleable* piece of the energy surface that fits itself to the problem statement. We would suggest that this is a large part of what makes transformers appear intelligent: not that they store more, but that the objective they optimise at inference is reassembled from the input, so a task never seen in training can still carve out its own basins. In-context learning, from this angle, is not a surprising emergent extra – it is what a context-built KDE does.

A.8 EXACT GRADIENTS THROUGH RMSNORM: THE RAYLEIGH PROJECTION

The energy is evaluated on $\boldsymbol{g} = \sqrt{d}\,\boldsymbol{x}/\|\boldsymbol{x}\|$ (RMSNorm; a learned gain $\boldsymbol{\gamma}$ only replaces $\nabla_{\boldsymbol{g}} E$ by $\boldsymbol{\gamma} \odot \nabla_{\boldsymbol{g}} E$ below, and we ignore the ϵ regulariser). Its Jacobian is

$$\boldsymbol{J} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} = \frac{\sqrt{d}}{\|\boldsymbol{x}\|}\Big(\boldsymbol{I} - \frac{\boldsymbol{x}\boldsymbol{x}^\top}{\|\boldsymbol{x}\|^2}\Big) = \frac{\sqrt{d}}{\|\boldsymbol{x}\|}\,\boldsymbol{P}_{\boldsymbol{g}}, \qquad \boldsymbol{P}_{\boldsymbol{g}} = \boldsymbol{I} - \frac{\boldsymbol{g}\boldsymbol{g}^\top}{d}, \tag{27}$$

using $\|\boldsymbol{g}\|^2 = d$. $\boldsymbol{J}$ is symmetric, so

$$\nabla_{\boldsymbol{x}} E = \boldsymbol{J}^\top \nabla_{\boldsymbol{g}} E = \frac{\sqrt{d}}{\|\boldsymbol{x}\|}\Big(\nabla_{\boldsymbol{g}} E - \boldsymbol{g}\,\frac{\boldsymbol{g}^\top \nabla_{\boldsymbol{g}} E}{d}\Big). \tag{28}$$

$\boldsymbol{P}_{\boldsymbol{g}}$ is the orthogonal projector onto the tangent space of the sphere at $\boldsymbol{g}$: it removes the radial component, as in the gradient of a Rayleigh quotient $\boldsymbol{v}^\top \boldsymbol{A}\boldsymbol{v}/\boldsymbol{v}^\top \boldsymbol{v}$, whence the name. Two consequences follow. First, $\boldsymbol{x}^\top \nabla_{\boldsymbol{x}} E = 0$: since E depends on $\boldsymbol{x}$ only through its direction, its

gradient has no radial part. Second, the descent guarantee. With the exact gradient, the update $\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta\,\Pi\nabla_{\boldsymbol{x}}E$ changes the energy by $\delta E = -\eta\,\nabla_{\boldsymbol{x}}E^\top\Pi\,\nabla_{\boldsymbol{x}}E = -\eta\,\|\boldsymbol{S}^\top\nabla_{\boldsymbol{x}}E\|^2 \le 0$ to first order, for $\Pi = \boldsymbol{S}\boldsymbol{S}^\top + (\boldsymbol{A} - \boldsymbol{A}^\top)$. With the update $\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta\,\Pi\nabla_{\boldsymbol{g}}E$ used by Energy Transformer and NRGPT,

$$\delta E = -\eta\,\frac{\sqrt{d}}{\|\boldsymbol{x}\|}\,\nabla_{\boldsymbol{g}}E^\top\boldsymbol{P}_{\boldsymbol{g}}\,\Pi\,\nabla_{\boldsymbol{g}}E, \tag{29}$$

which is not sign-definite in general, because $\boldsymbol{P}_{\boldsymbol{g}}\Pi$ is not positive semi-definite; it is non-positive when Π commutes with $\boldsymbol{P}_{\boldsymbol{g}}$ or when $\nabla_{\boldsymbol{g}}E$ is already tangent. The two updates differ by the radial component of $\nabla_{\boldsymbol{g}}E$ and by the step-size factor $\sqrt{d}/\|\boldsymbol{x}\|$. In the code, `energy_apply_rayleigh` applies $\boldsymbol{P}_{\boldsymbol{g}}$ and `energy_full_per_token_grad` applies the factor $\sqrt{d}/\|\boldsymbol{x}\|$, which together give equation 28.

B Shared computation between routing and expert forward

The Boltzmann router's central efficiency property is that it reuses the expert forward computation rather than adding to it. This section decomposes the per-expert arithmetic for the W_1W_2 form (§A.1) and compares the routing overhead with a conventional learned gate.

GEMM decomposition for the W_1W_2 expert. The expert output is $\nabla_h a_k = \mathrm{t}_1^{(k)} + \mathrm{t}_2^{(k)}$, the sum of two gradient terms. Four matrix–vector products, each of cost $d \times I_e$ MACs, produce it:

	operation	produces	shared with routing?
G1	$W_1^{(k)}h$	pre-activation $\to \varphi(W_1^{(k)}h)$	yes
G2	$W_2^{(k)}h$	second pre-activation	yes
G3	$\varphi(W_1^{(k)}h)^\top W_2^{(k)}$	$\mathrm{t}_1^{(k)}$ (1st gradient term)	yes
G4	$\big(\varphi'(W_1^{(k)}h) \odot W_2^{(k)}h\big)^\top W_1^{(k)}$	$\mathrm{t}_2^{(k)}$ (2nd gradient term)	no

The routing energy is $a_k = \mathrm{t}_1^{(k)} \cdot h/\sqrt{I_e}$: a dot product of cost d that consumes the output of G1–G3. G4 is needed only for the expert output. **Three of four GEMMs (75% of a selected expert's arithmetic) are shared with routing;** the marginal cost of routing is one d-dimensional dot product per expert plus a softmax over K logits, both negligible relative to the GEMMs.

For Hopfield experts the overlap $a_k = \|\varphi(W^{(k)}h)\|^2/I_e$ reuses the forward projection $W^{(k)}h$ that the gradient also requires, so the shared fraction is one of two GEMMs (50%).

Comparison with a Switch gate. A Switch MoE routes through a separate learned linear $\mathbf{G} \in \mathbb{R}^{d\times K}$, adding dK MACs per token that are entirely disjoint from the expert forward. The table below contrasts the two at the 400M hybrid configuration ($d = 1024$, $K = 32$, $k = 2$, $I_e = 5871$).

	Boltzmann (W_1W_2)	Switch (SwiGLU)
routing mechanism	energy from expert fwd	separate $d\to K$ gate
routing MACs added per token	$K \cdot d = 33$K (dot products)	$d \cdot K = 33$K (gate)
of which shared with expert fwd	33K (all)	0
per selected expert GEMMs	4 (3 shared + 1 new)	3 (all new)

In both architectures the per-token routing overhead ($\sim$33K MACs) is negligible against the selected-expert forward ($k \times 3$–$4 \times d \cdot I_e \approx$ 24–48M MACs). The structural difference is that the Boltzmann router's work is a subset of the expert forward and vanishes on the selected experts, whereas the Switch gate is an independent parameter block whose cost always adds.

Implication for sparse top-k. With the exact energy router, G1–G3 must be evaluated for *every* expert to determine the top-k, whether that expert is ultimately selected or not. The unselected experts' G1–G3 work is pure overhead, and this is what caps the saving at $\frac{1}{2}(1 + k/K)$ of dense.

The proxy router of §H.1 breaks this floor: a rank-r projection plus a small head replaces the all-K scoring at $\sim 0.2\%$ of the exact cost. At the configuration above the proxy reduces the effective routing cost from $3K \cdot d \cdot I_e \approx 577$M MACs to under 0.03M MACs, a factor of $K/2 = 16$. This is the transition from the $\frac{1}{2}$ floor to the k/K regime and the entire content of the arithmetic comparison with Switch at matched parameters.

C Extended related work

§2 states the two comparisons that bear directly on our construction. This section gives the longer account, and says explicitly which parts of the reading in §3 are already in the literature.

Energy-based models, and the cost of a deep-network energy. Energy-based learning replaces a normalised likelihood with a scalar energy whose minima are the model's preferred configurations (LeCun et al., 2006; Hopfield, 1982; Ackley et al., 1985; Hinton & Sejnowski, 1986). Modern Hopfield networks made the connection to attention explicit: a particular energy over stored patterns has a single update step that *is* softmax attention (Ramsauer et al., 2021; Krotov & Hopfield, 2016; Demircigil et al., 2017). The Energy Transformer (Hoover et al., 2023) builds a layer whose forward pass is gradient descent on such an energy, and NRGPT (Dehmamy et al., 2025) derives a full language-model block this way. Two recent lines push the idea to inference-time computation: Energy-Based Transformers (Gladstone et al., 2026) treat every prediction as an optimisation problem, descending a learned energy for a variable number of steps to obtain "System 2" behaviour, and Equilibrium Matching (Wang & Du, 2026) discards time-conditional diffusion dynamics in favour of a single time-invariant gradient field, sampling by gradient descent at inference.

These works share a design that our construction deliberately avoids: **the energy is the output of a deep network**. Two consequences follow, and both are structural rather than incidental. First, *every* optimisation step needs $\nabla_x E$, which is a full backward pass through the network that defines E; T refinement steps therefore cost $O(T)$ network traversals on top of the forward pass, and the gradient cannot be reused across steps because the iterate moves. Second, neither E nor ∇E has a closed form one can inspect: the energy is a black-box scalar, it does not decompose into named contributions, and its stationary points are reachable only numerically. One cannot read off *which* part of the model prefers a given configuration, nor solve for the minimiser analytically. Our energies are instead closed-form per expert – a single bilinear form $E_i(\boldsymbol{x}) = -\phi(W_{1i}\boldsymbol{x})^\top (W_{2i}\boldsymbol{x})$ – so the mixture's log-partition function is available in closed form, its gradient is a *weighted sum of per-expert gradients* with weights $p_i \propto e^{-\beta E_i}$, and the router is therefore an algebraic consequence of the energy rather than something computed by differentiating a network. The price of that tractability is expressiveness: a bilinear energy is far weaker than a transformer's, and we make no claim to the reasoning behaviour those works demonstrate.

Mixture-of-experts routing. Sparse mixtures route each token to a few of many experts through a learned gate (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Dai et al., 2024). The gate is trained by the task loss and, because nothing in that objective prefers balanced experts, it needs an explicit remedy: Switch and its descendants add an auxiliary load-balancing loss plus a z-loss (Fedus et al., 2022), while DeepSeek-V3 removes the auxiliary loss in favour of a per-expert bias nudged under `no_grad` toward under-loaded experts (DeepSeek-AI, 2024). Our routing weights come instead from the Boltzmann distribution over per-expert energies, so there are *no gate parameters to train* and no auxiliary term in the loss; balance is imposed by the chemical potential μ_k in $p_k \propto e^{-\beta(E_k+\mu_k)}$, which is the exact dual variable of the batch-marginal constraint and is solved under `no_grad` by a short Sinkhorn iteration. That places our balancing closer to DeepSeek-V3's loss-free bias than to Switch's auxiliary objective, with the difference that μ is *solved* rather than controlled, so it needs no gain and no clamp. Terminologically adjacent but substantively different work applies Friston's free-energy principle to MoE gating as a source of additive gate modifications rather than as a derivation of the router (Wong, 2026); that work retains the learned gate and reports results at four experts.

Hyperspherical and recurrent-depth energy transformers. Several of the ingredients of §3 appear separately in prior work and we do not claim them. Hu et al. (2025) formalise token dynamics as maximum-likelihood estimation on the hypersphere, with RMSNorm as the sphere constraint, re-

current shared-weight depth, a log-sum-exp attention energy, and a von Mises–Fisher mixture; their feedforward energy $-\sum_i \log \sum_m e^{d_m^\top x_i}$ has the same log-sum-exp form as ours, with the gradient $D\,\mathrm{softmax}(D^\top X)$. Two differences matter. Their mixture components are single learned direction vectors rather than experts with their own parameter matrices, so no mixture-of-experts layer and no router follows from it; and their attention log-sum-exp enters as a uniformity term rather than as a context likelihood. **What we claim as new is the identification of the log-sum-exp Hopfield gradient as a mixture of experts with per-expert feedforward maps and a native router**, together with the chemical-potential reading of balance.

Attention as kernel smoothing. The reading of attention as a non-parametric estimator built from the context also predates us. Tsai et al. (2019) give the kernel view of attention, and later work writes the context-token density explicitly and observes that dot-product attention follows from unnormalised Gaussian kernels. Our contribution here is narrower and quantitative: the von Mises–Fisher specialisation matched to RMSNorm, the identification of the per-kernel normaliser $C_{d_h}(\kappa_{AB})$ as the entire gap between softmax attention and an exact kernel density estimate, and a measurement of κ_{AB} on trained weights showing that the gap is small in the regime our models occupy (§3.6).

Optimal transport and Sinkhorn routing. The variational derivation of §3.4 has a direct precedent. Clark et al. (2022) derive routing as an entropy-regularised optimal transport between tokens and experts and solve it by alternating dual ascent, with one scalar per expert shared across tokens, which is exactly our μ. Lewis et al. (2021) and Zhou et al. (2022) solve related assignment problems, and the latter also reports dropping the auxiliary balance loss. **The mathematics of the balanced solution is therefore theirs, not ours.** Two things remain ours: their logits come from a learned router whereas ours are expert energies, so no gate is involved at any point; and S-BASE retains a load-balancing loss computed on pre-Sinkhorn decisions, so "solved rather than penalised" is a real if narrow distinction. The reading of μ as a chemical potential, and of balance as a change of ensemble with μ conjugate to occupancy, we have not found elsewhere; it is a reinterpretation of their mathematics rather than new mathematics.

Score-based generative modelling. Our motivation for parametrising an energy rather than a score is not that it fits better; on comparable architectures it does not (Salimans & Ho, 2021). It is that a score determines the log-density only up to one constant per connected component of the support, so the relative mass of well-separated modes is not identifiable from it (Song & Ermon, 2019; Wenliang & Kanagawa, 2020), whereas an explicit energy fixes every log-density difference globally, with the intractable normaliser cancelling. That is an argument about what the model class can support, not a property we demonstrate: our energy is trained by token cross-entropy, not fitted as a density.

D Loss curves and token efficiency

Figure 2 shows the training loss (cross-entropy in nats, `lm_loss`) as a function of tokens consumed for every arm in Table 1. Each panel corresponds to one model width; the x-axis is log-scaled so that early training behaviour is visible alongside the final converged region.

Two patterns are stable across all three scales. First, the Switch MoE reaches the lowest final loss at every width, consistent with Tables 1 and 2. Second, the curves separate early (before 0.1B tokens) and remain roughly parallel thereafter, which means the ordering is set by the first few percent of training and the gap neither closes nor widens with further tokens.

D.1 Token efficiency

To quantify the token cost of matching a reference arm's final quality, we define a *token efficiency ratio*: for arms A and B, the ratio is the fraction of B's total token budget at which A's smoothed loss first drops to B's final loss. A ratio below 1.0 means A is more token-efficient; a ratio above 1.0 (or "n/a" if A never reaches B's loss) means A is less efficient. Figure 3 shows the full matrix at d=1024.

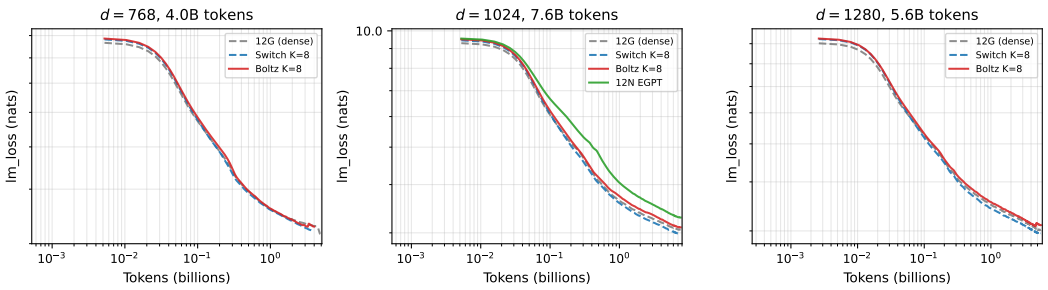

Figure 2: Training loss (nats) versus tokens consumed, plotted on log–log axes at three model widths. Dashed lines are baselines (12-layer dense GPT and Switch MoE); solid lines are energy-based models (Boltzmann MoE and 12N EGPT). A Gaussian smoothing kernel (σ=5 steps) is applied to reduce per-step noise. At every scale Switch converges to the lowest loss, dense GPT is second, and Boltzmann MoE trails by 0.04–0.12 nats. The 12N EGPT baseline at d=1024 starts from a visibly higher loss and converges last, consistent with its recurrence overhead at this token budget.

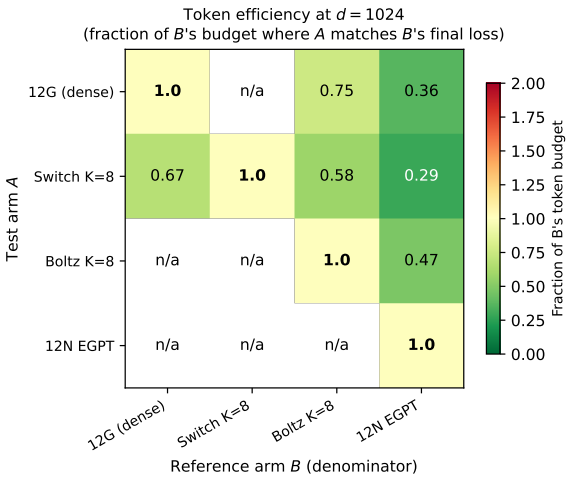

Figure 3: Token efficiency matrix at d=1024. Entry (i, j) is the fraction of arm j's token budget at which arm i first matches arm j's final `lm_loss`. Values below 1.0 (green) mean i reaches j's quality faster; "n/a" means i never reaches j's final loss within its own budget.

The key readings from the matrix are as follows. Switch reaches Boltzmann's final loss at 58% of Boltzmann's token budget, and the dense baseline's at 67% of the baseline's budget; it is the most token-efficient arm at this width. The dense 12G baseline reaches EGPT's final loss at 36% of EGPT's budget, confirming that pure recurrence without MoE is the least token-efficient architecture tested here. Boltzmann MoE never matches the dense baseline's or Switch's final loss within its own budget, so the deficit visible in Figure 2 is not merely a matter of convergence speed.

E Comprehensive results: nemotron-cc development sweep

Table 2 collects every completed arm from the development sweep described in §4. All arms train on a single shard of nemotron-cc web-hq (`p2_0`, 10.4B tokens), with the Granite-4.0 tokeniser, cosine LR schedule at peak 10^{-3}, and bf16 FSDP-2. `Avg11` is the eleven-task zero-shot accuracy mean in percentage points (§F.2); PPL is wikitext word-level perplexity (lower better); `lm_loss` is the final training cross-entropy in nats (lower better). Arms are grouped by scale (hidden dimension) and sorted by `Avg11` within each group. Incomplete runs are excluded. Token budgets differ across scales but are matched *within* each scale group (column "Tokens").

Table 2: **All completed arms in the nemotron-cc development sweep.** Param counts from `audit_config`; † marks baselines (learned gate or dense, no energy routing). "rec" = recurrent (one block applied multiple times); "deep" = distinct layers, no recurrence.

Arm	Arch	K/k	d	Tokens	lm_loss	Avg11	PPL
d=768**, 4.0B tokens**							
Switch MoE †	6G1x6S	8/2	768	4.0B	3.210	**44.72**	**42.1**
Boltz-MoE rec (w1w2, K=8)	6G1x6E	8/2	768	4.0B	3.283	43.96	45.4
12-layer dense †	12G	–	768	4.0B	**3.164**	43.95	39.2
Boltz-MoE deep, iso-act (K=4)	6S6E	4/2	768	4.0B	3.287	43.47	46.7
6S6E deep (K=8, iso-total)	6S6E	8/2	768	4.0B	3.312	43.14	47.3
6S6E deep (K=4, narrow)	6S6E	4/2	768	4.0B	3.415	42.36	56.0
d=768**, 1.0B tokens (probes)**							
E4: Boltz-MoE rec, proj=psd_anti	6G1x6E	8/2	768	1.0B	3.553	40.76	66.4
E5: Boltz-MoE rec, proj=unconstrained	6G1x6E	8/2	768	1.0B	3.574	40.65	67.8
P1: pure EGPT 1x6N, proj=identity	1x6N	–	768	1.0B	4.308	36.36	169.3
P2: pure EGPT 1x6N, proj=unconstrained	1x6N	–	768	1.0B	4.257	35.46	159.7
d=1024**, 7.6B tokens**							
Switch MoE †	6G1x6S	8/2	1024	7.6B	**2.982**	**46.01**	**31.4**
12-layer dense †	12G	–	1024	7.6B	3.057	45.56	34.4
12S all-Switch †	12S	4/2	1024	7.6B	3.036	45.51	33.3
Boltz-MoE deep, iso-act (K=4)	6S6E	4/2	1024	7.6B	3.079	45.07	35.1
6S6E deep (K=4, narrow)	6S6E	4/2	1024	7.6B	3.142	44.75	39.9
Boltz-MoE rec (w1w2, K=8)	6G1x6E	8/2	1024	7.6B	3.101	44.73	36.2
12N EGPT (no MoE) †	12N	–	1024	7.6B	3.288	42.77	47.5
d=1280**, 5.6B tokens**							
Switch MoE †	6G1x6S	8/2	1280	5.6B	**2.948**	**46.89**	**30.0**
Boltz-MoE deep, iso-act (K=4)	6S6E	4/2	1280	5.6B	3.051	46.17	34.3
12-layer dense †	12G	–	1280	5.6B	3.016	46.16	32.9
Boltz-MoE rec (w1w2, K=8)	6G1x6E	8/2	1280	5.6B	3.098	45.45	34.3
6S6E deep (K=4, narrow)	6S6E	4/2	1280	5.6B	3.058	45.01	34.3

E.1 Code lineage: legacy vs. composable energy MLP

Arms B and C test whether the composable refactored code (`EnergyFF_BoltzmannMoE`, 2026-06-28) reproduces the legacy class (`BoltzmannMoE_Energy_MLP`) at matched settings. Both are 6G1x6E recurrent, d=768, w1w2 dense, K=8.

Table 3: **Legacy vs. composable code at matched settings** (d=768, 6G1x6E, w1w2 dense, K=8, 4.0B tokens).

Arm	lm_loss	Avg11	PPL
B: Legacy `BoltzmannMoE_Energy_MLP`	–	43.62	42.5
C: Composable `EnergyFF_BoltzmannMoE`	–	43.96	43.4

The two implementations agree to within the run-to-run noise floor (0.34pp Avg11, 0.9 PPL). The composable class is used for all subsequent arms.

E.2 Projection type: psd_anti vs. unconstrained

The energy model's update rule is $h \leftarrow h - \Pi\nabla E$, where Π is a learned projection. `psd_anti` constrains $\Pi = S^\top S$ (positive semi-definite, guaranteeing energy descent); `unconstrained` learns a free $d \times d$ matrix.

At 1.0B tokens the two are statistically indistinguishable (0.11pp, inside the 0.038-nat noise floor). This is consistent with the full-budget finding at 134M on the 32B-token mixture, where

Table 4: **Projection type ablation** (d=768, 6G1x6E, w1w2, K=8, 1.0B tokens). Both use identical architecture; only `energy_proj_type` differs.

Arm	lm_loss	Avg11	PPL
E4: psd_anti	3.553	**40.76**	**66.4**
E5: unconstrained	3.574	40.65	67.8

`routing_norm=none` closed a 0.76pp accuracy gap. At this scale, the constraint on Π neither helps nor hurts.

E.3 Projection vs. no projection (identity)

A stronger test: does the projection matrix Π matter *at all*, compared to $\Pi = I$ (a plain gradient step)? Arms P1 and P2 are pure recurrent energy models (one block applied six times, no MoE, no softmax-attention prefix) at d=768.

Table 5: **Identity vs. unconstrained projection** (d=768, pure 1x6N EGPT, no MoE, 1.0B tokens).

Arm	lm_loss	Avg11	PPL
P1: $\Pi = I$ (identity)	4.308	**36.36**	169.3
P2: Π unconstrained	4.257	35.46	**159.7**

The learned projection improves loss (-0.051 nats) and perplexity (-9.6), both lower-variance metrics, while the Avg11 ordering reverses ($+0.9$pp for identity). At one seed and 1.0B tokens this is unresolved; the continuous metrics slightly favour a learned Π.

E.4 Expert count and width at fixed active parameters

Arms G3/G4/G5 use K=4 experts with narrower width (to hit the same total params as the K=8 arms), while G6/G7/G81 use K=4 with width chosen to match *active* parameters with the Switch baselines. The iso-active design (G6/G7/G81) consistently outperforms the narrower design (G3/G4/G5).

Table 6: **Expert width: iso-total vs. iso-active** (6S6E deep, K=4 k=2, three scales). Iso-active arms match the Switch baseline's active params; iso-total arms match total params. I_e is per-expert intermediate size, I_e/d the ratio to hidden dimension.

Scale	Design	I_e	I_e/d	lm_loss	Avg11	PPL
d=768	G3: iso-total (narrow)	486	0.63	3.415	42.36	56.0
	G6: iso-active (wide)	1041	1.36	3.287	**43.47**	**46.7**
d=1024	G4: iso-total (narrow)	700	0.68	3.142	44.75	39.9
	G7: iso-active (wide)	1037	1.01	3.079	**45.07**	**35.1**
d=1280	G5: iso-total (narrow)	897	0.70	3.058	45.01	34.3
	G81: iso-active (wide)	1037	0.81	3.051	**46.17**	**34.3**

The iso-active design wins at every scale, by 1.1pp at d=768, 0.3pp at d=1024 and 1.2pp at d=1280. The gap is largest where I_e/d is smallest (G3 at 0.63), consistent with the observation in §4 that per-expert width below $\sim 0.5d$ is dead on arrival.

E.5 Deep (6S6E) vs. recurrent (6G1x6E)

Table 2 shows that 6S6E deep consistently matches or beats 6G1x6E recurrent at $d \geq 1024$, while using half the FLOPs (no block is applied more than once). At d=768 the recurrent arm leads by 0.49pp, but the gap inverts at d=1024 ($+0.34$pp for deep) and widens at d=1280 ($+0.72$pp).

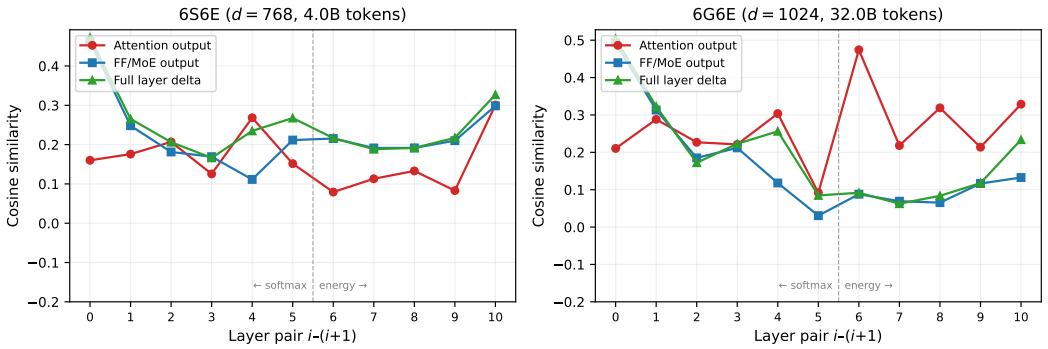

Figure 4: **Consecutive-layer cosine similarity of layer updates** in two deep hybrids: `6S6E` at d=768 (4.0B tokens, single web shard; left) and `6G6E` at d=1024 (32.0B tokens, math-bearing mixture; right), on 32,768 tokens of held-out text. Curves are the attention and FF/MoE contributions *after* the projection Π (they sum to the full layer update) and the full update; energy layers are right of the dashed line. Values are cosines in $[-1, 1]$.

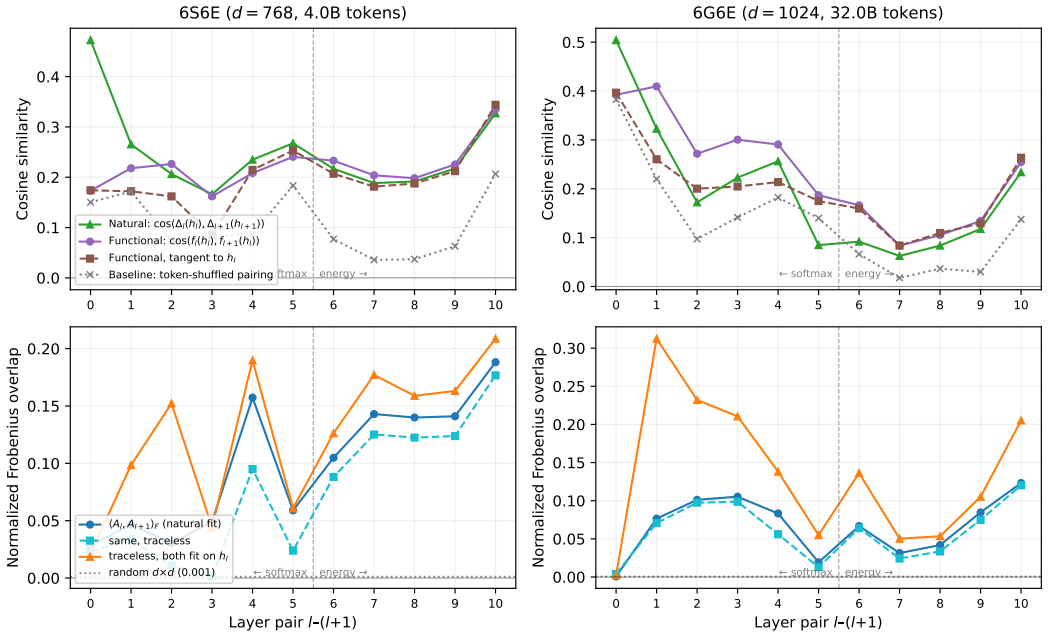

Figure 5: **Functional alignment.** For each consecutive pair, blocks l and $l{+}1$ receive the *same* input states and we report the cosine of their updates (robust to the rotation of successive steps under a fixed flow on the sphere), a token-shuffled baseline, and the normalised Frobenius overlap of least-squares linear operators $\Delta \boldsymbol{g} \approx \boldsymbol{A}_l \boldsymbol{g}$ fitted on 32,768 tokens ($\gg d$, so each $\boldsymbol{A}_l$ is determined; identity part removed; a random pair gives ≈ 0.001).

E.6 Layer output alignment in deep energy models

Above the shuffled baseline, the energy blocks of `6S6E` are more aligned than its softmax layers (excess functional cosine $\approx$ 0.15 vs. 0.07; operator overlap $\approx$ 0.12 vs. 0.03), and neighbouring energy blocks are more aligned than non-adjacent ones (0.21 vs. 0.145). In the 32B-token `6G6E` the ordering appears to reverse (softmax $+0.10$–0.19, energy $+0.06$–0.12), but see the encoding-depth sweep below. The two models differ in data, width, budget and softmax-layer type, so the difference cannot be attributed to training length. The linear fits explain only $R^2 \approx 0.3$–0.5 of the energy-layer updates.

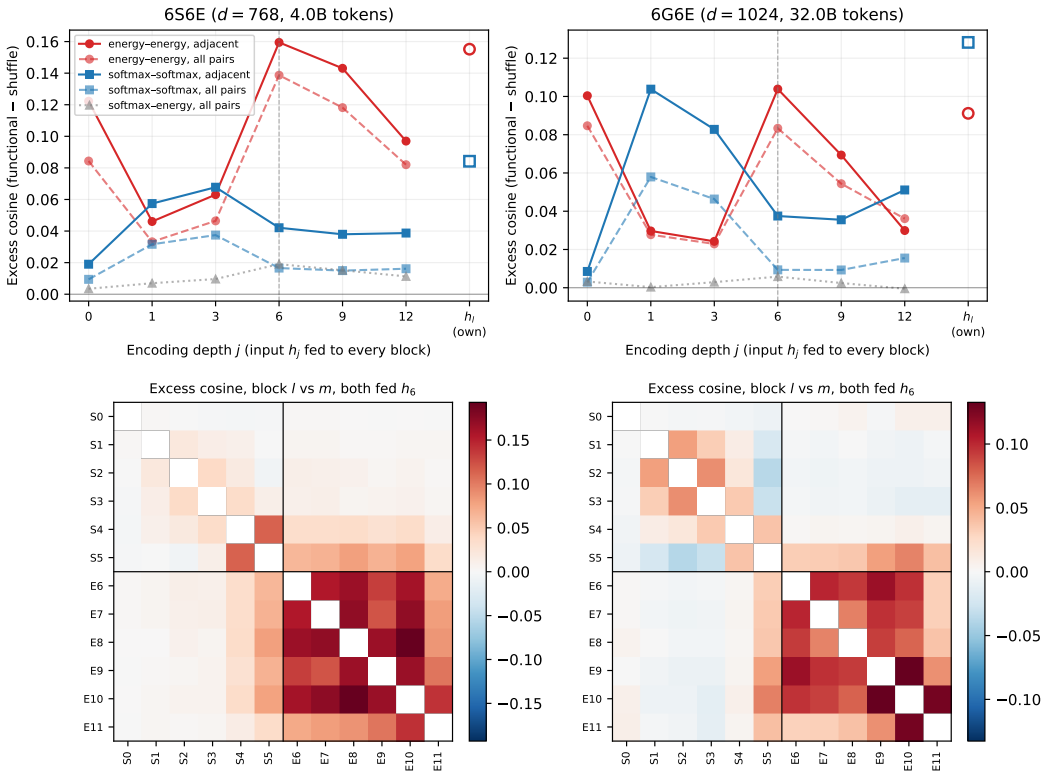

Figure 6: **Functional alignment against encoding depth.** Top: excess cosine (functional minus token-shuffled baseline) of block updates on a shared input $\boldsymbol{h}_j$, versus j, for energy–energy, softmax–softmax and cross pairs. Bottom: 12×12 excess-cosine matrices at j=6. 32,768 tokens per model.

Encoding depth. The comparison above scores each pair on its own input $\boldsymbol{h}_l$, so energy and softmax pairs are scored on different states. Feeding one encoded state $\boldsymbol{h}_j$ (after j layers) to *every* block removes that confound (fig. 6). Excess cosine for adjacent pairs, energy–energy vs. softmax–softmax:

j	0	1	3	6	9	12	own $\boldsymbol{h}_l$
6S6E, 4B	.12/.02	.05/.06	.06/.07	**.16**/.04	.14/.04	.10/.04	.16/.08
6G6E, 32B	.10/.01	.03/.10	.02/.08	**.10**/.04	.07/.04	.03/.05	.09/.13

Each group is most aligned on inputs from its own part of the network (energy blocks at j=6–9, softmax blocks at j=1–3), and on the encoder output energy blocks lead by 3–4$\times$ in both models; the apparent reversal at 32B in the "own $\boldsymbol{h}_l$" column comes from scoring the two groups on different inputs. Softmax–energy cross pairs stay near zero (≤ 0.02). Energy blocks are aligned even on the raw embedding (j=0: 0.12 vs. 0.02), where softmax blocks are not.

Random-initialisation control. The same measurement on an untrained 6S6E (d=768, its own initialisation, same tokens) gives excess cosines of $|x| \leq 0.002$ for every group and depth (j=0, j=6, own $\boldsymbol{h}_l$), and fitted-operator overlaps at the random-matrix level (≤ 0.004). The alignment is therefore produced by training, not by the shared energy-gradient form of the blocks or the positive semi-definite Π.

An arm trained with an explicit alignment regulariser (G2) is inconclusive: it stopped at 1.9B tokens, its loss matches the unregularised arm to within 0.002 nats at matched steps, and the regulariser's effective weight was only $\sim 10^{-5}$ after the auxiliary-loss scaling, so it was essentially inactive.

tabcolsep 3pt

Architecture	Blocks	k/K	Seed	Active/Total	FLOPwt	Avg11 %	WikiPPL
Boltz-MoE deep	6S6E	2/4	7	137M/170M	136.6M	42.44	47.60
Boltz-MoE deep	6S6E	2/4	13	137M/170M	136.6M	42.92	46.91
mean ($\pm$ half-range)						42.68 ± 0.24	47.26 ± 0.34

Table 7: **Seed replicates of the d=768 Boltz-MoE deep arm of Table 1.** Two further seeds (7, 13) of the 6S6E arm, trained with its configuration unchanged except for the seed, which sets both the initialisation and the data order: 7,629 steps at 524,288 tokens per step, i.e. 4.00B tokens of nemotron-cc, cosine LR $= 10^{-3}$. Every layer holds a mixture: six softmax-attention layers with a Switch MoE $(2/4)$ followed by six energy layers with a Boltzmann MoE $(2/4)$. The energy rows are evaluated with the cheap proxy router selecting the experts, as deployed.

Sanity check on a recurrent model. In 6G1x6E, where one energy block is applied six times, the functional cosine between consecutive applications is 1.000 when the per-iteration Sinkhorn potential is held fixed and 0.95–1.0 with it, while the consecutive-delta cosine falls as low as 0.19 for the same function. The functional measure recovers identical dynamics that the naive measure misses.

E.7 12N PURE EGPT: NO-MOE ENERGY BASELINE

Arm F5 (12N, d=1024) is a 12-layer energy transformer with dense EnergyFF_W1W2 blocks and no mixture routing. It scores Avg11=42.77 against the 12-layer dense GPT's 45.56, a 2.79pp deficit at matched total parameters (318M vs 255M). Energy attention and the energy MLP together do not compensate for the absence of softmax attention at this token budget, which is why all reported architectures in Table 1 retain a softmax-attention prefix or interleave.

E.8 SEED REPLICATES

Two further seeds of the 6S6E arm (Table 7) land within 0.48pp of each other on Avg11 and 0.69 on WikiPPL. The Switch baseline, evaluated identically, leads their mean by 1.99pp and 5.16 PPL, about 4× and 8× the seed-to-seed gap, so the ordering of Table 1 at this scale does not rest on the particular seed. Two caveats bound what this shows: two seeds give a range, not a variance estimate, and the Switch baseline spends 43% more FLOPs per token (FLOPwt 195.7M against 136.6M) through its recurrent block, while activating 3% fewer parameters.

F IMPLEMENTATION DETAILS

All numbers in this appendix are measured on the released code path; where a quantity is an analytic count rather than a measurement we say so explicitly.

F.1 DATA AND TRAINING SETUP

Corpus. The main results (table 1) are trained on a single shard of the high-quality web subset of Nemotron-CC (Su et al., 2025), 10.4B tokens in Megatron format, packed to sequence length 4096. The tokeniser is Granite-4.0 tiktoken with a vocabulary of 100,352, and input and output embeddings are tied. The largest budget (9.5B tokens) is below the shard size, so every run is single-epoch, and all arms at a given width draw the same tokens.

Optimisation. AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.1 and gradient clipping at 1.0; peak learning rate 10^{-3} with 1000 warmup steps and cosine decay over the rest of the run (no constant phase); bf16 mixed precision.

Budgets. 4.0B tokens at d=768, 7.6B at d=1024, and 5.6B and 9.5B (1.5× Chinchilla) at d=1280. Tokens per step is GPUs × micro-batch × accumulation × sequence length, with the GPU count set by the launcher, so we verified every budget from the run's logged throughput.

Earlier mixture. The measurements in §H and table 8 use an earlier 32B-token mixture: two Nemotron-CC web shards at 0.35 each plus MegaMath-Web-Pro and FineMath-3+ at 0.15 each, peak learning rate 2×10^{-3}.

Systems. FSDP-2 (per-parameter sharding), block-level gradient checkpointing, `torch.compile`. Wall-clock is quoted only between arms sharing a host allocation: placement alone moved step time by up to 2.7×, through host contention rather than GPU sharing.

F.2 Evaluation protocol

Harness and pinning. All numbers come from EleutherAI `lm-evaluation-harness` at commit `ad3f4d0c` (version **0.4.9.2**), vendored unmodified into the repository so that every collaborator evaluates with byte-identical code. We previously used a floating `pip install lm-eval`, which had resolved to 0.4.11; we diffed the two over everything that can affect a score and found the task YAMLs for all fifteen tasks identical except for Hugging Face `dataset_path` renames (`sciq` → `allenai/sciq`, `super_glue` → `aps/super_glue`, `gsm8k` → `openai/gsm8k`, and similar), with no changes to prompts, few-shot counts, filters, metrics, stop sequences or aggregation. The remaining code differences are a `likelihood` passthrough metric we do not use and typing/docstring modernisation. The pin is therefore for reproducibility, not a correction.

Zero-shot for the accuracy suite. None of the multiple-choice tasks set `num_fewshot` and we do not override it, so all of them run **0-shot**. This is worth stating because 0-shot numbers at the 145M–680M scale sit close to chance on several tasks and are not comparable to few-shot numbers reported elsewhere.

The reported aggregate, and two corrections to our own earlier tables. Three aggregates have been in circulation in our records and the tables did not always say which. The oldest, `avg9`, averaged *nine* tasks with **MMLU excluded** (a dataset installation problem at the time) and used `acc_norm` on all six tasks that report it. An intermediate one, `avg10`, folded MMLU *into* the mean and still omitted `race` and `lambada_openai`, which were unrunnable before we pinned `pyarrow`≥20 (2026-08-03). **This paper reports neither.** All numbers here are `Avg11`, defined below: it is the aggregate used across the wider group of projects we compare against, so that our figures and theirs are directly comparable without restatement, which is the reason we changed to it.

Two consequences a reader should have in front of them. First, because `race` (≈28%) and `lambada_openai` (≈23%) sit near chance at this scale while MMLU leaves the mean, **`Avg11` runs** 1.6**–**2.8**pp below `avg10`** on the same checkpoint. That gap is a scoring convention and not a model effect, which is exactly why `avg9`, `avg10` and `Avg11` figures must never appear in the same table. Second, we found a discrepancy between our earlier *description* of `avg10` and the numbers actually produced under it: the text said `acc_norm` was used for five tasks with plain `acc` for `sciq`, whereas every stored figure in fact scored `sciq` with `acc_norm` as well (plain `acc` would have raised each arm by ≈0.7pp). The published numbers were self-consistent with each other; the prose describing them was not. `Avg11` scores `sciq` with `acc_norm`, so the definition below now states what the code has always done.

Every value in this paper was recomputed from the raw per-task harness output of each checkpoint; nothing was rescaled, interpolated, or carried across conventions. The headline figures move as follows, and no comparison in the paper reverses as a result:

	`avg9`	`avg10`	`Avg11` (reported here)
Boltzmann, soft routing	0.501	0.4832	**0.4578**
learned router, top-2	0.499	0.4817	**0.4533**
margin	+0.17pp	+0.15pp	**+0.45pp**
no-MoE, iso-compute	0.489	0.4750	0.4476
Boltzmann, top-2 trained sparse	0.4856	0.4714	0.4455

A margin of +0.45pp on an eleven-task mean is still inside the seed variation we would expect, so we do not claim a quality win for energy routing from this comparison; we claim parity, and locate the contribution in the efficiency results.

The reported aggregate. We report `Avg11`: the unweighted mean over exactly eleven zero-shot tasks, using `acc_norm` for the six where length-normalised accuracy is standard (`arc_challenge`, `arc_easy`, `hellaswag`, `openbookqa`, `piqa`, `sciq`) and plain `acc` for the other five (`boolq`, `copa`, `winogrande`, `race`, `lambada_openai`). A checkpoint missing any of the eleven is reported as incomplete rather than averaged over the subset it happens to have.

MMLU and GSM8K are deliberately not in the mean, and we report them as separate columns instead. At 134M parameters both are at chance: across every arm of our earlier 134M sweep, MMLU spans 24.2–26.6% against 25% for random choice on four options, and GSM8K-CoT spans 1.6–2.4%. Averaging an at-chance task into an aggregate does not merely add variance, it compresses the spread between arms – a difference of 1pp over eleven informative tasks reads as 0.9pp once a twelfth at-chance task is averaged in – so we quote both per arm and draw no architectural conclusion from either. Perplexity is likewise reported separately: `wikitext word_perplexity` (`loglikelihood_rolling` on `wikitext-2-raw-v1`). Where a single GSM8K figure is quoted it is the `gsm8k_cot flexible-extract` score, for the reason given below.

GSM8K: few-shot counts and answer extraction. Both GSM8K variants are `generate_until` with greedy decoding (`do_sample: false`, temperature 0, `repeats: 1`).

- `gsm8k` – **5-shot**, exemplars drawn from the train split. Prompt `"Question: {{question}}\nAnswer:"`; generation stops at `"Question:"`, `"</s>"` or `"<|im_end|>"`. Two scoring filters are reported: **`strict-match`** extracts with `#### (\-?[0-9\.\,]+)`, i.e. it requires the model to reproduce the literal `####` separator of the GSM8K gold format; **`flexible-extract`** uses `(-?[$0-9.,]{2,})|(-?[0-9]+)` with `group_select: -1`, i.e. it takes the *last* number appearing in the completion. Matching ignores case and strips `","`, `"$"`, everything up to and including `#### `, and a trailing period.
- `gsm8k_cot` – **8-shot**, with the eight fixed chain-of-thought exemplars supplied verbatim in the task config (`fewshot_config.sampler: first_n`), not sampled. Prompt `"Q: {{question}}\nA:"`. Here `strict-match` extracts with `The answer is (\-?[0-9\.\,]+).`; `flexible-extract` is the same last-number regex as above.

Non-standard reporting choice. Energy-based models in this family frequently produce a correct final number without emitting the `####` separator or the phrase "The answer is". Under `strict-match` such a completion scores zero, so strict-match systematically understates them. Where we summarise GSM8K with a single figure we therefore use `flex_avg`, the mean of the `flexible-extract` scores of `gsm8k` and `gsm8k_cot`. Because `flexible-extract` takes the last number in the completion, it can also credit a lucky trailing number, so it is an upper bound where strict-match is a lower bound; we report both filters per task and use `flex_avg` only for compactness.

BBH requires a local fix, and our BBH numbers are not stock. The upstream `bbh_fewshot` task ships **without a `filter_list`**, so `exact_match` is a verbatim string comparison between the raw continuation and the gold target. With the upstream prompt `"Q: {{input}}\nA:"` and `target_delimiter: " "`, a model that copies the few-shot format emits `" False"` against a gold `"False"`, and `" False" == "False"` is false. Every model therefore scores exactly 0.0000 on all 27 subtasks – which is what our earlier BBH runs reported. The parallel `bbh_zeroshot` configs in the same package *do* ship per-subtask `flexible-extract` chains (`\b(True|False)\b`, `\(([A-Z])\)`, `[-0-9]+`, ...); the few-shot variant simply has none wired up.

We fix this by post-hoc rescoring rather than forking the package: BBH is run with `--log_samples`, and the per-subtask zero-shot regexes are then applied to both prediction and target, with a generic fallback (strip whitespace, take the first line, strip trailing punctuation, casefold) for `dyck_languages` and `word_sorting`. On one checkpoint this moved the macro-average from 0.0000 to 0.2649. Consequently **our BBH numbers are not comparable to stock-harness BBH numbers** unless those are rescored the same way, and any BBH figure we publish must carry this note.

Architecture	Blocks	k/K	LR	Active/Total	FLOPwt	Avg11 %	WikiPPL
Switch MoE †	12S	2/8	1×10^{-3}	129M/200M	129.1M	46.07	39.06
Switch MoE †	12S	2/8	2×10^{-3}	129M/200M	129.1M	**46.11**	**38.96**
Learning-rate sweep							
Hybrid (energy MoE)	8S4E	$2/8\cdot2/32$	1×10^{-3}	122M/200M	122.2M	44.45	43.67
Hybrid (energy MoE)	8S4E	$2/8\cdot2/32$	2×10^{-3}	122M/200M	122.2M	44.65	42.94
Hybrid (energy MoE)	8S4E	$2/8\cdot2/32$	3×10^{-3}	122M/200M	122.2M	45.24	43.29
Hybrid (energy MoE)	8S4E	$2/8\cdot2/32$	4×10^{-3}	122M/200M	122.2M	44.75	43.52
Hybrid (energy MoE)	8S4E	$2/8\cdot2/32$	5×10^{-3}	122M/200M	122.2M	44.74	43.92
Router ablations at LR 2×10^{-3}, one change each from the row above							
proxy init random→SVD	8S4E	$2/8\cdot2/32$	2×10^{-3}	122M/200M	122.2M	45.52	42.96
proxy rank $16\rightarrow32$	8S4E	$2/8\cdot2/32$	2×10^{-3}	125M/203M	124.8M	45.56	42.95
proxy out-dim $512\rightarrow1024$	8S4E	$2/8\cdot2/32$	2×10^{-3}	123M/201M	123.3M	45.37	42.98
proxy loss coef $0.01\rightarrow0.05$	8S4E	$2/8\cdot2/32$	2×10^{-3}	122M/200M	122.2M	44.61	41.79
candidates $p=2\rightarrow4$	8S4E	$2/8\cdot2/32$	2×10^{-3}	122M/200M*	122.2M*	43.88	42.47

Table 8: **A 200M screening sweep: learning rate and cheap-router hyper-parameters for the energy hybrid, at 15.73B tokens.** Twelve arms at ≈200M total parameters, each trained for 30,000 steps at 524,288 tokens per step (4 GPUs × micro-batch 8 × 4 accumulation steps × 4096), i.e. 15.73B tokens – half the 32B budget of our earlier runs – with a plain cosine schedule (500 warmup steps). The data use the earlier 70/30 web/mathematics weights but draw from a document subset of the two web shards, so batches are *not* token-identical to our earlier 32B-token runs; compare rows within this table only. 8S4E is eight softmax-attention layers with a Switch MoE (k/K=2/8) followed by four energy layers with a Boltzmann MoE (2/32); 12S puts a Switch MoE in all twelve. Active/Total/FLOPwt as in table 1 (audit_config, identical to the instantiated model's parameter count); the two larger proxy arms are genuinely bigger, because the proxy router has its own parameters. The energy rows are evaluated with the cheap proxy router selecting the experts, as deployed. * p=4 exactly re-scores four experts per token, which touches ≈2.4M more expert parameters than its Active count shows. Bold marks the best Avg11 and WikiPPL; † marks baselines. Single seed per arm.

Decoding and inference settings. batch_size 4, bfloat16 weights, single GPU, trust_remote_code enabled (the energy architectures are custom AutoModel classes). Datasets are resolved from a local Hugging Face cache with HF_DATASETS_OFFLINE=1 and HF_HUB_OFFLINE=1, so no evaluation depends on network state.

A caveat on comparing to our own earlier tables. Before this protocol was unified, three launcher scripts in our repository carried three different task lists (fifteen, thirteen and eleven entries), so historical "Avg" figures were not always computed over the same tasks. The definition above is now the single source of truth, and any table mixing pre- and post-unification numbers should be regenerated rather than concatenated.

G A 200M SCREENING SWEEP ON A MATH-BEARING MIXTURE

A 200M sweep does not close the gap to Switch; only the distillation weight moves the loss. Before spending 400M compute we screened the energy hybrid's learning rate and the four knobs of its cheap router at 200M (Table 8). Three readings survive the noise floor. First, **the Switch baseline leads every energy arm on the continuous metrics**: by 0.069 nats of lm_loss over the energy arm at the same learning rate and 0.051 nats over the best energy arm, both outside the 0.038-nat run-to-run spread, and by 2.8–5.0 in WikiPPL. Switch also activates 5.6% more parameters (129.1M against 122.2M), so part of that lead is compute. Second, **the energy hybrid is insensitive to learning rate**: $1\text{–}5\times10^{-3}$ spans 0.014 nats with a flat optimum at $2\text{–}3\times10^{-3}$, which confirms the 2×10^{-3} used throughout. Third, **the proxy's capacity and initialisation are not the bottleneck**. SVD initialisation, doubling the proxy rank and doubling its output width each move lm_loss by at most 0.0011 nats, and all three end at the same top-2 agreement with the exact router as the default

(0.74–0.76 against 0.75). Raising the distillation weight from 0.01 to 0.05 is the only change that moves loss and perplexity together (-0.018 nats, -1.1 PPL), and even that sits inside the noise floor. The proxy-capacity arms gain 0.7–0.9pp of `Avg11` with no loss movement at all, the pattern we declined to read as a quality gain at 134M, and we do not read it as one here.

H SPARSE ROUTING: PROXY, SURROGATE, AND MEASURED COST

H.1 CHEAP PROXY ROUTER

To reach the k/K cost regime the router must not touch every expert's weights. We fit a small proxy $\hat{E}(h) = \mathrm{MLP}(V^\top h)$ with $V \in \mathbb{R}^{d\times r}$ shared across experts and K outputs, trained with KL against the exact routing distribution on cached activations with the backbone frozen. Its cost is $dr + rh + hK$ multiply–accumulates per token against $d\,I_{\text{tot}}$ for the exact router – under 0.5% at $r = 32$, $h = 64$.

Fitting the head directly is essential, and we report a correction to our own earlier analysis. A cheaper diagnostic – evaluating the *exact* energy at the rank-r projection $V_r V_r^\top h$, which upper-bounds any router seeing only r linear features – behaves very differently across the two expert forms. For Hopfield experts, whose weight matrices are strikingly low rank in practice (participation-ratio effective rank 1.9–2.9 out of 1024; rank-16 captures ∼93% of $\|W\|_F^2$), it rises monotonically to 96.7% top-1 and 98.7% top-2 agreement at $r = 16$. For W_1W_2 experts the same estimator plateaus near 0.5 and is *non-monotone* in r (0.540 at r=2, 0.416 at r=16), which cannot be a valid ceiling since it must approach 1 as $r \to d$. We initially read this as evidence that cheap routing does not transfer to the W_1W_2 form. It does: the projection diagnostic was simply the wrong instrument, because projecting h perturbs a bilinear energy in a way that does not preserve ordering. A *fitted* head on the same activations reaches 97.8% top-1 and 97.3% top-2 agreement (held-out KL = 0.0042) at $r = 32$, $h = 64$ – 26.9K against 12.58M multiply–accumulates, or 0.214% of the exact router.

Initialising the proxy from the expert weights, not from noise. The proxy's functional form is the exact energy with each $W_k \in \mathbb{R}^{I_e\times d}$ replaced by a factored pair: $\hat{A}_k(h) = \mathrm{mean}\big(\phi(B_k V_k^\top h)^2\big)$ with $V_k \in \mathbb{R}^{d\times r}$ and $B_k \in \mathbb{R}^{m\times r}$. It is therefore a rank-r factorisation of W_k – but one whose factors we were learning from a random start, by distillation alone. Given that Hopfield weights are already close to low rank (above), the principled initialisation is the factorisation itself: take $W_k = U\Sigma V^\top$ and set $V_k := V_{:,1:r}$, with B_k the m rows of $U\Sigma_{1:r}$ of largest norm. Applied at the dense-to-sparse switch – at initialisation W is random and its SVD carries no information – this costs one thin SVD per expert, once.

On a trained 1B checkpoint ($K = 64$, $r = 16$, $m = 512$, $I_e = 2846$), top-2 agreement with the exact energies over four mixture blocks:

proxy parameterisation	top-2 agreement
random initialisation (floor)	0.038
distilled (as trained)	0.089
SVD of W_k	**0.554**

a 6.2× improvement over what distillation had reached, for no additional training. Two honest limits. The distilled baseline here had received only ∼300 sparse steps of distillation, so this compares a warm start against an early-training proxy rather than a converged one. And the SVD is optimal for approximating $W_k h$ in Frobenius norm, which is *not* the same objective as *ordering* $\mathrm{mean}(\phi(W_k h)^2)$ across experts; 0.554 is consequently a starting point rather than a ceiling, and distillation continues from it. We therefore treat SVD initialisation as removing the cold start, not as replacing the fit.

End to end on the real model with a frozen backbone, paired next-token NLL over 94,131 identical tokens ($K = 4$, 145M):

router	perplexity	Δ
exact energy router (dense)	6.1814	–
proxy router (dense)	6.3183	+0.137
exact energy router + top-2	6.0943	−0.087
proxy router + top-2	**6.1529**	**−0.029**
proxy router + top-1	6.6835	+0.502

The operative row is the fourth: a router costing 0.21% of the exact one, combined with top-2 truncation, is *slightly better* than the exact dense router it replaces.

But this does not generalise, and the pattern is the opposite of what the projection diagnostic predicted. Repeating the measurement on six further checkpoints, the cost of substituting the proxy ranges from a small gain to a substantial loss:

checkpoint	experts	top-1 agree	exact dense PPL	ΔPPL, proxy + top-2
K=4, soft-trained	W_1W_2	0.978	6.1814	**−0.029**
K=16, top-2 trained	W_1W_2	0.957	7.3082	**−0.690**
K=32, top-2 trained	Hopfield	0.962	6.6278	+0.132
K=32, top-1 trained	Hopfield	0.974	6.7300	+0.292
K=16, top-2 trained	Hopfield	0.976	6.5528	+0.405
K=16, soft-trained	Hopfield	0.934	6.4527	+1.660

Two things follow. First, **high top-1 agreement is not sufficient**: at 0.934–0.978 agreement the perplexity cost still spans -0.69 to $+1.66$, so the 2–7% of mis-routed tokens are not benign and agreement should not be reported as if it were the result. Second, **the proxy transfers well to the W_1W_2 form and poorly to Hopfield** – the reverse of what the rank-r projection ceiling suggested, where Hopfield reached 96.7% monotonically and W_1W_2 plateaued near 0.5. We therefore treat the projection analysis as diagnostic only and report end-to-end perplexity as the outcome that matters.

A consistent secondary pattern: sparsely-trained checkpoints tolerate the approximation far better than densely-trained ones ($+0.13$ to $+0.41$ versus $+1.66$ for Hopfield), which is the same robustness that makes trained-sparse truncation free. The usable configuration is therefore *train sparse, then distil the router*: at K=32, k=2 that costs $+0.13$ perplexity (2% relative) for a router at 0.2% of exact cost and a $16\times$ reduction of the mixture block.

We note that the naive analytic shortcut fails: approximating $\|\varphi(Wh)\|^2 \approx \frac{1}{2}\|Wh\|^2$ (exact for a linear unit, and the natural guess for ReLU) attains under 1% top-1 agreement, because $\varphi(z)^2$ is not proportional to z^2 pointwise for trained weights.

H.2 Surrogate router: a learned gate distilled to the Boltzmann posterior

The surrogate replaces the proxy's factorised energies with a free-form head $\hat{q}(\cdot \mid \boldsymbol{g}) = \mathrm{softmax}(\boldsymbol{H}(\boldsymbol{g}))$, where $\boldsymbol{H} : \mathbb{R}^d \to \mathbb{R}^K$ is a linear map or a one-hidden-layer MLP. Architecturally it is a Switch gate. It differs in two ways. First, its only training signal is $\lambda\,\mathrm{KL}\big(p(\cdot \mid \boldsymbol{g}) \,\|\, \hat{q}(\cdot \mid \boldsymbol{g})\big)$ against the exact Boltzmann posterior, computed on a detached copy of $\boldsymbol{g}$, so the language modelling loss never reaches it and it cannot reshape the representation. Second, it only *nominates*: it proposes $p \geq k$ candidates, whose exact energies are then evaluated and re-ranked to the final k, and the mixture weights are the renormalised posterior equation 11. Because the proxy's rank-r projection is tied to the Hopfield energy form, while the surrogate makes no assumption about the expert, it is the natural choice for W_1W_2 experts, whose bilinear energy is poorly ordered by a projection of $\boldsymbol{g}$ (§H.1). On a 134M W_1W_2 hybrid (K=16) trained through the surrogate, deployed sparse routing scores 44.93 `Avg11` against 45.01 with exact all-K routing on the same checkpoint (-0.08pp, perplexity $40.70 \to 40.85$; table 10). In our earlier grid every Hopfield arm used the proxy and every W_1W_2 arm the surrogate, so the two routers cannot be compared independently of the expert form.

H.3 COST: THE ROUTER IS SHARED, NOT CHEAP

In both energy forms the routing energies and the expert outputs are computed from the *same* matrix products: the router performs exactly half the block's multiplies and that half is reused by whichever experts it selects. Routing therefore costs nothing extra – but the router evaluates *all* K experts and so cannot be skipped, which caps exact-router top-k at $\frac{1}{2}(1 + k/K)$ of dense and **cannot beat** $2\times$ however small k is. A learned gate has the opposite structure: a $d \to K$ linear, negligible and separate from the experts, obtaining the full k/K saving exactly. Sharing buys elegance and forfeits skippability.

The consequence is that Boltzmann-MoE is competitive on arithmetic only if the routing decision is approximated. Substituting a cheap proxy – a rank-r projection and a small head, under 0.3% of the exact router's multiplies – removes the floor:

Table 9: Mixture-block cost per token at matched mixture parameters (12.58M), $k = 2$. Avg11 is the eleven-task accuracy mean in percentage points, higher better; each row's accuracy is measured with the router that row names, on the same checkpoint. The proxy row is a checkpoint trained with *dense* routing and sparsified afterwards, which we show below is the worst case – arms trained through the proxy lose essentially nothing (Table 10).

configuration	MACs/token	vs. Switch	Avg11 %	k/K
Switch SwiGLU, K=16	1.58M	1.00×	**44.83**	0.125
Boltzmann, K=32, *exact* router	13.37M	8.43×	44.58	**0.062**
Boltzmann, K=32, proxy router	**1.60M**	**1.01×**	43.04	**0.062**

With the exact router the method is $8.4\times$ off the pace and not worth building. With the proxy it reaches **parity on arithmetic** while routing twice as sparsely. What that approximation costs in accuracy depends entirely on whether the model was trained through it, and the distinction is large enough that we report the two cases separately.

Sparsifying *after* the fact is expensive. The proxy row above is the K=32 arm trained with dense soft routing and given a rank-16 proxy afterwards: selecting 2 of 32 through that proxy costs 1.54pp of Avg11 ($44.58 \to 43.04$) and takes wikitext word-perplexity from 40.65 to 69.05. A second post-hoc arm degrades further, to perplexity 269.94. We report these because they bound what post-hoc sparsification of an energy mixture can be expected to do, and the bound is unflattering.

Training *with* the sparse path removes almost all of that cost (Table 10). End to end on the mixture block, proxy routing with grouped dispatch measures $19.0\times$ over dense soft routing at 431 TFLOP/s, whereas masking without a proxy measures $1.00\times$ – it skips no arithmetic at all. None of this is a *training* saving: every arm evaluates all experts and masks, so the claim is inference-only.

Across the eight arms above the line the mean change is -0.13pp. We deliberately do *not* describe any arm as improved: re-evaluating intermediate checkpoints of two arms at 8, 16, 24 and 32B gives seven within-checkpoint measurements whose mean is $+0.09$pp with a standard deviation of 0.37pp, so the whole spread – including the largest positive reading – is consistent with zero. That series also shows the penalty is **flat in training budget**: it is already indistinguishable from zero at 8B and does not narrow thereafter, so free sparsity is not something a long run earns. The same holds in the failing direction – the pure recurrent arm discussed below loses 6.90pp at 8B and 7.96pp at 32B, so quadrupling the tokens moves a 7pp defect by about 1pp. The penalty is therefore a property of the architecture rather than of training duration, which makes it *cheap to audit*: a sparse configuration can be accepted or rejected at a quarter of the budget. Taking 0.37pp as the empirical noise floor, selecting 2 of 64 experts at 1B (-0.37pp) and 2 of 16 at 134M (-0.15pp) are both within one standard deviation of free. Perplexity rises slightly and almost everywhere ($+0.1$ to $+2.1$), which is the signature of a small approximation that the accuracy metric is too coarse to resolve. Compared with the 1.54pp and $+28.4$ perplexity that the same proxy family costs when grafted onto a dense-trained arm, the conclusion is that **co-adaptation, not the proxy's raw fidelity, is what makes the sparsity free**. We can make that claim sharply: the post-hoc arm's proxy agrees with the exact top-2 on 0.67 of tokens while the 400M sandwich's agrees on only 0.43, and it is the arm with the *worse* proxy that loses nothing. Once the experts are trained to be selected by this router, its disagreements with the exact energy ranking are no longer errors.

Table 10: **What the cheap router actually costs, on arms that trained through it.** Each row is one checkpoint evaluated twice: *orc* computes all K expert energies and bypasses the cheap router, *sps* lets the rank-16 proxy (or, for the `w1w2` rows, a distilled MLP head) nominate the p candidates whose exact energies then re-rank to the final $k = 2$. Avg11 is the eleven-task accuracy mean in percentage points, **higher better**; Δ is sps $-$ orc, so negative means sparsity costs accuracy. ppl is wikitext word-level perplexity, **lower better**. Only *sps* is deliverable at the arithmetic cost of Table 9.

arm	K	Avg11^{orc}	Avg11^{sps}	Δ	ppl orc$\rightarrow$sps
134M sandwich	16	43.45	44.34	$+\mathbf{0.89}$	$40.98 \rightarrow 41.79$
134M w1w2, unc. proj.	16	44.32	44.37	$+0.06$	$41.09 \rightarrow 40.74$
400M sandwich	32	44.52	44.52	$+0.00$	$41.13 \rightarrow 43.05$
134M w1w2, surrogate	16	45.01	44.93	-0.08	$40.70 \rightarrow 40.85$
134M hybrid	16	44.82	44.67	-0.15	$41.06 \rightarrow 41.82$
400M hybrid	32	47.29	46.99	-0.30	$30.39 \rightarrow 31.53$
1B stacked	64	47.79	47.42	-0.37	$29.46 \rightarrow 30.13$
400M deep 12$\times$	32	46.92	45.97	-0.96	$30.14 \rightarrow 32.22$
134M pure recurrent	16	42.00	34.04	$\mathbf{-7.96}$	$66.08 \rightarrow 446.72$

The last row is a genuine exception, and a controlled experiment decomposes it rather than leaving it unexplained. The pure recurrent arm applies one mixture block 12 times where every other arm applies its shared block at most six, and it carries a *single* cheap-router head for all 12 applications. Two defects were candidates: the Sinkhorn chemical potential μ was not persisted to the checkpoint, so the router is trained tilted and evaluated untilted; and one head may be unable to serve twelve different activation distributions. Two exports of this arm differing in *exactly* that one field give:

μ persisted	routing	Avg11	ppl
no	oracle	42.00	66.08
no	sparse	34.04	446.72
yes	oracle	40.98	67.36
yes	sparse	36.47	156.49

Restoring μ recovers 3.45pp of the 7.96pp collapse and cuts the perplexity blowup from $6.8\times$ to $2.3\times$, so the missing μ is roughly **half** the effect – and it is amplified by sparse routing precisely because an untilted router changes *which* experts are selected, not merely how they are weighted. A -4.51pp penalty nevertheless survives, and the stored head's agreement with the exact top-2 falls from 0.71 at the first application to ~ 0.49 at later ones, so we attribute the remainder to one head being asked to serve too many applications.

What separates this arm from the ones that pay nothing is *how much of the network is routed*. Writing the routed fraction as routed block applications over all block applications, every arm at two-thirds or below pays nothing measurable (-0.37 to $+0.89$pp, within one standard deviation of the 0.37pp noise floor above), whereas the true sandwich at three-quarters pays -1.27pp (3.7 standard deviations) and this pure arm at unity pays -7.96pp (21.6). The cleanest comparison holds recurrence depth fixed at six applications and varies only the fraction: the hybrid at one-half loses 0.15pp where the true sandwich at three-quarters loses 1.27pp, which recurrence count alone cannot explain. We report this as a design rule rather than a mechanism – it rests on one controlled pair, and the two arms differ in depth as well as fraction – but the rule is the part that matters in practice, and it covers every configuration we advocate: **keep at most about two-thirds of block applications behind the cheap router and the sparsity is free**. The actionable form of this is a requirement rather than a caveat: a recurrent arm needs its μ persisted *and*, beyond about six applications, more than one router head before its sparse path can be trusted. Note also that recalibrating μ costs 1.02pp on the *oracle* path, so it trades dense accuracy for sparse robustness rather than being free.

Training cost, which the inference story does not cover. Everything above concerns *inference* arithmetic. The training cost is a separate matter and it runs against us. At matched parameters and matched tokens, our hybrid performs twelve block applications per token (six encoder blocks

plus one energy block iterated six times) with energy attention, against seven for a plain stack with softmax attention. Measured directly on the 400M pair now training – identical parameters to within 0.01%, identical token budget, identical hardware – the hybrid runs at 2.49 s/step against 0.55 s/step for the energy-free baseline, a factor of 4.5. So even parity with that baseline is bought at roughly $4.5\times$ the training compute, and a reader comparing at equal training cost rather than equal parameters should expect the energy-free stack to come out ahead. The recurrence that makes the energy interpretation possible is not free, and we would rather state its price than let the inference-side numbers stand in for it.