MEMORY AS COMPUTATION:
ASSOCIATIVE MEMORY AS A FOUNDATION FOR MODERN AI

A Dissertation
Presented to
The Academic Faculty

By

Benjamin Hoover

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Machine Learning in the
School of Computational Science and Engineering

Georgia Institute of Technology

July 2026

MEMORY AS COMPUTATION:
ASSOCIATIVE MEMORY AS A FOUNDATION FOR MODERN AI

Thesis committee:

Duen Horng Chau, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Judy Hoffman
Department of Computer Science
University of California, Irvine

Zsolt Kira
School of Interactive Computing
Georgia Institute of Technology

Kartik Goyal
School of Interactive Computing
Georgia Institute of Technology

Dmitry Krotov
Co-Founder & CEO
Dynamical Mind

Date approved: July 6, 2026

How mind emerges from brain is to me the deepest question posed by our humanity.

John Hopfield

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Modern AI has achieved extraordinary capabilities, but it has done so through increasingly expensive and opaque systems. At the same time, leading AI architectures strongly resemble older theories of physical memory systems — most famously, the Hopfield Network — that compute by iteratively correcting errors. This thesis shows that using these *energy-based associative memories* (AMs) as a design language for the next generation of AI architectures points toward a paradigm where interpretable, efficient, and robust AI emerges from physical computation built around memory, energy, and dynamics.

(1) **Modernizing Associative Memory**. Classical AMs are too rigid for modern workloads. We propose the *Energy Transformer* (ET), a novel AM that recurrently minimizes a Lyapunov energy resembling a transformer. ET matches vanilla transformer performance while using $>10\times$ fewer parameters that are directly interpretable. *Energy GPT* (NRGPT) scales ET to language modeling, competitively performing against GPT baselines on key benchmarks. *Memory in Plain Sight* connects AMs to diffusion modeling, where denoising generation is synonymous with memory retrieval in the absence of Lyapunov stability.

(2) **Generalizing Associative Memory**. Modern AMs are built using Dense Associative Memories (DenseAMs) that improve Hopfield Networks, but have limited generalizability and memory capacity tightly coupled to their size. We introduce the *Distributed representation for DenseAM* (DRDAM) to decouple memory capacity from parameter count, distributing memories across all synaptic connections. In *Log-Sum-ReLU DenseAM* (LSRDAM), we introduce optimal density estimation kernels to elicit unprecedented numbers of *emergent* memories, boosting creativity while maintaining exponential memory capacity.

(3) **Unifying Associative Memory**. We need a systematic design language to scale DenseAMs. We propose the *Hierarchical Associative Memory User eXperience* (HAMUX), the first work to distill the primitives of AMs into a library of composable neuron and synaptic-layer energies. By expressing AMs as sums of local energy components, HAMUX provides a framework for building hierarchical AMs atop flexible neural network operations.

This dissertation brings the rigor of physics to AI architectures, grounding the resurgence of energy-based models in a theory of physical computation. Our research has been featured in *Nature Reviews* and *Quanta Magazine* (ET and *Memory in Plain Sight*), and recognized at flagship AI conferences by a NeurIPS Spotlight (LSRDAM, top 3%), two full tutorials at ICML and AAAI, and four dedicated workshops across NeurIPS, ICLR, and ICCV.

CHAPTER 1
INTRODUCTION

Artificial Intelligence (AI) has revolutionized fields such as computer vision, biology, and natural language processing at a rapid pace. Powering this revolution are transformer-based architectures [3, 4] and diffusion-based generative models [5, 6], two overlapping paradigms whose performance scales with larger model sizes, more data, and more compute. This success has produced an epidemic of large, expensive, and non-transparent models, forcing us to consider whether the current paradigm of machine learning is the only path toward capable AI. *Can we find a class of architectures that are more interpretable, more theoretically grounded, and still capable of scaling to modern workloads?* I believe the answer is yes.

Curiously, many of the dominant systems and modeling paradigms in modern AI increasingly resemble physical systems that compute by iteratively correcting errors. Models built from transformers repeatedly route information between tokens to inpaint masks or predict missing next tokens, while diffusion-based generation explicitly turns noise into data through a sequence of denoising steps. These systems are usually explained through the language of prediction, attention, or generative sampling, but from a physical perspective they mechanistically look like dynamical systems that move corrupted inputs toward more plausible outputs. The practical success of these models therefore exposes a theoretical curiosity: it

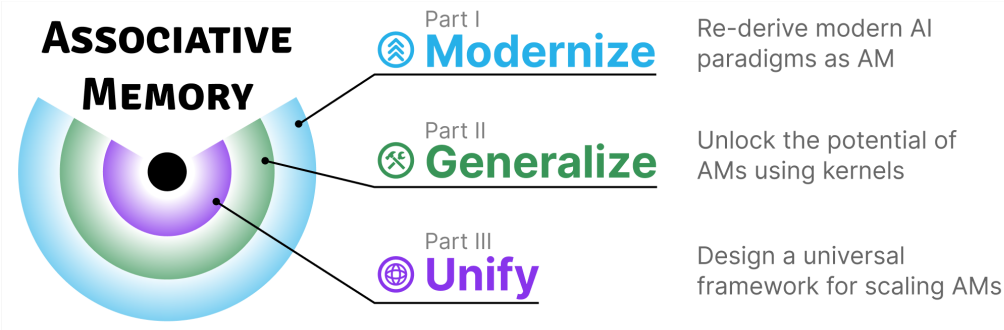

Figure 1.1: **Memory can be used as a design language for the next generation of AI architectures**. This thesis demonstrates that energy-based associative memory can scale to **modern** architectures and workloads (Part I), **generalize** to support compression and creative generation using kernels (Part II), and be **unified** into a compositional design framework (Part III) — providing the tooling for a paradigm where interpretable, efficient, and robust AI emerges from physical computation built around memory, energy, and dynamics.

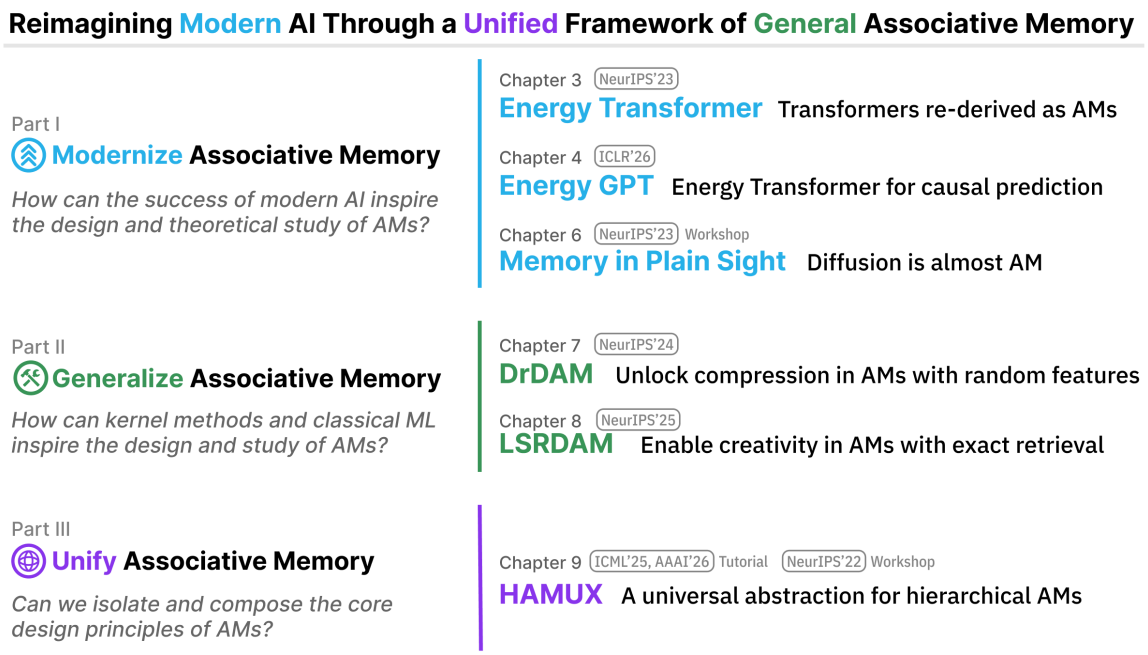

Figure 1.2: **Organizational structure of the thesis**. Part I **modernizes** AMs for contemporary architectures and workloads, Part II **generalizes** AMs through kernels to support compression and creative generation, and Part III **unifies** AMs into compositional design principles for building larger energy-based systems.

seems that modern AI has converged toward computation that resembles dynamical systems. Yet we lack a satisfying theoretical framework for designing these dynamics.

Iterative error-correction is a hallmark of early models of associative memory — a cognitive phenomenon that describes the universal human experiences of learning and remembering as an association game. Early models of *energy-based associative memory* (AM, for short) by Amari and Hopfield provided a principled mathematical description of memory as iterative error-correction and pattern completion [7, 8, 9, 10]. We now refer to this model as the *Hopfield Network*. The Hopfield Network revitalized interest in neural computation by describing it as a dynamical, fixed-point relaxation process that could be realized as an emergent property of large collections of neurons and synapses. In this model, AMs store information in learned weights (synapses) that shape an energy landscape, and stored information is retrieved by iteratively pushing noisy neuron states toward states of low-energy. However, the limited capacity and scalability of Hopfield Networks made many assume that explicit memory systems were impractical for modern machine learning.

This thesis argues that modern AI has unknowingly rediscovered the core principles of AM and physical memory systems, making the Hopfield Network more relevant to

the study and design of AI than ever before. By *using memory as a design language for the next generation of AI architectures*, we develop models from a paradigm where interpretable, efficient, and robust AI emerges from a theory of physical computation built around memory, energy, and dynamics. We do this while simultaneously answering two fundamental questions that once limited the field of AM research: whether explicit memory systems can *compress* information, and whether they can exhibit *genuinely novel* generations.

1.1 Thesis Overview

The central difficulty is that "memory" seems too simple to describe general intelligence. By definition, a memory system seems only capable of retrieving what it has seen, but intelligence requires more than retrieval: it requires abstraction, compression, adaptation, and generalization. To establish AMs as a viable foundation for modern AI, we must prove that these symptoms of intelligence arise naturally from proper formulations of AM across three dimensions: empirical evidence that AMs can scale to **modern** architectures and workloads (Part I), algorithmic solutions that allow AMs to **generalize** (Part II) to creative generations and compression, and **unified** design principles that establish tooling (Part III) to build more powerful AM architectures than those characterized in this work. This dissertation addresses these challenges through three complementary contributions, shown as an overview in Figure 1.2 and described in detail below:

1.1.1 Part I: Modernizing Associative Memories

Modern deep learning architectures are developed and understood as black-box feedforward networks, but they have somehow converged to paradigms that resemble the error-correcting dynamics of AMs. We seek to reformulate modern architectures using the principled theoretical foundation of AMs, while providing empirical evidence that they can scale to large data regimes.

Energy Transformer Chapter 3

We developed a novel AM architecture called *Energy Transformer* (ET) [11] which combines the energy of a Hopfield Network with a novel *attention energy* to create an architecture that looks strikingly like a standard transformer [3] trained to inpaint masked tokens. Unlike the

transformer, the forward pass through ET is *optimization*: a recurrent gradient descent on ET's energy (see Figure 1.3) that exposes the dynamics of a transformer-like architecture. ET achieves the performance of larger transformers on ImageNet inpainting and large-scale graph classification.

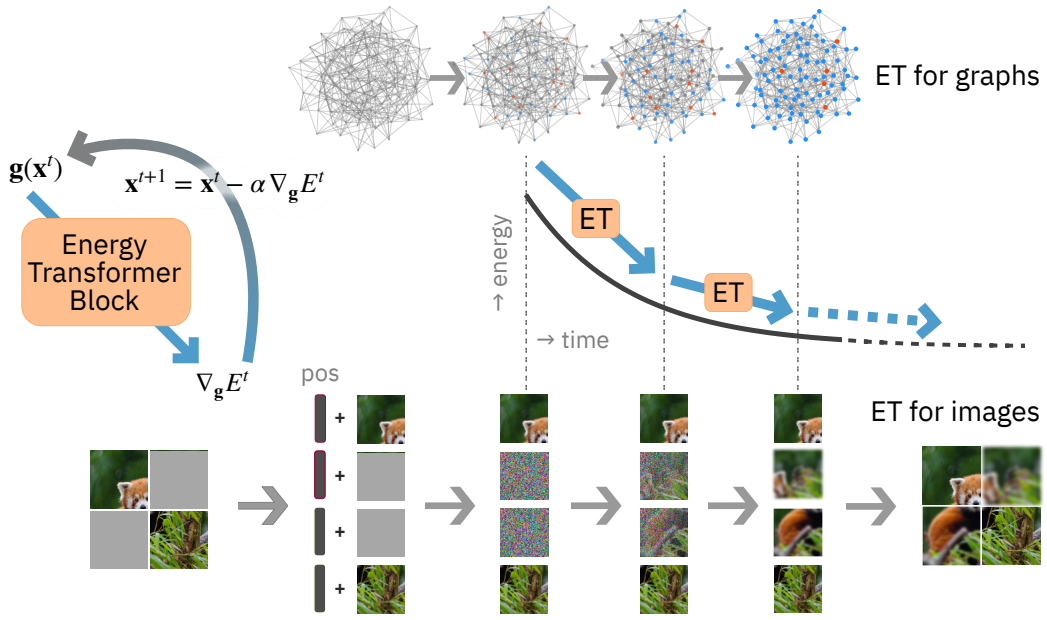

Figure 1.3: **ENERGY TRANSFORMER (ET) replaces a sequence of conventional transformer blocks with a single recurrent block dictated by a global energy function.** Token representations are updated according to a continuous-time differential equation, and each iteration reduces the energy of the set of tokens. The representations at or near the fixed point are then decoded to reconstruct masked tokens.

NRGPT: *An Energy-based Alternative for GPT* Chapter 4

We developed an energy-based alternative to GPT called *Energy GPT* (NRGPT) [12] that shows how GPT-style next-token prediction can be reframed as explicit, energy-based inference. Like ET, NRGPT replaces a purely feedforward interpretation of transformer computation with recurrent dynamics, where repeated applications of an NRGPT block update token states using gradients of attention and feedforward energies. Unlike ET, NRGPT optimizes the energy of each token independently. NRGPT performs well on algebraic ListOPS tasks, and OpenWebText language modeling, showing similar scaling laws to standard transformer blocks on the Shakespeare dataset.

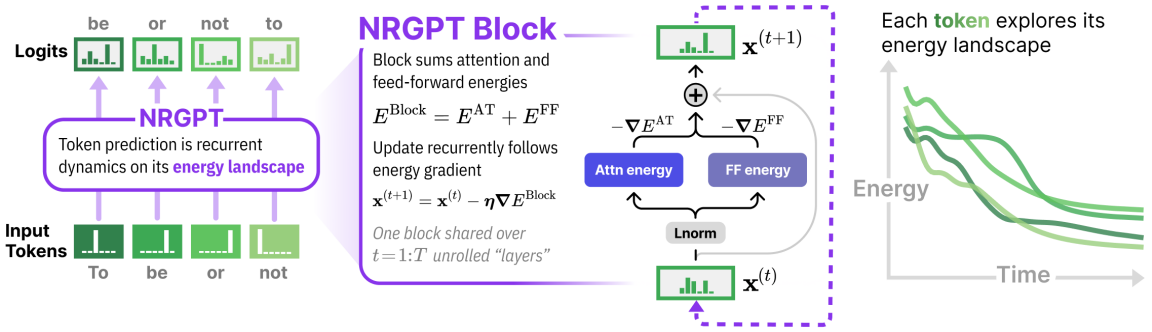

Figure 1.4: **NRGPT casts causal language modeling as an energy-based memory framework.** Like ET, the network is defined as the sum of an attention energy and a feedforward energy. Each token is transformed into the next token by exploring the energy landscape, producing a dynamical system where tokens can be thought of as particles moving on the network's energy landscape.

Memory in Plain Sight: A Survey of the Uncanny Resemblances between Diffusion Models and Associative Memories Chapter 5

We establish the mathematical connection between diffusion-model denoising and AM recall [13], uncovering their shared dynamics under deterministic flows while highlighting their structural divergences. We identify what standard diffusion models lack — such as a conservative score field and Lyapunov stability guarantees — to function as true physical computation systems of memory. This prescribes the architectural changes needed to design stable generative dynamics using the explicit physics of energy landscapes.

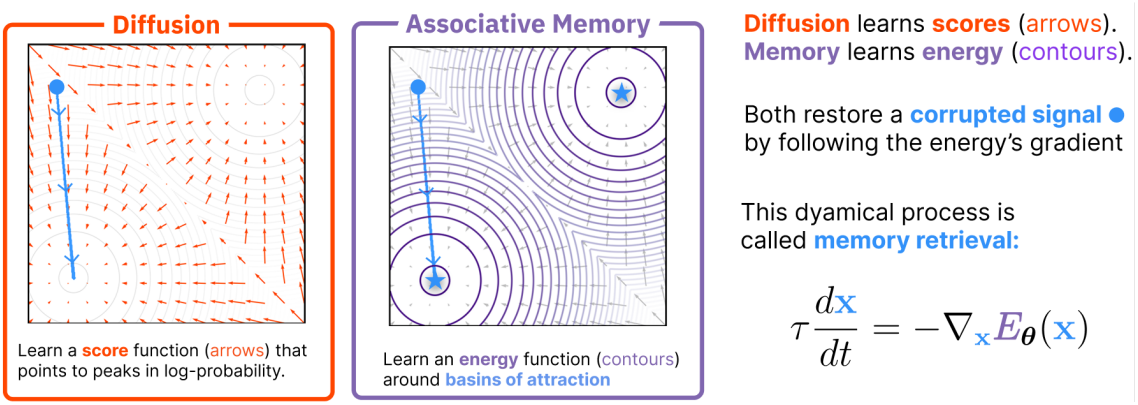

Figure 1.5: **MEMORY IN PLAIN SIGHT reveals the similarities between associative-memory recall and generative denoising** in diffusion models. Diffusion models learn to approximate scores (**orange** arrows) while associative memories learn explicit energy (**purple** contours). Both restore corrupted signals by minimizing energy, but only energy-based AMs describe a Lyapunov-stable dynamical system.

1.1.2 Part II: Generalizing Associative Memories

The modern AMs in Part I are built as compositions of Dense Associative Memories (DenseAMs), but DenseAMs have two limitations: their maximum memory capacity is tightly coupled to their parameter count, and their ability to exactly retrieve stored memories competes with their ability to generalize or create new generations. To resolve these bottlenecks, we leverage a striking yet little-known resemblance between DenseAMs and kernel-density estimation (KDE), where the energy of a DenseAM can be viewed as an unnormalized log-density constructed as a sum of kernels between stored patterns and an input. This connection suggests that we can uncover algorithmic solutions that allow explicit memory systems to compress information and produce novel memories.

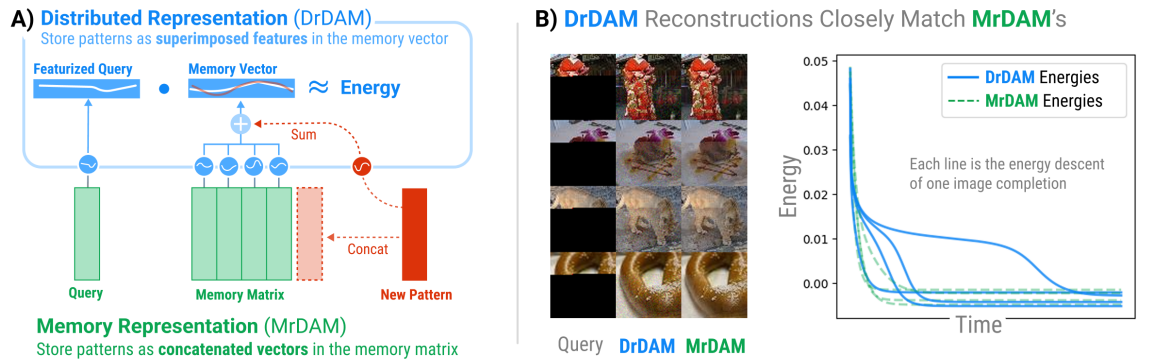

Figure 1.6: **DrDAM approximates both the energy and fixed-point dynamics of the traditional memory representation for DenseAMs while having a parameter space of constant size.** Instead of storing patterns as slots in a memory matrix, DrDAM superimposes patterns into a fixed-size memory tensor. In the distributed representation, adding new memories does not change the size of the memory tensor, while the approximate energy can still store and retrieve exponentially many memories.

DrDAM*: DenseAM through the Lens of Random Features* Chapter 6

We discover that using random features to approximate DenseAM energies decouples their memory capacity from their parameter count in what we call the *Distributed representation for DenseAM* (DrDAM) [14], effectively *compressing* patterns by distributing them across all available synaptic weights. Unlike traditional DenseAMs that require new synaptic connections for each stored pattern, our method enables variable capacity networks with fixed parameter counts to dynamically store streaming data.

LSRDAM*: DenseAM with Epanechnikov Energy* Chapter 7

Modifying DenseAM energies to use KDE-optimal kernels instead of standard RBF kernels can manufacture an unprecedented scale of *emergent* or spurious memories in the energy of a DenseAM, enabling a phenomenon akin to *creative hallucination* in explicit memory systems. We call this modified energy function the *Log-Sum-ReLU DenseAM* (LSRDAM) [15]. We observe that these emergent memories appear without compromising the network's ability to retrieve the original stored memories, simultaneously achieving both creativity and perfect storage.

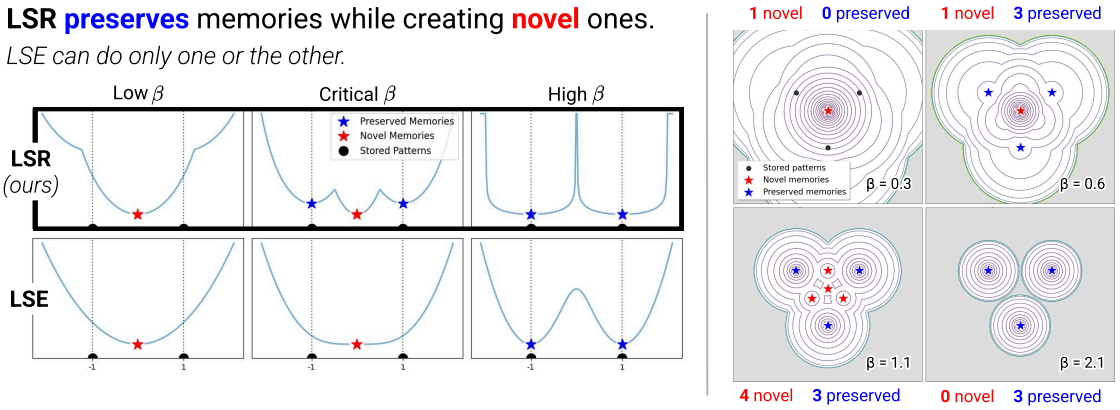

Figure 1.7: **LSRDAM can create abundantly more emergent memories than there are stored patterns**, under critical regimes of β. Left: 1D LSR vs. LSE energy landscape, where LSE is never capable of having more local minima than the number of stored patterns. Right: 2D LSR energy landscape, where increasing β creates novel local minima where basins intersect.

1.1.3 Part III: Unifying Associative Memories

Modern AMs now include many energy functions, but these models are often written as distinct systems with non-overlapping notation. This makes it difficult to compare existing architectures, combine their useful parts, or enable AMs to contain deep latent representations. After modernizing AMs and generalizing their kernel structure, we need better tooling to build AMs. We need to distill existing AM energy functions into a common design language that can be composed to form new architectures.

HAMUX*: A Universal Abstraction for Associative Memories* Chapter 8

We developed a core set of abstract computational primitives that can be composed to create all known AM architectures and unlock the development of new architectures with deep

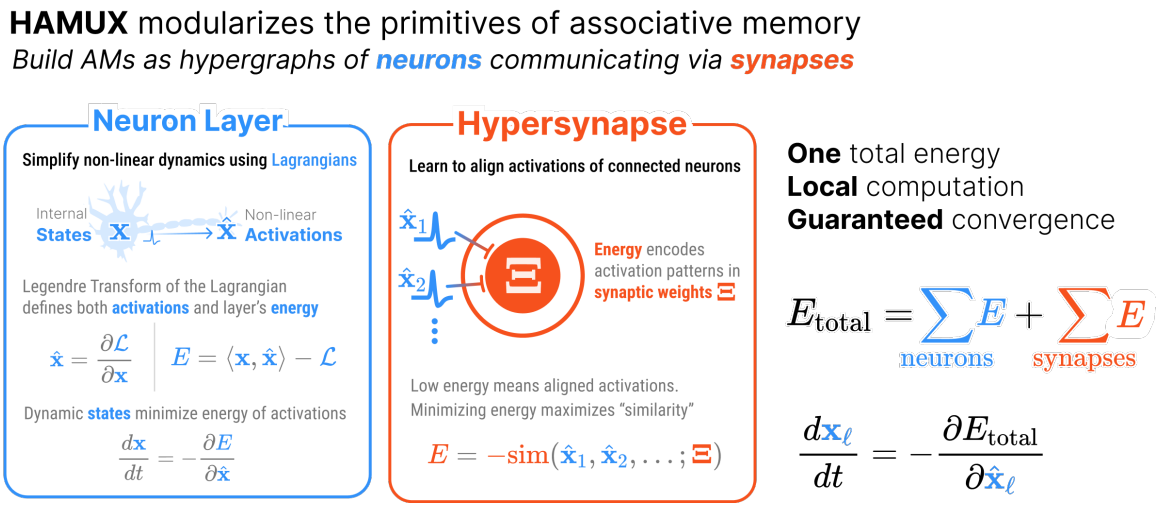

Figure 1.8: **HAMUX hypergraph diagrams are a graphical depiction of an AM whose total energy is the sum of neuron-layer and hypersynapse energies.** Inference is done recurrently, modeled by a system of differential equations where each neuron layer's hidden state updates to minimize the total energy. When all non-linearities are captured in the dynamic neurons, inference becomes a local computation that avoids differentiating through non-linearities.

latent representations. We call this abstraction **HAMUX**: the **H**ierarchical **A**ssociative **M**emory **U**ser e**X**perience [16, 17]. AMs built using the HAMUX framework have a template for achieving local computation and convergence guarantees under the framework's convexity and boundedness assumptions, while accommodating many different non-linear activations and undirected variants of modern architectural components such as convolutions, pooling, dropout, and attention. We show that this framework provides a foundation on which to ground future energy-based AM research.

1.2 Thesis Statement

Memory can serve as a design language for the next generation of AI architectures; by scaling energy-based associative memories (AMs) to **modern** architectures and workloads, **generalizing** them to support compression and creative generation using kernels, and **unifying** them into compositional principles, we establish a paradigm where interpretable, efficient, and robust AI emerges from physical computation built around memory, energy, and dynamics.

1.3 Research Contributions

This thesis makes AMs *useful* for modern AI by showing how AMs can be used to design practical modern architectures, reinterpret generative models, compress and extend memory systems with classical machine-learning tools, and build a broader research program around energy-based memory.

Groundbreaking algorithms that unlock practical utility for memory

- Pioneered novel algorithmic capabilities to make generative memories practical by ***uniting AMs and kernel theory*** from classical ML. For example, DRDAM equips dense memory with ***pattern compression*** capabilities and a ***one-shot memorization*** learning rule that avoids expensive optimization (Chapter 6). Meanwhile, LSRDAM solves the simultaneous memorization and generalization tradeoff of dense memories by leveraging theoretically optimal kernels (Chapter 7). LSRDAM received a ***Spotlight Paper*** award at NeurIPS (top 3% of 21k+ submissions) and an ***Oral Poster*** at the ICCV'25 workshop on Memory and Vision.

- Introduced AM algorithms that exemplify principled design for ***efficient architectures***, with ET using an order-of-magnitude ***fewer parameters*** than comparable transformers and emphasizing ***reusable weights*** and ***adaptable compute*** in its forward pass (Chapter 3). This architecture was awarded a hardware gift worth ∼$60k by NVIDIA's Academic Grant Program for advancing Robotics and Edge AI.

Fundamental theory to elevate physical computation to modern intelligent systems

- Developed a grounded theory for ***large, fixed-point computational systems***. Specifically, HAMUX describes a cohesive framework for constructing dynamical architectures with convergence guarantees (Chapter 8), while ET and NRGPT show that those architectures can scale to the performance and flexibility expected from large vision and language models (Chapters 3 and 4). ET has been covered in both ***Nature Reviews*** and ***Quanta Magazine***, and was a ***Spotlight Paper*** at the NeurIPS'23 workshop on Associative Memories & Hopfield Networks. HAMUX was a ***Spotlight Paper*** at the NeurIPS'22 workshop on Deep Learning & Differential Equations.

- Established a formal connection between denoising generation and memory recall, advancing a ***human-centric and physically grounded*** theory of memorization and creativity in generative models (Chapter 5). This work has been featured in two ***Quanta Magazine*** articles on the **physics of AI** and **diffusion models and creativity**.

Unified research agenda for accelerating innovations in generative memory

- This thesis brings together a suite of works that bridges performant AI architectures and physics by showing how transformer models, diffusion denoising, compressed memories, and hierarchical memories can all be described through energy-based memory dynamics. In particular, HAMUX establishes an explicit ***blueprint for scaling*** memory architectures using reusable components under physical computation constraints (Chapter 8). My research has ***catalyzed momentum*** for a vibrant AM community through ***two full tutorials*** and ***four dedicated workshops*** across flagship AI conferences such as ICML, AAAI, and NeurIPS. ET has been presented at over 100 conferences and lectures, including Los Alamos National Labs, Harvard CMSA, and the Aspen Center for Physics.

1.4 Impact

My research is making a significant impact on the research community and beyond:

- ET (Chapter 3) was **prominently featured in a *Nature Reviews*** article examining the renaissance of energy-based associative memory in AI, and has been presented at over **100 invited lectures**, including prestigious academic institutions such as Los Alamos National Labs, the Aspen Center for Physics, and Harvard CMSA. Both ET (Chapter 3) and MEMORY IN PLAIN SIGHT (Chapter 5) were **featured in a Quanta Magazine** cover story on the enduring legacy and transformative potential of Hopfield Networks.

- My research on memory has directly convened **over 1,500** students and researchers worldwide at flagship AI conferences, including **4 dedicated workshops** on AMs at NeurIPS, ICLR, and ICCV and **2 full tutorials** at ICML and AAAI.

- My work has been recognized by spotlight awards across top-tier AI venues: LSR-

DAM (Chapter 7) received the distinguished **Spotlight Paper Award** (top 3% of >21,000 submissions) at NeurIPS, the largest and most prestigious AI conference. ET was a **Spotlight Paper** at the NeurIPS'23 workshop on Associative Memory & Hopfield Networks, while HAMUX received a **Spotlight Paper** at the NeurIPS'22 workshop on Deep Learning & Differential Equations.

- My research on efficient AM algorithms was awarded a hardware gift worth ∼$60k by **NVIDIA's Academic Grant Program** for advancing Robotics and Edge AI.

CHAPTER 2
BACKGROUND AND RELATED WORK

2.1 Associative Memory Foundations

Associative memory is content-addressable storage, where a stored state is recovered from a related, partial, or corrupted cue [18, 9]. This contrasts with location-addressed random-access memory, where a supplied address specifies the storage location to read. Modern vector indexes and retrieval-augmented generation (RAG) are a form of associative memory that perform single-step retrieval of discrete vectors or documents by similarity search [19, 20].

The *energy-based associative memories* (AMs, for short) studied in this thesis take a different form, where retrieval is framed as iterative error correction via relaxation of an initial network state. This is the view of neural computation pioneered by the Hopfield Network [7, 9, 10], which posited that information storage and retrieval can emerge from the collective dynamics of many simple interacting units. During memory retrieval, the network state evolves toward a stable fixed point that matches the input query, as driven by an explicit energy landscape. The Hopfield network's update rules are readily adapted to integrated circuits, as the computation is carried out by the asynchronous parallel relaxation of a dynamical system whose local state updates decrease its global energy. This property makes the Hopfield Network an intriguing model of physical computation [9].

The AMs presented in this work are spiritual successors to the Hopfield Network.

2.2 From Classical Hopfield Networks to Dense Associative Memory

Early neural-network research supplied the artificial-neuron abstraction used by later memory models, beginning with McCulloch-Pitts threshold neurons [21]. Associative storage then became an explicit object of study in Willshaw networks, Amari's self-organizing nets, Cohen-Grossberg systems, Hopfield networks, and the statistical-mechanics analysis of Amit, Gutfreund, and Sompolinsky [22, 7, 23, 9, 24]. The classical Hopfield network [9, 10] is the standard starting point for this lineage because it is a simple and elegant energy-based model that makes storage, retrieval, and convergence explicit in a single energy function.

2.2.1 Classical Hopfield Networks (CHNs).

Consider a set of M binary or continuous stored patterns $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^{M}$ in an N-dimensional space. Let $\sigma(x_i)$ denote the neuron value associated with the internal state x_i. In the continuous CHN, σ is a monotone saturating nonlinearity such as $\tanh$; in the binary CHN, $\sigma_i \in \{-1, +1\}$ is the corresponding hard-threshold value [9, 10]. For $\boldsymbol{\sigma} = \sigma(\mathbf{x})$, a standard continuous CHN energy can be written as

$$E(\boldsymbol{\sigma}) = -\frac{1}{2}\boldsymbol{\sigma}^\top \boldsymbol{W}\boldsymbol{\sigma} + \sum_{i=1}^{N} \int_0^{\sigma_i} \sigma^{-1}(u)\, du, \tag{2.1}$$

where $\boldsymbol{W}$ is a symmetric synaptic weight matrix. Retrieval is described by an update rule that moves in the negative energy-gradient direction with respect to the neuron values:

$$\tau \frac{dx_i(t)}{dt} = -\frac{\partial E}{\partial \sigma_i} = \sum_{j=1}^{N} W_{ij}\sigma(x_j(t)) - x_i(t). \tag{2.2}$$

Because σ is monotone, this update preserves the energy-minimizing direction for the internal state dynamics and yields $\frac{d}{dt}E \leq 0$ under the model assumptions [10, 16]. The binary Hopfield network uses the corresponding hard-threshold update applied asynchronously:

$$\sigma_i^{(t+1)} = \operatorname{sign}\left(\sum_{j=1}^{N} W_{ij}\sigma_j^{(t)}\right), \tag{2.3}$$

which can be viewed as the zero-temperature limit of the same energy-minimizing principle. When the energy is bounded from below, these dynamics drive the state toward stable fixed points corresponding to local minima of the energy landscape [9, 10, 16]. Under this formulation, the network connects storage, retrieval, and error correction by mapping noisy inputs to clean stored patterns [13].

Adding stored patterns to a CHN. Adding memories to a CHN means writing their pairwise correlations into the synaptic matrix through a Hebbian outer-product rule:

$$W_{ij} = \frac{1}{N}\sum_{\mu=1}^{M} \xi_i^\mu \xi_j^\mu. \tag{2.4}$$

This storage rule is the sense in which additional patterns are stored by changing a fixed-size synaptic matrix rather than by appending separate retrieval documents or external memory slots [9, 24]. However, the storage capacity of CHNs is strictly limited to approximately $0.14N$ patterns, beyond which spurious states dominate and retrieval fails [9, 25].

2.2.2 Dense Associative Memories (DenseAMs).

The capacity limitations of the classical model motivated the development of Dense Associative Memories (DenseAMs), also known as the Modern Hopfield Network [26, 27, 28]. These models replace the quadratic energy function with stronger, non-quadratic nonlinearities to dramatically increase the storage capacity. A general formulation for the energy of a DenseAM is

$$E(\mathbf{x}) = -Q\left[\sum_{\mu=1}^{M} F\left(S\left[\boldsymbol{\xi}_\mu, \mathbf{g}(\mathbf{x})\right]\right)\right], \tag{2.5}$$

where $\mathbf{g}$ is a vector operation such as normalization, binarization, or an activation; S is a similarity function; F is a rapidly growing separation function; and Q is a monotone scaling function [26, 14, 15].

Adding stored patterns to a DenseAM. Adding memories to a DenseAM in the standard memory representation means appending each new stored pattern as a row of the memory matrix:

$$\boldsymbol{\Xi}^{(M+1)} = \begin{bmatrix} \boldsymbol{\Xi}^{(M)} \\ \boldsymbol{\xi}_{M+1}^{\top} \end{bmatrix}. \tag{2.6}$$

This storage rule is the sense in which additional patterns are stored by increasing the number of memory slots, and the exponential-capacity results are asymptotic statements about the large-memory-slot limit $M \to \infty$ [26, 27]. If $F(z) = z^n$ for integer $n > 2$, the storage capacity scales super-linearly with N [26]. Rectified power-law nonlinearities preserve monotone energy descent even when n is odd, so the energy remains a Lyapunov function under the appropriate retrieval dynamics [26, 16]. If $F(z) = \exp(z)$, the storage capacity scales exponentially, allowing the retrieval of up to $2^{N/2}$ patterns [27]. Subsequent work identified the connection between DenseAM retrieval dynamics and attention-like mechanisms in modern transformer architectures [28].

2.2.3 The Broader Associative Memory Ecosystem

The term "associative memory" is also used for many other kinds of content-addressable memory, including sparse distributed memories, Memory Networks, Key-Value Memories, Boltzmann Machines, and attention mechanisms in transformers [29, 30, 31, 32, 33, 34, 35, 3, 28]. This thesis focuses more narrowly on energy-based AMs, where retrieval is modeled as a fixed-point attractor process governed by a tractable scalar energy [9, 10, 16].

2.3 Energy-Based Modeling

The explicit energy functions of CHNs and DenseAMs place them within a broader and increasingly popular class of models called *energy-based models* (EBMs). EBMs are a flexible framework for modeling structured data by associating a scalar value, called the energy, with each configuration of variables [36, 37]. For a configuration $\mathbf{x}$, the model defines an energy function $E_{\boldsymbol{\theta}}(\mathbf{x})$ parameterized by $\boldsymbol{\theta}$ [36, 37]. In this formulation, lower energies are assigned to more compatible, stable, or desirable configurations, whereas higher energies correspond to less compatible or undesirable states [36]. Inference or sampling then searches for states assigned low energy, often using gradient-based or MCMC-style procedures such as Langevin dynamics [36, 38, 39, 37].

Gibbs or Boltzmann-style distributions express this ranking as a probability density:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{\exp\left(-E_{\boldsymbol{\theta}}(\mathbf{x})\right)}{Z_{\boldsymbol{\theta}}}, \tag{2.7}$$

where $Z_{\boldsymbol{\theta}}$ is the partition function that normalizes the density to integrate to one [40, 36]. Equivalently, negative log-density satisfies $-\log p_{\boldsymbol{\theta}}(\mathbf{x}) = E_{\boldsymbol{\theta}}(\mathbf{x}) + \log Z_{\boldsymbol{\theta}}$ [36, 37]. The scalar energy $E_{\boldsymbol{\theta}}(\mathbf{x})$ is often preferred to the normalized density $p_{\boldsymbol{\theta}}(\mathbf{x})$ because the partition function $Z_{\boldsymbol{\theta}}$ requires integrating or summing over the entire state space, which is often intractable [36, 37].

2.3.1 Explicit and Implicit EBMs.

Neural EBMs differ in whether they expose a scalar energy or only a vector field approximating its gradient [36, 39, 37]. Explicit EBMs model $E_{\boldsymbol{\theta}}(\mathbf{x})$ itself, either by directly outputting a scalar energy or by inducing one from another network output [39, 41]. Implicit EBMs instead learn a vector field related to $-\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x})$ without necessarily representing the

scalar energy [42, 43, 44, 45]. This distinction changes what the model predicts, but both views support generation by following an energy-descending vector field from an initial noisy state [36, 37, 16]. The explicit view is broader than generation: for prediction or classification, observed variables can be clamped while missing or output variables are updated by constrained energy minimization, a connection revisited in Chapter 8.

However, training explicit EBMs can be difficult, and running inference can be expensive [36, 39, 37, 16]. Likelihood-based training usually requires expectations under the model distribution, while inference often requires iterative gradient-based sampling or optimization through the energy network [39, 46, 37]. These difficulties motivate *score*-based generative models, which train a neural network $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})$ to approximate the score $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) \approx -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x})$ instead of exposing the scalar energy directly [42, 43, 44, 45]. This turns out to be much easier to train because the score has no dependence on the intractable partition function:

$$\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x}) \tag{2.8}$$

This relation lets a model approximate the vector field $-\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x})$ without exposing or normalizing the scalar energy itself [42, 43, 6, 44]. Diffusion and related flow-based generative models use these learned vector fields to transform noise into meaningful data samples [5, 47, 44, 48].

2.3.2 Other Forms of Iterative Neural Computation

More recently, looping and recurrent transformer architectures reuse computation across depth or segments by repeatedly applying an update rule [49, 50, 51]. Deep equilibrium models take a related fixed-point view by solving for an equilibrium of a neural transformation [52]. These architectures share the iterative flavor of EBM relaxation, but they typically do not provide a Lyapunov energy function for the update [10, 16].

2.4 Kernels and Statistical Views of Memory

Kernel methods compare data points using a similarity function $k(\mathbf{x}, \mathbf{x}')$, with support vector machines providing a canonical example of this paradigm [53]. Many kernels can be interpreted as inner products in implicit feature spaces, and random features approximate these kernels with finite-dimensional feature maps [54, 55]. In memory modeling, kernel views have been used to relate shallow memory architectures, Modern Hopfield Networks,

sparse Hopfield variants, and DenseAM-like retrieval rules [56, 57, 58, 59, 14, 16]. The kernel viewpoint is useful for DenseAMs because their energies aggregate similarities between a noisy query $\mathbf{x}$ and stored patterns $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^M$, which can often be read as a sum of kernels over memory slots [14, 15, 16].

Kernel density estimation (KDE) estimates a probability density by placing a kernel around each observed sample and summing their contributions [60, 61]. Given samples $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^M$, a KDE has the form

$$\hat{p}(\mathbf{x}) = \frac{1}{M}\sum_{\mu=1}^{M} k_h(\mathbf{x}, \boldsymbol{\xi}_\mu), \tag{2.9}$$

where k_h is a kernel with bandwidth h.

This makes KDE a useful statistical analogue for DenseAMs: after a negative log-transform, a kernel sum over stored patterns becomes an energy landscape over queries [14, 15, 16]. When the similarity is a Gaussian kernel, as is the case with traditional DenseAMs, this energy resembles the negative log of an unnormalized KDE [15, 16]. This statistical view provides the background for comparing DenseAM energy functions through kernel choice, including the Epanechnikov kernel and related results on KDE modes [60, 61, 62, 63, 64, 15].

2.5 Conclusion

Together, these perspectives define the technical vocabulary used throughout the dissertation. Associative memory supplies the storage and retrieval problem, energy-based modeling supplies the dynamical behavior, and KDE supplies a statistical lens for exploring DenseAM energy functions. The chapters that follow use this shared vocabulary to ask how AMs can be modernized, generalized, and ultimately treated as a unified algorithmic foundation for interpretable, efficient, and robust AI that naturally emerges from physical computation built around memory, energy, and dynamics.

Part I

MODERNIZING ASSOCIATIVE MEMORIES

Modern AI systems achieve many of the capabilities that AMs were originally designed to explain: completing missing information, correcting corrupted inputs, and moving representations toward more plausible states. Yet these systems are usually implemented using feedforward architectures rather than as dynamical memory systems with explicit energies. Part I asks whether the success of modern architectures can be *recast as evidence for AM* rather than as evidence against it.

This part begins by developing the ENERGY TRANSFORMER (ET), an AM whose recurrent energy minimization resembles the computation of a transformer block. It then extends this perspective to causal language modeling through the ENERGY GPT (NRGPT), showing how GPT-style next-token prediction can also be formulated as energy-based inference. We finally comment on how to reframe generative denoising as memory retrieval through MEMORY IN PLAIN SIGHT, connecting diffusion modeling to AM dynamics. Together, these chapters establish that AMs can scale toward the architectural patterns that define current AI systems.

Chapter 3

ENERGY TRANSFORMER. Benjamin Hoover*, Yuchen Liang*, Bao Pham*, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed J. Zaki, and Dmitry Krotov. *Advances in Neural Information Processing Systems (NeurIPS), 2023*. PDF

Chapter 4

NRGPT: An Energy-based Alternative for GPT. Nima Dehmamy*, Benjamin Hoover*, Bishwajit Saha*, Leo Kozachkov, Jean-Jacques Slotine, and Dmitry Krotov*. *International Conference on Learning Representations (ICLR), 2026*. PDF

Chapter 5

MEMORY IN PLAIN SIGHT: A Survey of the Uncanny Resemblances between Diffusion Models and Associative Memories. Benjamin Hoover, Hendrik Strobelt, Dmitry Krotov, Judy Hoffman, Zsolt Kira, and Duen Horng Chau. *arXiv preprint arXiv:2309.16750, 2023*. PDF

CHAPTER 3
ENERGY TRANSFORMER

Our work in this chapter combines aspects of three promising paradigms in machine learning: namely, attention mechanism, energy-based models, and associative memory. Attention is the power-house driving modern deep learning successes, but it lacks clear theoretical foundations. Energy-based models allow a principled approach to discriminative and generative tasks, but the design of the energy functional is not straightforward. At the same time, Dense Associative Memory models or Modern Hopfield Networks have a well-established theoretical foundation, and allow an intuitive design of the energy function. We propose a novel architecture, called the ENERGY TRANSFORMER (or ET for short), that uses a sequence of attention layers that are purposely designed to minimize a specifically engineered energy function, which is responsible for representing the relationships between the tokens. In this work, we introduce the theoretical foundations of ET, explore its empirical capabilities using the image completion task, and obtain strong quantitative results on the graph anomaly detection and graph classification tasks.

3.1 Introduction

Modern transformer architectures have become pervasive models in various domains of machine learning, including language, vision, and audio processing. Every transformer block uses four fundamental operations: attention, feed-forward multi-layer perceptron (MLP), residual connection, and layer normalization. Different variations of transformers result from combining these four operations in various ways. For instance, [65] proposes to frontload additional attention operations and backload additional MLP layers in a sandwich-like manner instead of interleaving them, [66] prepends an MLP layer before the attention in each transformer block, [67] uses neural architecture search methods to evolve even more sophisticated transformer blocks, and so on. Various methods exist to approximate the attention operation, multiple modifications of the norm operation, and connectivity of the block; see, for example, [68] for a taxonomy of different models. At present, however, the search for new transformer architectures is driven mostly by empirical evaluations, and the theoretical principles behind this growing list of architectural variations are missing.

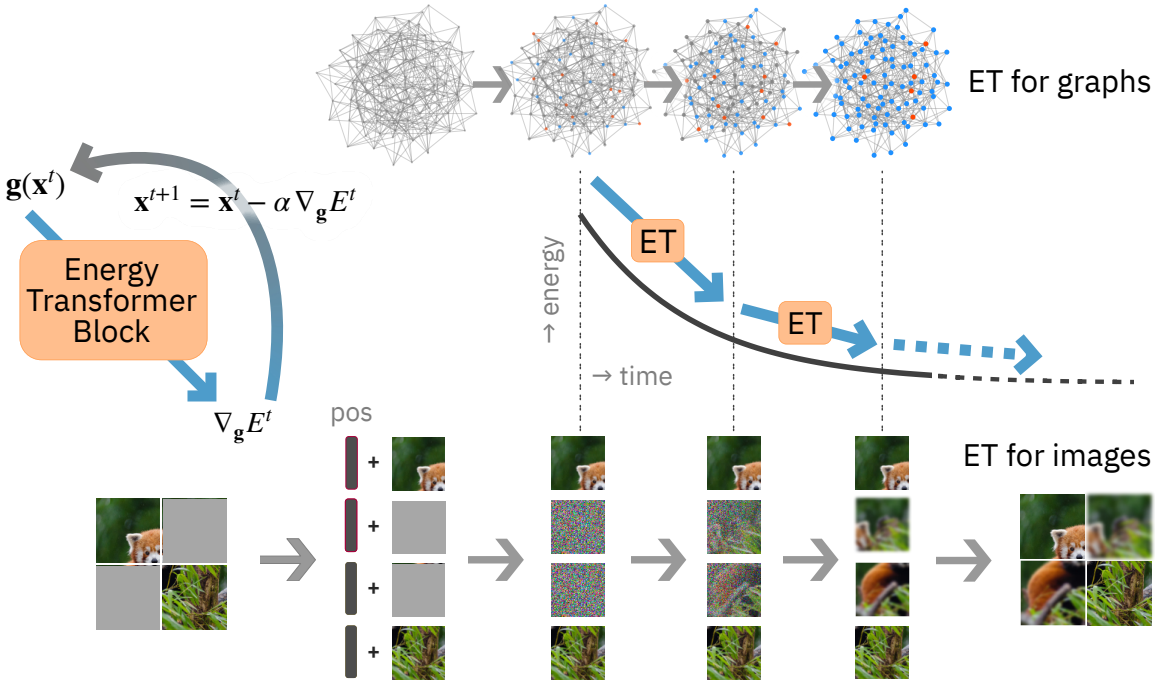

Figure 3.1: Overview of the ENERGY TRANSFORMER (ET). Instead of a sequence of conventional transformer blocks, a single recurrent ET block is used. The operation of this block is dictated by the global energy function. The token representations are updated according to a continuous time differential equation with the time-discretized update step $\alpha = dt/\tau$. On the image domain, images are split into non-overlapping patches that are linearly encoded into tokens with added learnable positional embeddings (POS). Some patches are randomly masked. These tokens are recurrently passed through ET, and each iteration reduces the energy of the set of tokens. The token representations at or near the fixed point are then decoded using the decoder network to obtain the reconstructed image. The network is trained by minimizing the mean squared error loss between the reconstructed image and the original image. On the graph domain, the same general pipeline is used. Each token represents a node, and each node has its own positional encoding. The token representations at or near the fixed point are used for the prediction of the anomaly status of each node, or the graph label.

Additionally, the computational role of the four elements remains the subject of discussions. Originally, [3] emphasized attention as the most important part of the transformer block, arguing that the learnable long-range dependencies are more powerful than the local inductive biases of convolutional networks. On the other hand more recent investigations [69] argue that the entire transformer block is important. The "correct" way to combine the four basic operations inside the block remains unclear, as does an understanding of the core computational function of the entire block and each of its four elements.

On the other hand, energy-based Associative Memory (AM) models, which trace their roots to Hopfield Networks [9, 10], have been gaining popularity in the machine learning community thanks to theoretical advancements pertaining to their memory storage capacity and novel architectural modifications [70]. Specifically, it has been shown that increasing the sharpness of the activation functions can lead to super-linear [26] and even exponential [27] memory storage capacity for these models, which is important for machine learning applications. This new class of Hopfield Networks is called Dense Associative Memories (DenseAMs) or Modern Hopfield Networks. Study [28] additionally describes how the attention mechanism in transformers is closely related to a special model of this family with the $\mathrm{softmax}$ activation function.

There are high-level conceptual similarities between transformers and DenseAMs, since both architectures are designed for some form of denoising of the input. Standard transformer models are typically pre-trained on a masked-token task, e.g., in the domain of Natural Language Processing (NLP), certain tokens in the sentence are masked and the model predicts the masked tokens. DenseAM models are designed for completing the incomplete patterns. They can be trained in a self-supervised way by predicting the masked parts of the pattern, or denoising the pattern.

There are also high-level differences between the two approaches. AMs are recurrent networks with a global energy function so that the network dynamics converges to a fixed point attractor state corresponding to a local minimum of the energy function. Standard transformer architectures are typically not described as dynamical systems at all. Rather, they are thought of as feed-forward networks built from the four computational elements discussed above. Even if one thinks about them as dynamical systems with tied weights, e.g., [52], there is no reason to expect that their dynamics converge to a fixed point attractor (see the discussion in [71]).

Additionally, a recent study [72] uses a form of Majorization-Minimization algorithms

[73] to interpret the forward path in the transformer block as an optimization process. This interpretation requires imposing certain constraints on the operations inside the block, and attempting to find an energy function that describes the constrained block. We take a complementary approach by using the intuition developed in AM models to *start* with an energy function that is engineered for the problem of interest. The optimization process and the resulting architecture of the transformer block in our approach is a *consequence* of this specifically chosen energy function.

Concretely, we use the recent theoretical advancements and architectural developments in DenseAMs to design an energy function tailored to route the information between the tokens. The goal of this energy function is to represent the relationships between the semantic contents of tokens describing a given data point (e.g., the relationships between the contents of the image patches in the vision domain, or relationships between the nodes' attributes in the graph domain). The core mathematical idea of our approach is that the sequence of these unusual transformer blocks, which we call the ENERGY TRANSFORMER (ET), minimizes this global energy function. Thus, the sequence of conventional transformer blocks is replaced with a single ET block, which iterates the token representations until they converge to a fixed point attractor state. In the image domain, this fixed point corresponds to the completed image with masked tokens replaced by plausible auto-completions of the occluded image patches. In the graph domain, the fixed point reveals the anomaly status of a node given its neighbors, see Figure 3.1, or the graph label. The energy function in our ET block is designed with the goal to describe the *relationships between the tokens*. Examples of relationships in the image domain are: straight lines tend to continue through multiple patches, given a face with one eye being masked the network should inpaint the missing eye, etc. In the graph domain, these are the relationships between the features of nodes; or features of nodes and graph label in graph classification.

The core mathematical principle of the ET block – the existence of the global energy function – dictates strong constraints on the possible operations inside the block, the order in which these operations are executed in the inference pass, and the symmetries of the weights in the network. As a corollary of this theoretical principle, the attention mechanism of ET is different from the attention mechanism commonly used in feed-forward transformers [3]. Lastly, our network may be viewed as an example of a broader class of Energy-Based Models [36] (EBMs), introduced previously in this thesis and frequently discussed in the AI community. The proposed model is defined through the specific choice of the energy

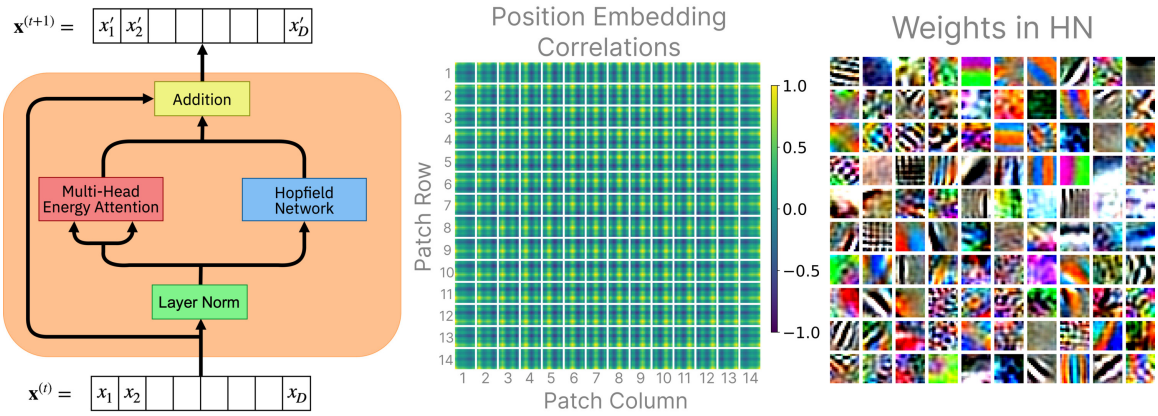

Figure 3.2: **Left**: Inside the ET block. The input token $\mathbf{x}^{(t)}$ passes through a sequence of operations and gets updated to produce the output token $\mathbf{x}^{(t+1)}$. The operations inside the ET block are carefully engineered so that the entire network has a global energy function, which decreases with time and is bounded from below. In contrast to conventional transformers, the ET-based analogs of the attention module and the feed-forward MLP module are applied in parallel as opposed to consecutively. **Center**: The cosine similarity between the learned position embedding of each patch and every other patch. In each cell, the brightest patch indicates the cell of consideration. **Right**: 100 selected memories stored in the HN memory matrix, visualized by the decoder as 16x16 RGB image patches. This visualization is unique to our model, as traditional transformers cannot guarantee image representations in the learned weights.

function, which, on the one hand, is suitable for the computational task of interest, and, on the other hand, results in optimization equations closely related to the forward pass in feed-forward transformers.

3.2 Energy Transformer Block

We now introduce the theoretical framework of the ET network. For clarity of presentation, we use language associated with the image domain. For the graph domain, one should think about "image patches" as nodes on the graph.

The overall pipeline is similar to the Vision Transformer networks (ViTs) [74] and is shown in Figure 3.1. An input image is split into non-overlapping patches. After passing these patches through the encoder and adding the positional information, the semantic content of each patch and its position is encoded in the token x_{iA}. In the following the indices $i, j, k = 1...D$ are used to denote the token vector's elements, indices $A, B, C = 1...N$ are used to enumerate the patches and their corresponding tokens. It is helpful to think about each image patch as a physical particle, which has a complicated internal state described

by a D-dimensional vector $\mathbf{x}_A$. This internal state describes the identity of the particle (representing the pixels of each patch), and the particle's positional embedding (the patch's location within the image). The ET block is described by a continuous time differential equation, which describes interactions between these particles. Initially, at $t = 1$ the network is given a set containing two groups of particles corresponding to open and masked patches. The "open" particles know their identity and location in the image. The "masked" particles only know where in the image they are located, but are not provided the information about what image patch they represent. The goal of ET's non-linear dynamics is to allow the masked particles to find an identity consistent with their locations and the identities of open particles. This dynamical evolution is designed so that it minimizes a global energy function, and is guaranteed to arrive at a fixed point attractor state. The identities of the masked particles are considered to be revealed when the dynamical trajectory reaches the fixed point. Thus, the central question is: how can we design the energy function that accurately captures the task that the ENERGY TRANSFORMER needs to solve?

The masked particles' search for identity is guided by two pieces of information: identities of the open particles, and the general knowledge about what patches are in principle possible in the space of all possible images. These two pieces of information are described by two contributions to the ET's energy function: the energy-based attention and the Hopfield Network, respectively, for reasons that will become clear in the next sections. Below we define each element of the ET block in the order they appear in Figure 3.2.

3.2.1 Layer Norm

Each token, or a particle, is represented by a vector $\mathbf{x} \in \mathbb{R}^D$. At the same time, most of the operations inside the ET block are defined using a layer-normalized token representation

$$g_i = \gamma \frac{x_i - \bar{x}}{\sqrt{\frac{1}{D}\sum_j \left(x_j - \bar{x}\right)^2 + \varepsilon}} + \delta_i, \qquad \text{where} \qquad \bar{x} = \frac{1}{D}\sum_{k=1}^{D} x_k \tag{3.1}$$

The scalar γ and the vector elements δ_i are learnable parameters, ε is a small regularization constant. Importantly, this operation can be viewed as an activation function for the neurons and can be defined as a partial derivative of the Lagrangian function (this is discussed later

in Equation (8.3), and also in [75, 76, 77])

$$L = D\gamma\sqrt{\frac{1}{D}\sum_j \left(x_j - \bar{x}\right)^2 + \varepsilon} \; + \; \sum_j \delta_j x_j, \qquad \text{so that} \qquad g_i = \frac{\partial L}{\partial x_i} \tag{3.2}$$

3.2.2 Multi-Head Energy Attention

The first contribution to the ET's energy function is responsible for exchanging information between the particles (tokens). Similarly to the conventional attention mechanism, each token generates a pair of queries and keys (ET does not have a separate value matrix; instead the value matrix is a function of keys and queries). The goal of the energy-based attention is to evolve the tokens in such a way that the keys of the open patches are aligned with the queries of the masked patches in the internal space of the attention operation. Below we use index $\alpha = 1...Y$ to denote elements of this internal space, and index $h = 1...H$ to denote different heads of this operation. With these notations the energy-based attention operation is described by the following energy function:

$$E^{\text{ATT}} = -\frac{1}{\beta}\sum_{h=1}^{H}\sum_{C=1}^{N} \log\left(\sum_{B\neq C} \exp\left(\beta A_{hBC}\right)\right) \tag{3.3}$$

where the attention matrix A_{hBC} is computed from query and key tensors as follows:

$$\begin{aligned} A_{hBC} &= \sum_{\alpha} K_{\alpha hB}\, Q_{\alpha hC}, \qquad && \mathbf{A} \in \mathbb{R}^{H\times N\times N} \\ K_{\alpha hB} &= \sum_{j} W^{K}_{\alpha hj}\, g_{jB}, \qquad && \mathbf{K} \in \mathbb{R}^{Y\times H\times N} \\ Q_{\alpha hC} &= \sum_{j} W^{Q}_{\alpha hj}\, g_{jC}, \qquad && \mathbf{Q} \in \mathbb{R}^{Y\times H\times N} \end{aligned} \tag{3.4}$$

and the tensors $\mathbf{W}^K \in \mathbb{R}^{Y\times H\times D}$ and $\mathbf{W}^Q \in \mathbb{R}^{Y\times H\times D}$ are learnable parameters.

From the computational perspective each patch generates two representations: query (given the position of the patch and its current content, where in the image should it look for the prompts on how to evolve in time?), and key (given the current content of the patch and its position, what should be the contents of the patches that attend to it?). The log-sum energy function in Equation (3.3) is minimal when for every patch in the image its queries are aligned with the keys of a small number of other patches connected by the attention map. Different heads (index h) contribute to the energy additively.

3.2.3 Hopfield Network Module

The next step of the ET block, which we call the Hopfield Network (HN), is responsible for ensuring that the token representations are consistent with what one expects to see in realistic images. The energy of this sub-block is defined as:

$$E^{\text{HN}} = -\sum_{B=1}^{N}\sum_{\mu=1}^{K} G\Big(\sum_{j=1}^{D}\xi_{\mu j}\, g_{jB}\Big), \qquad \xi \in \mathbb{R}^{K\times D} \tag{3.5}$$

where $\xi_{\mu j}$ is a set of learnable weights (memories in the Hopfield Network), and $G(\cdot)$ is an integral of the activation function $r(\cdot)$, so that $G(\cdot)' = r(\cdot)$. Depending on the choice of the activation function this step can be viewed either as a classical continuous Hopfield Network [10] if the activation function grows slowly (e.g., $r(\cdot)$ =ReLU), or as a modern continuous Hopfield Network [26, 28, 75] if the activation function is sharply peaked around the memories (e.g., $r(\cdot)$ = power, or softmax). The HN sub-block is analogous to the feed-forward MLP step in the conventional transformer block but requires that the weights of the projection from the token space to the hidden neurons' space to be the same (transposed matrix) as the weights of the subsequent projection from the hidden space to the token space. Thus, the HN module here is an MLP with shared weights that is *applied recurrently*. The energy contribution of this block is low when the token representations are aligned with some rows of the matrix ξ, which represent memories, and high otherwise.

3.2.4 Dynamics of Token Updates

The inference pass of the ET network is described by the continuous time differential equation, which minimizes the sum of the two energies described above

$$\tau\frac{dx_{iA}}{dt} = -\frac{\partial E}{\partial g_{iA}}, \qquad \text{where} \qquad E = E^{\text{ATT}} + E^{\text{HN}} \tag{3.6}$$

Here x_{iA} is the token representation (input and output from the ET block), and g_{iA} is its layer-normalized version. The first energy is low when each patch's queries are aligned with the keys of its neighbors. The second energy is low when each patch has content consistent with the general expectations about what an image patch should look like (memory slots of the matrix ξ). The dynamical system in Equation (3.6) finds a trade-off between these two desirable properties of each token's representation. For numerical evaluations, Equation (3.6)

is discretized in time.

To demonstrate that the dynamical system in Equation (3.6) minimizes the energy, consider the temporal derivative

$$\frac{dE}{dt} = \sum_{i,j,A} \frac{\partial E}{\partial g_{iA}} \frac{\partial g_{iA}}{\partial x_{jA}} \frac{dx_{jA}}{dt} = -\frac{1}{\tau} \sum_{i,j,A} \frac{\partial E}{\partial g_{iA}} M^A_{ij} \frac{\partial E}{\partial g_{jA}} \leq 0 \tag{3.7}$$

The last inequality sign holds if the symmetric part of the matrix

$$M^A_{ij} = \frac{\partial g_{iA}}{\partial x_{jA}} = \frac{\partial^2 L}{\partial x_{iA} \partial x_{jA}} \tag{3.8}$$

is positive semi-definite (for each value of index A). The Lagrangian in Equation (3.2) satisfies this condition.

3.2.5 Relationship to Modern Hopfield Networks and Conventional Attention

One of the theoretical contributions of our work is the design of the energy attention mechanism and the corresponding energy function in Equation (3.3). Although heavily inspired by prior work on Modern Hopfield Networks, our approach is fundamentally different from it. Our energy function in Equation (3.3) may look somewhat similar to the energy function of a continuous Hopfield Network with the $\mathrm{softmax}$ activation function. The main difference, however, is that in order to use Modern Hopfield Networks recurrently (as opposed to applying their update rule only once) the keys must be constant parameters (called memories in the Hopfield language). In contrast, in our energy attention network the keys are *dynamical variables* that evolve in time with the queries.

To emphasize this further, it is instructive to write explicitly the ET attention contribution to the update dynamics in Equation (3.6). It is given by (for clarity, assume only one head of attention):

$$-\frac{\partial E^{\text{ATT}}}{\partial g_{iA}} = \sum_{C \neq A} \sum_{\alpha} W^Q_{\alpha i} K_{\alpha C} \underset{C}{\mathrm{softmax}}\Big(\beta \sum_{\gamma} K_{\gamma C} Q_{\gamma A}\Big) + W^K_{\alpha i} Q_{\alpha C} \underset{A}{\mathrm{softmax}}\Big(\beta \sum_{\gamma} K_{\gamma A} Q_{\gamma C}\Big)$$

In both terms the $\mathrm{softmax}$ normalization is done over the token index of the keys, which is indicated by the subscript in the equation. The first term in this formula is the conventional attention mechanism [3] with the value matrix equal to $\mathbf{V} = (\mathbf{W}^Q)^T \mathbf{K} = \sum_\alpha W^Q_{\alpha i} K_{\alpha C}$. The second term is the brand new contribution that is missing in the original attention

mechanism. The presence of this second term is crucial to make sure that the dynamical system in Equation (3.6) minimizes the energy function if applied recurrently. This second term is the main difference of our approach compared to the Modern Hopfield Networks. The same difference applies compared to the other recent proposals [72].

Lastly, we want to emphasize that our ET block contains two different kinds of Hopfield Networks acting in parallel, see Figure 3.2. The first one is the energy attention module, which is inspired by, but not identical to, Modern Hopfield Networks. The second one is the "Hopfield Network" module, which can be either a classical or Modern Hopfield Network. These two should not be confused.

For completeness, the contribution of the "Hopfield Network" module to the update equation in Equation (3.6) can be written as

$$-\frac{\partial E^{\text{HN}}}{\partial g_{iA}} = \sum_{\mu=1}^{K} \xi_{\mu i} G'\Big(\sum_{j=1}^{D} \xi_{\mu j} g_{jA}\Big) \tag{3.9}$$

which is applied to every token individually (there no mixing of different tokens).

3.3 Qualitative Inspection of the ET framework on ImageNet

We trained[1] the ET network on the masked image completion task using ImageNet-1k dataset [78]. Each image was broken into non-overlapping patches of 16x16 RGB pixels, which were projected with a single affine encoder into the token space. Half of these tokens were "masked", by replacing them with a learnable MASK token. A distinct learnable position encoding vector was added to each token. Our ET block then processes all tokens recurrently for T steps. The token representations after T steps are passed to a simple linear decoder (consisting of a layer norm and an affine transformation). The loss function is the standard MSE loss on the occluded patches. See Appendix A.1 for more details on the implementation and the hyperparameters.

Examples of occluded/reconstructed images (unseen during training) are shown in Figure 3.3. In general, our model learns to perform the task very well, capturing the texture in dog fur (column 3) and understanding meaningful boundaries of objects. However, we observe that our single ET block struggles to understand some global structure, e.g., failing to capture both eyes of the white dog (column 5) and completing irregular brick patterns

[1] The code is available: https://github.com/bhoov/energy-transformer-jax.

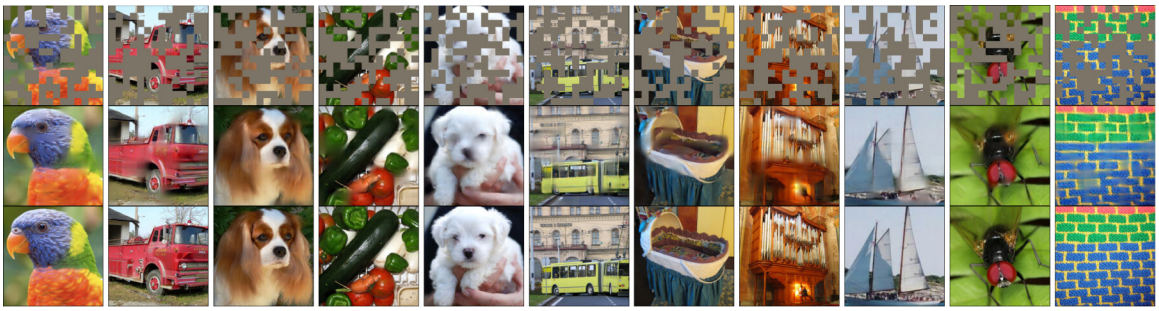

Figure 3.3: Reconstruction examples of our Energy Transformer using images from the ImageNet-1k validation set. *Top row:* input images where 50% of the patches are masked with the learned MASK token. *Middle row*: output reconstructions after 12 time steps. *Bottom row*: original images.

in the name of extending the un-occluded borders (last column). We additionally inspect the positional encoding vectors associated with every token, Figure 3.2, where the model learns a locality structure in the image plane that is very similar to the original ViT [74]. The positional embedding of each image patch has learned higher similarity values for neighboring tokens than for distant tokens.

Our network is unique compared to standard ViTs in that the iterative dynamics only *move* tokens around in the same space from which the final fixed point representation can be decoded back into the image plane. This functionality makes it possible to visualize essentially any *token representation*, *weight*, or *gradient of the energy* directly in the image plane. This feature is highly desirable from the perspective of interpretability, since it makes it possible to track the updates performed by the network directly in the image plane as the computation unfolds in time. In Figure 3.2 this functionality is used for inspecting the learned weights of the HN module directly in the image plane. According to our theory, these weights should represent basis vectors in the space of all possible image patches. These learned representations look qualitatively similar to the representations typically found in networks trained on image datasets, e.g., [79].

We additionally visualize the gradients of the energy function (which are equal to the token updates, see Equation (3.6)) of both ATTN block and the HN block, see Figure 3.4. Early in time, almost all signal to the masked tokens comes from the ATTN block, which routes information from the open patches to the masked ones; no meaningful signal comes from the HN block to the masked patch dynamics. Later in time we observe a different phenomenon: almost all signal to masked tokens comes from the HN module while ATTN contributes a blurry and uninformative signal. Thus, the attention module is crucial early in

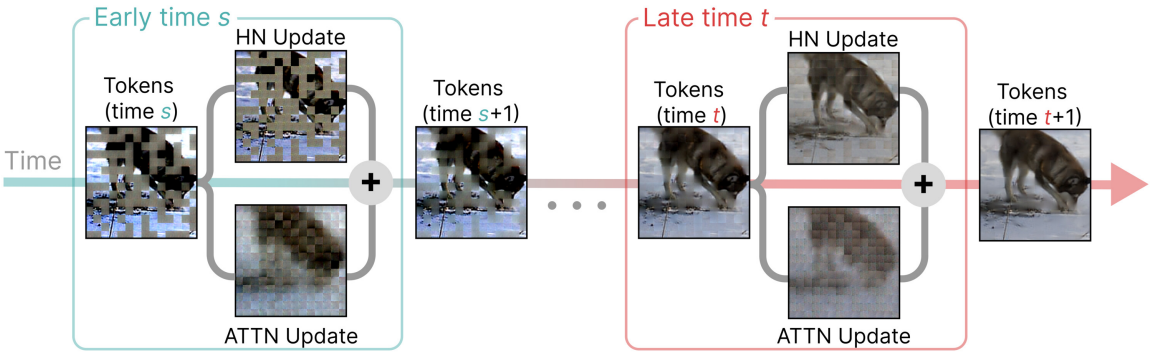

Figure 3.4: Token representations and gradients are visualized using the decoder at different times during the dynamics. The Energy Attention (ATTN) block contributes general structure information to the masked patches at *earlier* time steps, whereas the Hopfield Network (HN) significantly sharpens the quality of the masked patches at *later* time steps.

Table 3.1: Performance on Yelp, Amazon, T-Finance, and T-Social datasets with different training ratios. Following [80], mean and standard deviation over 5 runs with different train/dev/test split are reported (standard deviations are only included if they are available in the prior work). Best results are in **bold**. Our model is state of the art or near state of the art on every category.

	Datasets	Split	GraphConsis	CAREGNN	PC-GNN	BWGNN	MLP	GT	ET (Ours)
Macro-F1	Yelp	1%	$56.8_{\pm 2.8}$	$62.1_{\pm 1.3}$	$59.8_{\pm 1.4}$	$61.1_{\pm 0.4}$	$53.9_{\pm 0.2}$	$61.7_{\pm 0.4}$	$\mathbf{63.0_{\pm 0.6}}$
		40%	$58.7_{\pm 2.0}$	$63.3_{\pm 0.9}$	$63.0_{\pm 2.3}$	$71.0_{\pm 0.9}$	$57.5_{\pm 0.8}$	$68.7_{\pm 0.4}$	$\mathbf{71.5_{\pm 0.1}}$
	Amazon	1%	$68.5_{\pm 3.4}$	$68.7_{\pm 1.6}$	$79.8_{\pm 5.6}$	$\mathbf{90.9_{\pm 0.7}}$	$74.6_{\pm 1.2}$	$88.6_{\pm 0.5}$	$89.3_{\pm 0.7}$
		40%	$75.1_{\pm 3.2}$	$86.3_{\pm 1.7}$	$89.5_{\pm 0.7}$	$92.2_{\pm 0.4}$	$79.1_{\pm 1.2}$	$91.7_{\pm 0.8}$	$\mathbf{92.8_{\pm 0.3}}$
	T-Finance	1%	71.7	73.3	62.0	84.8	61.0	81.5	$\mathbf{85.1_{\pm 1.0}}$
		40%	73.4	77.5	63.1	86.8	70.5	83.6	$\mathbf{88.2_{\pm 1.0}}$
	T-Social	1%	52.4	55.8	51.1	75.9	50.0	64.3	$\mathbf{79.1_{\pm 0.7}}$
		40%	56.5	56.2	52.1	**83.9**	50.3	68.2	$83.5_{\pm 0.4}$
AUC	Yelp	1%	$66.4_{\pm 3.4}$	$75.0_{\pm 3.8}$	$\mathbf{75.4_{\pm 0.9}}$	$72.0_{\pm 0.5}$	$59.8_{\pm 0.4}$	$72.5_{\pm 0.6}$	$73.2_{\pm 0.8}$
		40%	$69.8_{\pm 3.0}$	$76.1_{\pm 2.9}$	$79.8_{\pm 0.1}$	$84.0_{\pm 0.9}$	$66.5_{\pm 1.0}$	$81.9_{\pm 0.5}$	$\mathbf{84.9_{\pm 0.3}}$
	Amazon	1%	$74.1_{\pm 3.5}$	$88.6_{\pm 3.5}$	$90.4_{\pm 2.0}$	$89.4_{\pm 0.3}$	$83.6_{\pm 1.7}$	$89.0_{\pm 1.2}$	$\mathbf{91.9_{\pm 1.0}}$
		40%	$87.4_{\pm 3.3}$	$90.5_{\pm 1.6}$	$95.8_{\pm 0.1}$	$\mathbf{98.0_{\pm 0.4}}$	$89.8_{\pm 1.0}$	$95.4_{\pm 0.6}$	$97.3_{\pm 0.4}$
	T-Finance	1%	90.2	90.5	90.7	91.1	82.9	90.0	$\mathbf{92.8_{\pm 1.1}}$
		40%	91.4	92.1	91.2	94.3	87.1	88.2	$\mathbf{95.0_{\pm 3.0}}$
	T-Social	1%	65.2	71.2	59.8	88.0	56.3	81.4	$\mathbf{91.9_{\pm 0.6}}$
		40%	71.2	71.8	68.4	**95.2**	56.9	82.5	$93.9_{\pm 0.2}$

the network dynamics, feeding signal to masked patches from the visible patches, whereas the HN is crucial later in the dynamics as the model approaches the final reconstruction, sharpening the masked patches. All the qualitative findings presented in this section are in accord with the core computational strategy of the ET block as it was designed theoretically in § 3.2.

3.4 Graph Anomaly Detection

Having gained empirical insights in the image domain, we turn to quantitatively evaluating ET's performance on the graph anomaly detection problem[2], a task with plenty of strong and recently published baselines. Graph Convolutional Networks (GCN) [81] have been widely used for this task due to their capability of learning high level representations of graph structures and node attributes [82, 83]. However, vanilla GCNs suffer from the over-smoothing problem [84]. In each layer of the forward pass, the outlier node aggregates information from its neighbors. This averaging makes the features of anomalies less distinguishable from the features of benign or normal nodes. Our approach does not suffer from this problem, since the routing of the information between the nodes is done through the energy-based attention, and the information aggregation is based on the attention scores.

For anomaly detection on graphs in the ET framework, consider an undirected graph with N nodes. Every node has a vector of raw attributes $\mathbf{y}_A \in \mathbb{R}^F$, where F is the number of node's features. Every node also has a binary label l_A indicating whether the node is benign or anomalous. We focus on node anomaly and assume that all edges are trusted. The task is to predict the label of a node given the graph structure and the node's features. Since there are far more benign nodes in the graph than anomalous ones, anomaly detection can be regarded as an imbalanced node classification task.

First, similarly to images, the feature vectors for every node are converted to a token representation using a linear embedding $\mathbf{E}$ and adding a learnable positional embedding λ_A

$$\mathbf{x}_A^{t=1} = \mathbf{E}\, \mathbf{y}_A + \lambda_A \tag{3.10}$$

where the superscript $t = 1$ indicates the time of the update of the ET dynamics. This token representation is iterated through the ET block for T iterations. When the retrieval dynamics becomes stable, we have the final representation for each node $\mathbf{x}_A^{t=T}$ (or more precisely $\mathbf{g}_A^{t=T}$, since the outputs are additionally passed through a layer norm operation after the final ET update). This output is concatenated with the initial (layer normalized) token to form the final output

$$\mathbf{g}_A^{\text{final}} = \mathbf{g}_A^{t=1} \,||\, \mathbf{g}_A^{t=T} \tag{3.11}$$

where $||$ denotes concatenation. Following [80], the node representation $\mathbf{g}_A^{\text{final}}$ is fed into

[2]The code is available: https://github.com/zhuergou/Energy-Transformer-for-Graph-Anomaly-Detection/.

an MLP with a sigmoid activation function to compute the anomaly probabilities p_A. The weighted cross entropy

$$\text{Loss} = \sum_A \Big[\omega\ l_A \log(p_A) + (1 - l_A) \log(1 - p_A)\Big] \tag{3.12}$$

is used to train the network. Above, ω is the ratio of the benign labels ($l_A = 0$) to anomalous ones ($l_A = 1$). Attention is restricted to 1-hop neighbors of the target node.

3.4.1 Experimental Evaluation

Four datasets are used for the experiments. YelpChi dataset [85] aims at opinion spam detection in Yelp reviews. Amazon dataset is used to detect anomalous users under the Musical Instrument Category on *amazon.com* [86]. T-Finance and T-Social datasets [80] are used for anomalous account detection in transactions and social networks, respectively. For these four datasets, the graph is treated as a homogeneous graph (i.e., all the edges are of the same type), and a feature vector is associated with each node. The task is to predict the label (anomaly status) of the nodes. For each dataset, either 1% or 40% of the nodes are used for training, and the remaining 99% or 60% are split $1:2$ into validation and testing sets; see Appendix A.2 for details.

We compare with state-of-the-art approaches for graph anomaly detection, which include GraphConsis [87], CAREGNN [88], PC-GNN [89] and BWGNN [80]. Additionally, multi-layer perceptrons (MLP) and Graph Transformer (GT) [90] are included in the baselines for completeness. Following previous work, macro-F1 score (unweighted mean of F1 score) and the Area Under the Curve (AUC) are used as the evaluation metrics on the test datasets [91]. See Appendix A.2 for more details on training protocols and the hyperparameters choices. The results are reported in Table 3.1. Our ET network demonstrates very strong results across all the datasets.

3.5 Graph Classification with ET

To fully explore the efficacy of ET, we further evaluate its performance on the graph classification problem[3]. Unlike the anomaly detection task, where we predict a label for every node, in the graph classification task, a single label is predicted for the entire graph.

[3]The code is available: https://github.com/Lemon-cmd/Energy-Transformer-For-Graph.

Table 3.2: Graph classification performance on eight datasets of TUDataset. Following [92], mean and standard deviation obtained from 100 runs of 10-fold cross validation are reported. For baselines standard deviations are only included if they are available in prior work. If the entry is unavailable in prior literature it is denoted by '-'; best results are in **bold**. The performance difference between non-baseline approaches (including ours) and the baselines (specified by their gray cell) is indicated by ▼(decrease), ▲(increase), and ▼(no change within the error bars) along with the value.

	Dataset							
Method	PROTEINS	NCI1	NCI109	DD	ENZYMES	MUTAG	MUTAGENICITY	FRANKENSTEIN
WKPI (kmeans)	$78.5_{\pm 0.4}$ ▼(6.4)	$87.5_{\pm 0.5}$	$85.9_{\pm 0.4}$ ▼(1.5)	$82.0_{\pm 0.5}$ ▼(13.7)	-	$85.8_{\pm 2.5}$ ▼(14.2)	-	-
WKPI (kcenters)	$75.2_{\pm 0.4}$ ▼(9.7)	$84.5_{\pm 0.5}$ ▼(3.0)	$87.4_{\pm 0.3}$	$80.3_{\pm 0.4}$ ▼(15.4)	-	$88.3_{\pm 2.6}$ ▼(11.7)	-	-
Spec-GN	-	$84.8_{\pm 1.6}$ ▼(2.7)	$83.6_{\pm 0.8}$ ▼(3.8)	-	$72.5_{\pm 5.8}$ ▼(5.9)	-	-	-
Norm-GN	-	$84.9_{\pm 1.7}$ ▼(2.6)	$83.5_{\pm 1.3}$ ▼(3.9)	-	$73.3_{\pm 8.0}$ ▼(5.1)	-	-	-
GWL-WL	$75.8_{\pm 0.6}$ ▼(9.1)	-	-	-	$71.3_{\pm 1.1}$ ▼(7.1)	-	-	$78.9_{\pm 0.3}$
HGP-SL	$84.9_{\pm 1.6}$	$78.5_{\pm 0.8}$ ▼(9.1)	$80.7_{\pm 1.2}$ ▼(6.7)	$81.0_{\pm 1.3}$ ▼(14.7)	$68.8_{\pm 2.1}$ ▼(9.6)	-	$82.2_{\pm 0.6}$	-
DSGCN	$77.3_{\pm 0.4}$ ▼(7.6)	-	-	-	$78.4_{\pm 0.6}$	-	-	-
U2GNN	$80.0_{\pm 3.2}$ ▼(4.9)	-	-	$95.7_{\pm 1.9}$	-	$88.5_{\pm 7.1}$ ▼(11.5)	-	-
NDP	$73.4_{\pm 3.1}$ ▼(11.5)	$74.2_{\pm 1.7}$ ▼(13.3)	-	$72.8_{\pm 5.4}$ ▼(22.9)	$44.5_{\pm 7.4}$ ▼(34.9)	$87.9_{\pm 5.7}$ ▼(12.1)	$77.9_{\pm 1.4}$ ▼(4.3)	-
ASAP	$74.2_{\pm 0.8}$ ▼(10.7)	$71.5_{\pm 0.4}$ ▼(16.0)	$70.1_{\pm 0.6}$ ▼(17.3)	$76.9_{\pm 0.7}$ ▼(18.8)	-	-	-	$66.3_{\pm 0.5}$ ▼(12.6)
EvoG	-	-	-	-	55.7 ▼(22.7)	**100.0**	-	-
ET (Ours)	$\mathbf{90.3}_{\pm 0.7}$ ▲(5.4)	$\mathbf{90.1}_{\pm 0.1}$ ▲(2.6)	$\mathbf{90.5}_{\pm 0.1}$ ▲(3.1)	$\mathbf{95.9}_{\pm 0.8}$ ▲(0.2)	**99.8** ▲(21.4)	$96.6_{\pm 0.2}$ ▼(3.4)	$\mathbf{98.7}_{\pm 0.1}$ ▲(16.5)	$\mathbf{99.8}_{\pm 0.1}$ ▲(20.9)

Consider a graph G, where each node has a raw feature vector $\mathbf{y}_A \in \mathbb{R}^F$ with F feature-dimension. Each vector is projected to the token space yielding $\mathbf{x}_A \in \mathbb{R}^D$, where D is the token dimension. A learnable CLS token $\mathbf{x}_{\text{CLS}}$ is concatenated to the set of tokens resulting in the token matrix $\mathbf{X} \in \mathbb{R}^{(N+1)\times D}$, and a learnable linear projection of the top k smallest eigen-vectors of the normalized Laplacian matrix, following [93], is added to $\mathbf{X}$. Graph structural information is provided to ET within the attention operation. We also use a stacked version of the Energy Transformer consisting of S ET blocks, where each block has the same number of temporal unfolding steps T and its own LayerNorm. The final representation of the CLS token $\mathbf{x}_{\text{CLS}}^{\ell=S,\, t=T}$ is projected via a linear embedding into the predictor space. A softmax over the number of classes is applied to this predictor representation, and the cross-entropy loss is used to train the network. See Appendix A.3 for full details.

3.5.1 Experimental Evaluation

Eight graph datasets from the TUDataset [92] collection are used for experimentation. NCI1, NCI109, MUTAG, MUTAGENICITY, and FRANKENSTEIN are a common class of graph datasets consisting of small molecules with class labels representing toxicity or biological activity determined in drug discovery projects [92]. Meanwhile, DD, ENZYMES, and PROTEINS represent macromolecules. The task for both DD and PROTEINS is to classify whether a protein is an enzyme. Lastly, for ENZYMES, the task is to assign enzymes to one of the six classes, which reflect the catalyzed chemical reaction [92]. See Table A.4 for more details of the datasets.

We compare ET with the current state-of-the-art approaches for the mentioned datasets, which include WKPI-kmeans [94], WKPI-kcenters [94], DSGCN [95], HGP-SL [96], U2GNN [97], and EvoG [98]. Additionally, approaches [95, 99, 100, 101, 102], which are close to the baselines, are included to further contrast the performance of our model. Following the 10-fold cross validation process delineated in [92], accuracy score is used as the evaluation metric and reported in Table 3.2. In general, we have observed that the modified ET demonstrates strong performance across the eight datasets. With the exception of MUTAG, ET beats other methods on all the baselines by a substantial margin.

3.6 Discussion and Conclusions

A lot of recent research has been dedicated to understanding the striking analogy between Hopfield Networks and the attention mechanism in transformers. At a high level, the main message of our work is that a transformer-like block (including feed-forward MLP, layer normalization, and residual connections) can be designed as a single large energy-based associative memory, not just attention alone. At a deeper level, we use recent advances in the field of Hopfield Networks to design a novel energy function that is tailored for dynamical information routing between the tokens; and representation of a large number of relationships between those tokens. We have tested the ET network qualitatively on the image completion task, and quantitatively on node anomaly detection and graphs classification. The qualitative investigation reveals the perfect alignment between the theoretical design principles of our network and its empirical computation. The quantitative evaluation demonstrates strong results, which stand in line or exceed the methods recently developed specifically for these tasks. We believe that the proposed network will be useful for other tasks and domains (e.g.,

NLP, audio, and video), which serve as future directions for a comprehensive investigation.

There are two metrics describing computational costs of our architecture: memory footprint of the model and the number of flops. In terms of the memory footprint our model wins (is smaller) compared to feedforward transformers with independent weights and even ALBERT-style shared-weight transformers (see Appendix A.5 and Table A.8). Regarding the number of flops, our model has the same parametric scaling as the conventional transformers (quadratic in the number of tokens), but the energy attention has a constant factor of two more flops (due to the second term in the attention-induced updates of the tokens).

CHAPTER 4
NRGPT: AN ENERGY-BASED ALTERNATIVE FOR GPT

Generative Pre-trained Transformer (GPT) architectures are the most popular design for language modeling. Energy-based modeling is a different paradigm that views inference as a dynamical process operating on an energy landscape. We propose a minimal modification of the GPT setting to unify it with the EBM framework established by the ENERGY TRANSFORMER. The inference step of our model, which we call eNeRgy-GPT (NRGPT), is conceptualized as an exploration of the tokens on the energy landscape. We prove, and verify empirically, that under certain circumstances this exploration becomes gradient descent, although these conditions do not necessarily lead to the best performing models. We demonstrate that our model performs well for simple language (Shakespeare dataset), algebraic ListOPS tasks, and richer settings such as OpenWebText language modeling. We also observe that our models may be more resistant to overfitting, doing so only during very long training.

4.1 Introduction

Modern transformer architectures represent a dominant paradigm in autoregressive language modeling [3]. In a typical setting, a sequence of tokens describing a text is passed through several transformer layers and mapped onto a new sequence, which is a copy of the original one shifted by one token and appended by the token that follows the initial sequence. At training time, this network is trained through self-supervised training, and at inference time the network is used for next token prediction. This is the standard Generative Pre-trained Transformer (GPT) setting, which is the first step in Large Language Model (LLM) design [103].

Energy-based modeling [36] is another prominent paradigm in the modern AI landscape that historically goes back to Hopfield Networks [9]. In this framework the operation of the neural network is defined by a scalar energy function. Proper samples generated by the model (those that resemble training data) correspond to low energy states, while unrealistic samples (with large deviations from the training data distribution) correspond to high energy states. Thus EBMs are particularly appealing for their theoretical properties, because they

view the "forward pass" through a deep network as an explicit optimization problem that maximizes the likelihood of the input data. Practically, this approach unlocks better tooling for systematic exploration of the solution space and enables natural solutions for both variable computation and model alignment via regularizers (see Appendix B.2).

Although at face value these two approaches look very different, in recent years a growing number of studies hint at deep connections. Von Oswald et al. [104] showed evidence that in-context learning (ICL) may be gradient descent by constructing explicit weights such that the forward pass was GD on MSE loss. Ahn et al. [105] further showed that transformers learn a preconditioned GD for ICL. However, both of these works make significant simplifications, such as considering only *linear* transformers, omitting the softmax.

Other works have attempted to reconcile transformers and EBM from several angles. For instance, the ENERGY TRANSFORMER [11], discussed in Chapter 3, is an architecture, which is simultaneously a transformer and an energy-based model. In the image domain, the typical setting would be to reconstruct a set of masked tokens (patches) given the set of open tokens. The network solves this task by performing a gradient descent of the energy on the space of tokens at inference time. This architecture is inspired by energy-based AM models [26] and for this reason solves the following problem: given a partially incomplete pattern, complete it in a meaningful way. This aspect of the core design makes it difficult to apply Energy Transformers to GPT settings, in which the sequence needs to be transformed to a shifted sequence by means of going through the network. Intuitively, in Energy Transformers the masked tokens need to evolve rapidly to match the missing parts of the pattern (e.g., image or graph), while the open tokens need to stay almost constant to barely adjust for the smooth transitions between the masked and the open tokens within the pattern. This is in drastic contrast with the GPT setting, in which there are no masked tokens at all. Rather, every token needs to evolve into the following token in the sequence.

A different line of work is inspired by "System 2" thinking and attempts to design an energy-based network for processing language [106]. In this study, transformers are used as an architectural motif that casts text into a scalar energy function. While models of this nature have benefits for language processing, they belong exclusively to the class of energy-based models, and are unrelated to the GPT settings, commonly used in most LLMs.

Individual modules within the transformer block, such as attention, have also been studied from the perspective of inference time optimization [107, 108]. In this line of work, peculiar clustering properties of tokens have been observed. Energy-based optimization has

also been studied in [72] from the perspective of majorization-minimization algorithms.

Despite this growing list of studies dedicated to synergies between autoregressive transformers and energy-based models, at present it remains unknown how to cast the commonly used GPT setting into a well-defined energy-based framework. Our work tackles this gap. We refer to our model as e**N**e**R**gy **G**enerative **P**re-trained **T**ransformer or NRGPT. The input sequence of tokens is mapped onto a shifted sequence of tokens, which includes the next word, see Figure 4.1. The mapping is performed by a neural network, which recurrently applies the NRGPT block to the sequence of tokens. Each application of the block uses gradients of the network energy functions to update the state of the tokens. Each token has its own energy landscape, which is dependent on the states of other tokens. Specifically, our contributions are:

- We design an energy function and an update rule that describes the GPT setting with several possible variants including learnable inference rate and normalization operations: LayerNorm and RMSNorm.

- We obtain excellent results on nested ListOPS tasks, including arithmetic operations, min/max selection, etc.

- We show the feasibility of using NRGPT for language modeling on Shakespeare and OpenWebText datasets.

- We do a systematic comparison of performance scaling of recurrent transformers and NRGPT.

- We study empirically the properties of dynamical trajectories of tokens on the energy landscapes of our models.

4.2 Energy-based modeling

In generative modeling our goal is to generate samples with a distribution close to observed datapoints. If we manage to learn an approximate likelihood function for the dataset, we can generate data by sampling. This is also the premise of Energy-Based Models (EBM). An example of EBM would be Dense Associative Memory [26], where datapoints are stored in minima of an energy function. But more generally, the energy can represent a

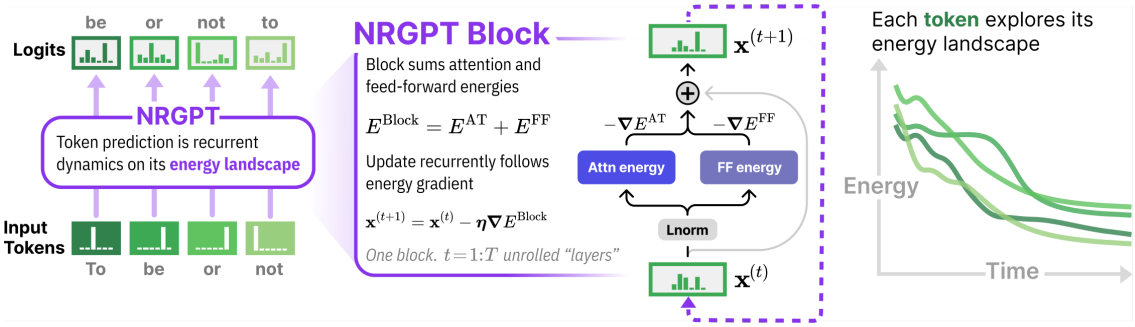

Figure 4.1: **NRGPT** casts the standard GPT setting into an energy-based framework. The network is defined as the sum of two energies: an **attention** energy and a **feedforward** energy. Each **token** is transformed into the next token by exploring the energy landscape. Recurrent application of the NRGPT block produces a dynamical system where each token can be thought of as a particle moving on the network's energy landscape.

negative-log-likelihood, $E(\mathbf{x}) = -\log P(\mathbf{x})$. In this case, the global minima of the energy represent maximum likelihood solutions. The deeper the energy, the higher the likelihood of that datapoint. One strategy to train an EBM is to first learn the energy function by fitting the distribution of the data. The sampling process would then be separate from learning the energy.

However, in high dimensional data, learning the distributions is notoriously difficult due to the curse of dimensionality. Diffusion models solve this problem by starting from high noise and cooling down. Diffusion models do not learn an explicit energy function, only its gradients, the score function. Yet, having an explicit energy function could enable us to explore the solution space in ways not easily afforded by the implicit score function of diffusion models. So can we build a model which learns the energy directly?

Similar to diffusion models, we use an end-to-end process, where learning the energy and generating datapoints are both done in one pass. The key idea is to have a differentiable sampling process which allows us to learn the parameters of the energy during sampling. Since real datapoints should have low energies, we choose a gradient-based sampling process. Note that we do not need to descend all the way to a minimum (i.e. maximum likelihood solution), since we want diverse samples. Instead, we do a fixed number of energy GD steps and demand that the final point matches real datapoints.

$$\text{Generated data: } \mathbf{x}^{(T)}, \qquad \mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \boldsymbol{\eta}^{(t)} \nabla E(x^{(t)}), \tag{4.1}$$

for a fixed number of steps T, where $\mathbf{x}^{(0)}$ is some random initial point. Here $\boldsymbol{\eta}^{(t)}$ is a matrix

that may depend on $\mathbf{x}$. This matrix has many different names, e.g., kinetic rates in physics, preconditioner in optimization, etc. We will call this matrix the *inference rate*, since it determines the size of the steps that the inference dynamics takes on the energy landscape. But how do we judge whether the output matches a real datapoint? One way would be to have a judge, like the discriminator in a GAN. Another setting where judging the output is more natural is autoregressive language modeling where the new datapoints are the next tokens and can be matched to the training text. In this case $\mathbf{x} \in \mathbb{R}^{N \times D}$ represents a real data sequence of length N embedded in D dimensions. In causal language modeling the energy should take $\mathbf{x}_{<N} = (\mathbf{x}_1 \dots \mathbf{x}_{N-1})$ as input and predict $\mathbf{x}_N$, as in

$$\mathbf{x}_N^{(t+1)} = \mathbf{x}_N^{(t)} - \boldsymbol{\eta} \boldsymbol{\nabla} E(\mathbf{x}_N^{(t)} | \mathbf{x}_{<N}^{(t)}). \tag{4.2}$$

Following the observations of the Energy Transformer (ET), we will show that one can choose a parametrization for E such that the process of T-step GD closely resembles the forward-pass through a T layer GPT transformer with a weight-sharing pattern.

4.3 NRGPT Module

$$E_A = -\frac{1}{\beta} \sum_{h=1}^{H} \log \left(\sum_{B<A} \exp \left(\beta \, \mathbf{g}_B^T \boldsymbol{J}^h \mathbf{g}_A \right) \right) - \mathbf{g}_A^T \mathbf{W_2} \sigma \Big(\mathbf{W_1} \mathbf{g}_A \Big). \tag{4.3}$$

In this section we will start from the structure of the transformer model and derive the energy function whose gradients yield a layer which is very close in structure to a transformer layer. Let $\mathbf{x} \in \mathbb{R}^{D \times N}$ be an input sequence of length N embedded in D dimensions. We will denote its components by $\mathbf{x}_{Ai}$ with $A = 1 \dots N$ and $i = 1 \dots D$, or $\mathbf{x}_A$ suppressing the embedding index, but keeping the token index. Let $\mathbf{x}^{(t)}$ be the output sequence of layer t of the model with $\mathbf{x}^{(0)} = \mathbf{x}$. A conventional transformer layer has an Attention layer (AT) followed by a two-layer feedforward (FF) and LayerNorm (LN) in series

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \text{FF}\Big[\text{LN}\Big(\mathbf{x}^{(t)} + \text{AT}\big(\text{LN}(\mathbf{x}^{(t)})\big)\Big)\Big]. \tag{4.4}$$

But subsequent works such as GPT-J [109], PaLM [110], and Energy Transformer [11] showed that the following parallel design has good performance too

$$\textbf{Parallel transformer:} \quad \mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \text{AT}(\mathbf{g}^{(t)}) + \text{FF}(\mathbf{g}^{(t)}), \quad \mathbf{g}^{(t)} = \text{LN}(\mathbf{x}^{(t)}) \tag{4.5}$$

We choose this parallel transformer design as it is more suitable for our goal of replacing the transformer layer with the gradient of an energy.

If passing through a layer becomes one step of energy descent (ED), then all layers need to share weights. Therefore, our model will consist of a single module replacing the transformer block. Instead of different layers, we will be recurrently feeding the output of the layer back into itself, so that $\mathbf{x}^{(t)}$ will become step t of the ED instead of the layer number.

Update Rule An important point to note is that the update rule of NRGPT is slightly different from conventional gradient descent and is of the form

$$\dot{\mathbf{x}} = \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)} = -\boldsymbol{\eta}^{(t)} \frac{\partial E}{\partial \mathbf{g}^{(t)}}, \tag{4.6}$$

where $\boldsymbol{\eta}^{(t)} \in \mathbb{R}^{D\times D}$ is an inference rate matrix, which can be learnable. Nevertheless, this can be a valid descent on E as we can show that $E^{(t+1)} - E^{(t)} < 0$ under certain conditions, which depend on the normalization operation $\mathbf{g}$. We will derive these conditions for LayerNorm as well as RMSNorm, as well as when $\mathbf{g} = \mathbf{x}$, i.e., no normalization in Subsection 4.3.2. Next, we introduce the energy of NRGPT module.

4.3.1 Energy of NRGPT

Matching our update rule in Equation (4.6) to the parallel transformer in Equation (4.5), we define two terms in the energy, E^{AT} and E^{FF}

$$E = E^{\mathrm{AT}} + E^{\mathrm{FF}}, \qquad \boldsymbol{\eta} \mathrm{o}_g E^{\mathrm{AT}} = -\mathrm{AT}(\boldsymbol{g}), \qquad \boldsymbol{\eta} \mathrm{o}_g E^{\mathrm{FF}} = -\mathrm{FF}(\boldsymbol{g}). \tag{4.7}$$

We begin by introducing the attention layer and deriving the energy function for the self-attention mechanism. Then, we derive the energy function for the FF. Finally, we combine the two energy functions to obtain the total energy function for the transformer layer.

Attention. Consider a multi-head attention module with H heads, and hidden dimension $Y = D/H$, index h enumerates heads and runs $h = 1 \dots H$. Its query, key, value and projection weights are

$$\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V, \boldsymbol{W}^P \in \mathbb{R}^{H\times Y\times D}. \tag{4.8}$$

Using the standard definitions $\boldsymbol{K} = \boldsymbol{W}^K \mathrm{g}, \boldsymbol{Q} = \boldsymbol{W}^Q \mathrm{g}, \boldsymbol{V} = \boldsymbol{W}^V \mathrm{g}$, the MHA output for token A is[1]

$$\mathrm{AT}(\mathrm{g})_A = \sum_{h=1}^{H} \left[\boldsymbol{W}_h^P\right]^T \boldsymbol{V}_h \mathrm{SM}\left(\boldsymbol{K}_h^T \boldsymbol{Q}_{Ah}\right), \tag{4.9}$$

denoting $\boldsymbol{J} = \left[\boldsymbol{W}^K\right]^T \boldsymbol{W}^Q$, the softmax is defined as (we omit the self-interaction term $C = A$)

$$\mathrm{SM}(\boldsymbol{K}^T \boldsymbol{Q})_{BA} = \frac{\exp\left(\beta \boldsymbol{g}_B^T \boldsymbol{J} \boldsymbol{g}_A\right)}{\sum_{C<A} \exp\left(\beta \boldsymbol{g}_C^T \boldsymbol{J} \boldsymbol{g}_A\right)}, \quad \beta = \frac{1}{\sqrt{Y}}. \tag{4.10}$$

Following [11], define the attention energy

$$E_A^{\mathrm{AT}}(\boldsymbol{g}) = -\frac{1}{\beta} \sum_h \alpha_h \log\Big[\sum_{B<A} \exp\left(\beta \boldsymbol{g}_B^T \boldsymbol{J}_h \boldsymbol{g}_A\right)\Big], \tag{4.11}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^H$ is a learnable weight.

Taking the gradient of E^{AT} w.r.t. $\boldsymbol{g}_A$ and using Equation (4.7), the resulting attention layer becomes

$$\mathrm{AT}(\boldsymbol{g})_A = -\boldsymbol{\eta} \frac{\partial E_A^{\mathrm{AT}}(\boldsymbol{g})}{\partial \boldsymbol{g}_A} = \sum_{h=1}^{H} \alpha_h \boldsymbol{\eta} \boldsymbol{J}_h^T \boldsymbol{g} \mathrm{SM}\left(\boldsymbol{g}^T \boldsymbol{J}_h \boldsymbol{g}_A\right). \tag{4.12}$$

Both standard attention and this update have the form $\mathrm{AT}(\mathrm{g}) = \boldsymbol{M}\mathrm{g}\mathrm{SM}(\mathrm{g}^T \boldsymbol{J} \mathrm{g})$, but in the standard attention $\boldsymbol{M} = [\boldsymbol{W}^P]^T \boldsymbol{W}^V$ and in NRGPT attention is $\boldsymbol{M} = \alpha \boldsymbol{\eta} \boldsymbol{J}^T$. That is,

$$\text{Original: } \left[\boldsymbol{W}_h^P\right]^T \boldsymbol{W}_h^V \equiv \text{Energy: } \alpha_h \boldsymbol{\eta} \boldsymbol{J}_h^T. \tag{4.13}$$

In principle $\boldsymbol{W}^V$ and $\boldsymbol{W}^P$ can be merged into one matrix. [111] also experimented with removing $\boldsymbol{W}^V$ and $\boldsymbol{W}^P$ and found that in the setting without skip connections, these two weights could be largely omitted.

Feed-Forward network. The FF network generally has two layers $\mathrm{FF}(\boldsymbol{g}_A) = \boldsymbol{W}^{2^T} \sigma\left(\boldsymbol{W}^1 \boldsymbol{g}_A\right)$ with weights $\boldsymbol{W}^1, \boldsymbol{W}^2 \in \mathbb{R}^{M \times D}$, with M being the size of the hidden layer. A possible

[1] Usually the projection weights $\boldsymbol{W}^P$ are defined as $D \times D$ and the head outputs are concatenated, into an $N \times (YH) = N \times D$ matrix before multiplying by W^P. This is equivalent to our definition.

choice for this network is a Dense Associative Memory [26]. In this case

$$E^{\text{FF}} = -\sum_{A=1}^{N} \mathbf{1}^T F\left(\boldsymbol{W}^1 \boldsymbol{g}_A\right), \quad \text{s.t. } F' = \sigma$$
$$\text{FF}(\boldsymbol{g}_A) = -\eta \frac{\partial E^{\text{FF}}}{\partial \boldsymbol{g}_A} = \boldsymbol{\eta} \boldsymbol{W}^{1^T} \sigma\left(\boldsymbol{W}^1 \boldsymbol{g}_A\right), \tag{4.14}$$

where $\mathbf{1}$ is an M-dimensional vector of ones. Both the standard MLP in transformers and the FF update in Equation (4.14) have the form $\boldsymbol{M}\sigma(\boldsymbol{W}^1 g)$. In the standard MLP, $\boldsymbol{M} = \boldsymbol{W}^{2^T}$, whereas in NRGPT we get $\boldsymbol{M} = \boldsymbol{W}^1 \boldsymbol{\eta}^\top$. That is,

$$\text{Original: } \boldsymbol{W}^{2^T} \equiv \text{Energy: } \boldsymbol{W}^1 \boldsymbol{\eta}^T. \tag{4.15}$$

As an example, in order for E^{FF} to reproduce the FF of transformers with $\sigma(z) = \text{ReLU}(z) = \max(z, 0)$, the function F should be

$$F(z) = \frac{1}{2}\sigma(z)^2. \tag{4.16}$$

Of course, the FF module can be replaced by other, more general, MLP networks. Essentially, any scalar function, which is additive in token index, can serve as a valid form of FF network. In the experiments (§ 4.4) we will detail our choices of E^{FF}.

Distinction from prior work NRGPT's energy is distinct from prior works like that of Energy Based Transformers (EBT) [106] and Energy Transformer (ET) [11]. Specifically, EBT computes energies of next tokens an output of a standard transformer forward pass, while NRGPT describes a parameterized energy whose *gradient* returns a transformer block. Repeatedly minimizing the energy by following this gradient resembles a complete transformer forward pass. On the other hand, ET's architecture appears more similar to NRGPT's design, but its design was unsuited for *causal* token generation. We expand on these differences further in Appendix B.1.

4.3.2 Normalization of tokens

These normalizations have the form

$$\boldsymbol{g} = \boldsymbol{\gamma} \odot \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\sqrt{\frac{1}{D}\|\boldsymbol{x} - \boldsymbol{\mu}\|^2 + \epsilon}} + \boldsymbol{\delta}, \tag{4.17}$$

with $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}]$ for LayerNorm, and $\boldsymbol{\mu} = 0, \boldsymbol{\delta} = 0$ for RMSNorm. Here $\boldsymbol{\gamma}, \boldsymbol{\delta} \in \mathbb{R}^D$ and $\odot$ is elementwise multiplication. Many recent models such as Qwen [112] and Llama [113] use RMSNorm [114].

Proposition 4.1 (Energy Descent)**.** *The update rule in Equation (4.6) results in asymptotically decreasing energy, $\dot{E}_A = E_A^{(t+1)} - E_A^{(t)} < 0$, if the inference rate is $\boldsymbol{\eta} = c\,\mathrm{diag}(\gamma)$ with $c \in \mathbb{R}_{>0}$. This asymptotic behavior begins after a transient phase where previous tokens are converging.*

Sketch of proof. See Appendix B.4 for the full proof. The Jacobian of $\boldsymbol{g}_A$ can be written as $\partial \boldsymbol{g}_A / \partial \boldsymbol{x}_A = \frac{1}{r_A}\boldsymbol{\Gamma}\boldsymbol{P}_A$, where $\boldsymbol{\Gamma} = \mathrm{diag}(\gamma)$, $r_A > 0$ is some norm of $\boldsymbol{x}_A$ and $\boldsymbol{P}_A$ is an approximate $D \times D$ projection matrix and p.s.d. Using this and Equation (4.6), once $\dot{\boldsymbol{x}}_B = 0$ for $B < A$, we get $\dot{E}_A = -r_A^{-1}\,\mathrm{Tr}\,\partial_{\boldsymbol{g}_A} E_A \boldsymbol{\Gamma}\boldsymbol{P}_A\boldsymbol{\eta}^T \left(\partial_{\boldsymbol{g}_A} E_A\right)^T$. When $\boldsymbol{\eta} = \boldsymbol{\Gamma}$ we get $\dot{E} < 0$. □

There also exist x-dependent solutions of the form $\boldsymbol{\eta} = \boldsymbol{M}(\boldsymbol{x})\boldsymbol{P}_A\boldsymbol{\Gamma}$, where $\boldsymbol{M}(\boldsymbol{x})$ is an arbitrary positive semi-definite (psd) matrix, but they mean $\boldsymbol{\eta}$ is itself a neural network. To remain close to the structure of conventional transformers, we work with the x-independent $\boldsymbol{\eta} = c\boldsymbol{\Gamma}$ solution with $c > 0$. In principle, $\boldsymbol{\eta}$ can also have a part contributing to the anti-symmetric part of $\boldsymbol{\Gamma}\boldsymbol{P}_A\boldsymbol{\eta}^T$, but we could not find an x-independent solution for it. While $\boldsymbol{\eta}^{(t)} = c^{(t)}\boldsymbol{\Gamma}$ puts severe restrictions on the preconditioning matrix, each layer is still allowed to have a different $c^{(t)}$ constant.

Yet, as we show in Appendix B.4, LayerNorm together with Lipschitz activation functions in FF (e.g. ReLU) will lead to a bounded energy, which cannot explode. In this case, a generic $\boldsymbol{\eta}$ may still yield convergence. In fact, only a small set of $\boldsymbol{\eta}$ which yield perpetually oscillating solutions may not lead to convergence, although the exact conditions need to be derived.

Preconditioner without layer normalization. Several works have shown careful initialization and scaling can replace normalization entirely [115, 116]. Doing so would remove

much of the restrictions on the preconditioner η. In a T-layer transformer, instead of layer normalization, T-Fixup [115] and DeepNet [116] initialize some weights and x by a scale proportional to $T^{-1/4}$ and change most layer outputs to $f(g) \to \alpha f(x)$, with $\alpha \propto T^{-1/2}$. In this case, η becomes much less restricted.

Proposition 4.2 ($\dot{E}$ without normalization)**.** *When $\boldsymbol{g} = \boldsymbol{x}$, to get $\dot{E} < 0$, the symmetric part, $\boldsymbol{\eta}_+ = (\boldsymbol{\eta} + \boldsymbol{\eta}^T)/2$ needs to be psd.*

Proof. In this case $\dot{\boldsymbol{x}} = -\boldsymbol{\eta} \mathrm{o}_{\boldsymbol{x}} E$ and

$$\dot{E} = -\sum_A \mathrm{Tr}\, \mathrm{o}_{\boldsymbol{x}_A} E \boldsymbol{\eta} \mathrm{o}_{\boldsymbol{x}_A} E^T = -\sum_A \mathrm{Tr}\, \mathrm{o}_{\boldsymbol{x}_A} E \boldsymbol{\eta}_+ \mathrm{o}_{\boldsymbol{x}_A} E^T, \tag{4.18}$$

which is negative when $\boldsymbol{\eta}_+$ is psd. □

The antisymmetric part $\boldsymbol{\eta}_- = (\boldsymbol{\eta} - \boldsymbol{\eta}^T)/2$ is not constrained by this and can be arbitrary at this point. Hence, without layer normalization, we can have $\boldsymbol{\eta}^{(t)}$ as learnable parameters of the form

$$\boldsymbol{\eta} = \boldsymbol{U}^T \boldsymbol{U} + \boldsymbol{V} - \boldsymbol{V}^T, \qquad \boldsymbol{U} \in \mathbb{R}^{D \times D'}, \boldsymbol{V} \in \mathbb{R}^{D \times D}, \tag{4.19}$$

where D' can be arbitrary, and each layer (depth) can have a different learnable $\boldsymbol{U}^{(t)}, \boldsymbol{V}^{(t)}$.

4.3.3 Asymptotic Stability of NRGPT

A peculiar aspect of NRGPT is the phenomenon of asymptotic stability. In order to illustrate it, consider a simplifying case when the inference rate matrix $\boldsymbol{\eta}$ is identity. In this case dynamical equations for tokens can be written as

$$\dot{x}_{iA} = -\frac{\partial E_A}{\partial g_{iA}}. \tag{4.20}$$

Additionally, $\boldsymbol{\gamma}$ can always be absorbed into $\boldsymbol{J}$ and the weights of the FF model. This way, the Jacobian becomes $\partial \boldsymbol{g}_A / \partial \boldsymbol{x}_A = \boldsymbol{P}_A$, which is p.s.d.. The key observation is that due to causal attention mask the energy E_A of token A only depends on the states of tokens $B \leq A$. Thus, for the first token

$$\dot{\boldsymbol{x}}_1 = -\frac{\partial E_1}{\partial \boldsymbol{g}_1}. \tag{4.21}$$

Since the energy of that token decreases with time

$$\dot{E}_1 = \frac{\partial E_1}{\partial \boldsymbol{g}_1}\frac{\partial \boldsymbol{g}_1}{\partial \boldsymbol{x}_1}\frac{d\boldsymbol{x}_1}{dt} = -\dot{x}_1^T \boldsymbol{P}_1 \dot{x}_1 \leq 0, \tag{4.22}$$

since $\boldsymbol{P}_1$ is psd. Additionally, since energy only depends on $\boldsymbol{g}$ – layernormalized tokens – it is bounded from below. Thus, the dynamics of $\boldsymbol{g}$ has to converge to a fixed point. This means that after a transitory period of time T_{tr} the derivative $\frac{d\boldsymbol{g}_1}{dt}$ vanishes.

Now, consider the network of two tokens

$$\dot{E}_2 = \frac{\partial E_2}{\partial \boldsymbol{g}_1}\frac{d\boldsymbol{g}_1}{dt} + \frac{\partial E_2}{\partial \boldsymbol{g}_2}\frac{\partial \boldsymbol{g}_2}{\partial \boldsymbol{x}_2}\frac{d\boldsymbol{x}_2}{dt} = -\dot{x}_2^T \boldsymbol{P}_2 \dot{x}_2 \leq 0, \tag{4.23}$$

the second equality is written assuming that we are looking at this quantity at $t > T_{\text{tr}}$. Thus the first term is zero. Same argument applies, the energy decreases with time and is bounded from below. Thus, eventually $\boldsymbol{g}_2$ freezes.

One can apply this argument recursively to each token and conclude that after a transitory period of time, all tokens stabilize and all $\boldsymbol{g}_A$ will eventually become constants. From the perspective of energy profiles, this leads to the following behavior: during transitory regime energies of individual tokens will evolve in time (they can both increase and decrease). After that transitory period is over the energies must stabilize and reach their constant values that become unchanged in the future. One can run the inference dynamics as long as desired after that, but no changes in energies will occur. This behavior is apparent from the numerical profiles of energies, see Figure 4.2. It is a distinct aspect of our models - the phenomenon that we call asymptotic stability.

4.4 Experiments

We tested our model on three datasets: ListOps, Shakespeare and Open Web Text (OWT). The details of the experimental settings and hyperparameters can be found in Appendix B.5. To evaluate the quality of text in the Shakespeare and OWT experiments, we use a number of metrics, including perplexity and diversity scores, explained in Appendix B.5.2.

Model Variants. The choice of the energy function is equivalent to the choice of architectures in neural networks. As our goal is to be as close to the original transformer architecture as possible, we experimented with a few settings for the FF network and found the following variants to be the best performing:

1. `NRGPT_H_FF1`: $E^{\text{FF}} = -\|\sigma(\boldsymbol{W}\boldsymbol{g}_A)\|^2$, same as in Equation (4.14), but with $\sigma =$ GELU.

2. `NRGPT_H_FF2W`: $E_A^{\text{FF}} = -\boldsymbol{g}_A^T \boldsymbol{W}^{2^T} \sigma(\boldsymbol{W}^1 \boldsymbol{g}_A)$, which yields

$$\text{FF}(\boldsymbol{g}_A) = \eta \left(\boldsymbol{W}^{2^T} \sigma(\boldsymbol{W}^1 \boldsymbol{g}_A) + \boldsymbol{W}^{1^T} \left(\sigma'(\boldsymbol{W}^1 \boldsymbol{g}_A) \odot \boldsymbol{W}^2 \boldsymbol{g}_A \right) \right) \tag{4.24}$$

   Here, the first term is the conventional FF of transformers, but the second is a somewhat odd network.

In case with two weights, we choose the hidden dimension between $\boldsymbol{W}^1$ and $\boldsymbol{W}^2$ to be $4 \times D$. All of these performed well, with the residual version showing best performance on ListOps, learning at even smaller sizes than our baseline, while also easily training at larger sizes with embedding size over 256. However, since some of these models deviate significantly from the FF of a recurrent GPT (the gradient results in an FF module with four layers), we decided to focus more on `NRGPT_H_FF1` and `NRGPT_H_FF2W`, which are much closer to the GPT FF. As baselines, we used `GPT_Rec_parallel`, which is a GPT-J model with a single transformer layer, feeding back recurrently into itself for a fixed number of times (mimicking number of layers). On Shakespeare and OWT, we also show results of a conventional GPT2-style deep transformer model.

4.4.1 Energy dynamics

NRGPT without constraints on the inference rate η is not forced to strictly decrease energy during inference and it may learn other exploration strategies for inference. Nevertheless, we would like to examine whether models which are explicitly forced to perform GD and reduce energy during inference can learn the tasks well. To better understand how our gradient-based update rule performs inference, we ran experiments on ListOps with large number of recurrent steps (30 steps). For these experiments, we set $\eta = 1$, which according to Proposition 4.1 forces the update rule to decrease energy. Figure 4.2 shows the evolution of total E, E^{AT} and E^{FF} along these trajectories. We observe that indeed in all trajectories the final energy is lower than initial. Each individual token trajectory is not required to be monotonically decreasing, as the dynamics of tokens is coupled. However, in accordance with our result in Subsection 4.3.3, after a transient stage, once all previous tokens start converging, the energy of the next token monotonically decreases.

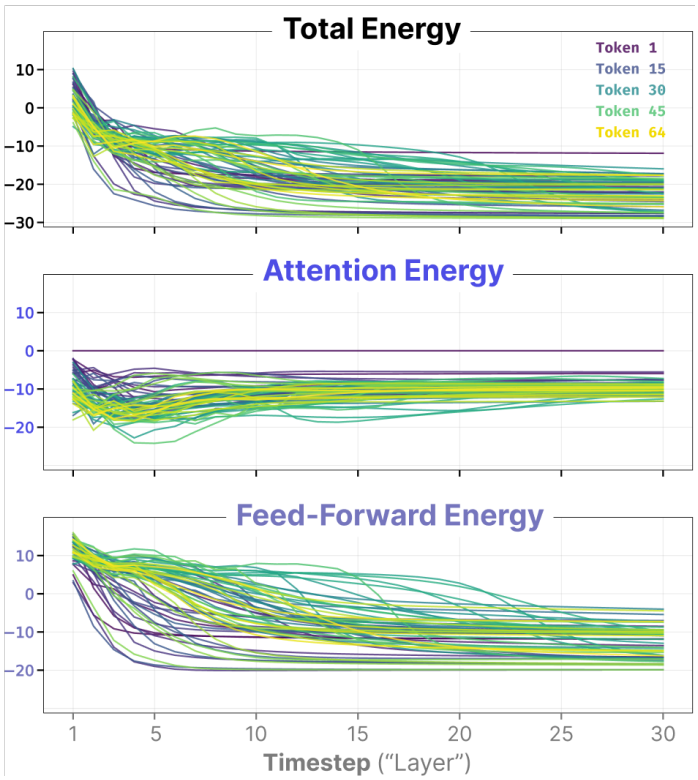

Figure 4.2: In NRGPT, tokens converge to stable states of low energy where the causal attention mask allows each token energy to fluctuate during inference. Shown are 64 tokens passed to an NRGPT model trained to predict ListOps equations.

4.4.2 ListOps

We perform experiments on nested math operations on lists of integers, which are a version of ListOps [117]. Our ListOps setting consists of three functions: maximum, median and sum modulo 20. Our inputs range from 0 to 19. Each data sample begins with nested equations like `SUM(2,MAX(4,13,1),MEDIAN(5,3,16))`. As performance metrics, we looked at accuracy on the mixed task, as well as the training and validation loss. Figure 4.3 shows the results for two of our model variants, `NRGPT_H_FF1` and `NRGPT_H_FF2W` as compared to `GPT_Rec_parallel`.

4.4.3 Shakespeare

We compare the performance of our `NRGPT_H_FF2W` and `NRGPT_H_FF1` with `GPT_Rec_parallel` and deep GPT for embedding sizes less than 1024 (Figure 4.4). In larger sizes, we ran many sweeps to find suitable hyperparameters such as the range of learning rates, resulting in the wide spread. Interestingly, our best models at large sizes achieve validation losses similar to

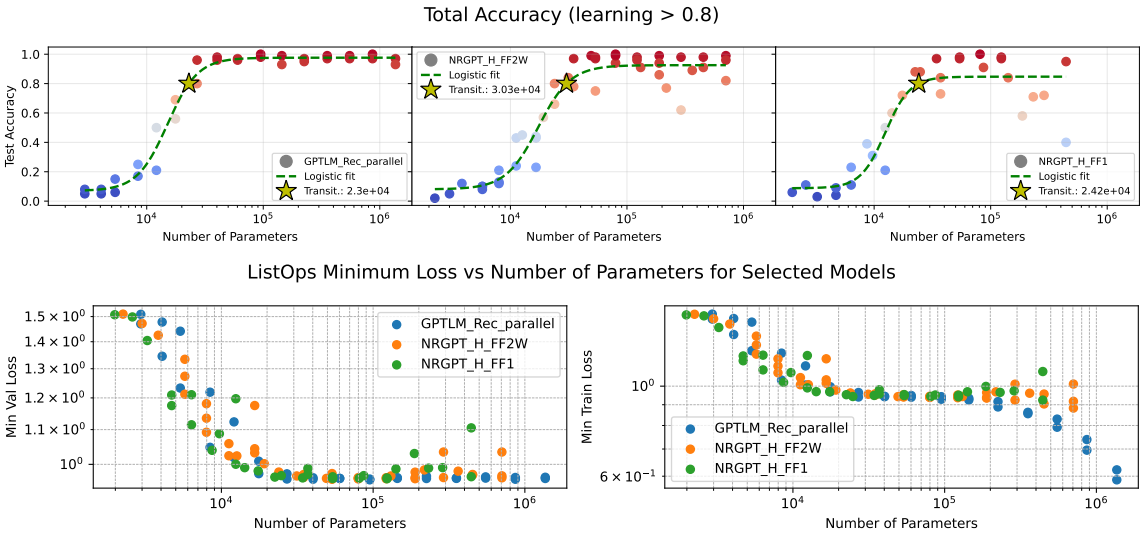

Figure 4.3: **Learning ListOps:** NRGPT variants match performance with a recurrent GPT model on ListOps accuracy parameter-transition points (top) and training/validation losses (bottom). The accuracy of models is tested on nested, mixed arithmetic tasks of maximum, median and sum modulo 20. For all plots, the x axis shows the *total parameter count* of the model. The yellow star indicates the transition to learning, which we define as where the logistic fit hit $> 80\%$ accuracy. The baseline model `GPT_Rec_parallel` shows the earliest learning transition at size 2.3×10^4, but our NRGPT variants are also similar, with `NRGPT_H_FF1` at 2.4×10^4 and `NRGPT_H_FF2W` at 2.98×10^4.

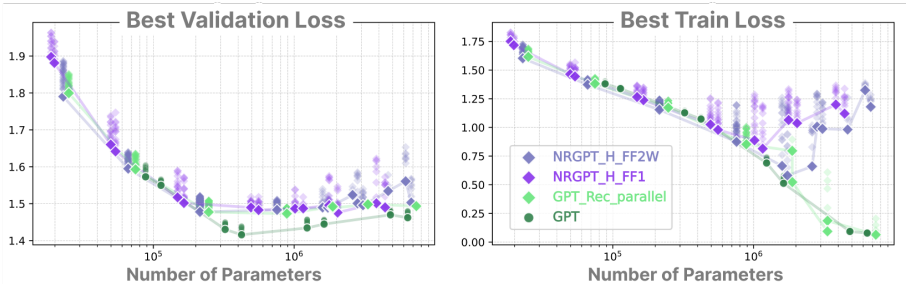

Figure 4.4: **Shakespeare scaling:** NRGPT achieves performance parity with recurrent GPT on Shakespeare across parameter sizes, as measured by *best validation loss* per number of parameters. For many embedding sizes, NRGPT also follows the same optimal training loss trajectory-per-parameter as both GPT and recurrent GPT baselines. However, NRGPT does not overfit Shakespeare at large parameter sizes. Connecting lines show the best performance at fixed parameter sizes. Transparent dots show different choices of hyperparameters — a larger spread indicates more sensitivity to hyperparameters. See Appendix B.5.3 for details.

the GPT baselines, but do so with much less overfitting to the training set.

Table 4.1: OWT performance at $n_{embed} = 768$.

	Training loss				
Model	mean ± std	min	max	# runs	# Param.
GPT	**2.905 ± 0.006**	**2.900**	**2.914**	5	124M
GPT_Rec_parallel	3.447 ± 0.046	3.395	3.494	5	85M
NRGPT_H_FF2W	3.456 ± 0.076	3.391	3.540	3	90M

	Validation loss				
Model	mean ± std	min	max	# runs	# Param.
GPT	**2.921 ± 0.005**	**2.915**	**2.929**	5	124M
GPT_Rec_parallel	3.454 ± 0.037	3.411	3.491	5	85M
NRGPT_H_FF2W	3.467 ± 0.073	3.404	3.548	3	90M

Table 4.2: Best Model Configurations and Quality Metrics for OWT. Note abbreviations: number of parameters → n_param, grammar quality score → gqs, average pairwise cosine similarity → apcs, distinct-1 → d-1 and distinct-2 → d-2.

Model	**Configuration**						**Metrics**				
	lr	min_lr	n_layer	n_head	n_embed	n_params	perplexity	gqs	apcs	d-1	d-2
GPT	7e-4	7e-5	12	12	768	124M	**75**	**0.978**	0.306	0.619	0.965
GPT_Rec_Parallel	6e-4	4e-4	12	12	768	**85M**	99	0.976	**0.336**	0.615	0.975
NRGPT_H_FF2W	1e-4	7e-5	12	12	768	90M	104	0.966	0.306	**0.674**	**0.984**

4.4.4 Open Web Text

Table 4.2 shows the best model configuration for baseline GPT, RGPT-parallel, and our model NRGPT with the respective generation quality metrics. We see that the generation quality of NRGPT is very competitive with GPT and RGPT-parallel while it contains around 34M less parameters than GPT. Figure B.1 shows examples of generated text by GPT, RGPT-parallel, and NRGPT for which the generation quality metrics are provided in Table 4.2. In these experiments, we consider transformer blocks configured at the same *width* — please see Appendix B.5.4 for more experiments across equal parameter counts, including experiments that show NRGPT's advantage on downstream tasks such as MMLU [118].

4.5 Limitations

NRGPT is an appealing theoretical construct for the inference process of GPT. In our experiments, we observe that NRGPT can achieve similar performance to GPT and its recurrent variants on ListOps and Shakespeare, and competitive OWT generation diversity

despite worse OWT perplexity. However, NRGPT is computationally the gradient of an energy, which enforces weight sharing and limits how flexibly we can parameterize the architecture. We observe that this constraint also causes a larger amount of hyperparameter sensitivity than GPT variants. We also notice that the configurations of NRGPT that achieve competitive performance at comparable parameters actually have a slightly larger FLOP count than their standard transformer counterparts. Please see our extended discussion in Appendix B.3. In contrast to standard transformers, increasing the number of attention heads in NRGPT actually increases the parameters. We additionally observe that NRGPT has a more difficult time overfitting the training set, which is beneficial in small data regimes but is undesirable in the massive datasets used to train modern LLMs.

4.6 Discussion and Conclusions

We have presented NRGPT, a minimal modification of the GPT architecture that unifies autoregressive language modeling with energy-based modeling. Our analysis shows that under specific conditions on the inference rate matrix η, this process provably decreases energy after a transient period of time, providing a principled foundation for the dynamics, a phenomenon that we call asymptotic stability. Moreover, relaxing this constraint allows the model to learn its own energy exploration strategy for inference. Thus, our work complements previous studies suggesting that transformers perform GD during inference. Unlike past work, in our model inference is explicitly a gradient-based dynamics, while still maintaining an architecture very similar to GPT. Our experiments show that this framework performs comparably to a fully recurrent GPT model across parameter sizes while generally requiring fewer parameters. NRGPT represents a meaningful step toward understanding the architecture of transformers using energies.

CHAPTER 5
MEMORY IN PLAIN SIGHT

The generative process of Diffusion Models (DMs) has recently set state-of-the-art on many AI generation benchmarks. Though the generative process is traditionally understood as an "iterative denoiser", there is no universally accepted language to describe it. We introduce a novel perspective to describe DMs using the mathematical language of *memory retrieval* from the field of energy-based Associative Memories (AMs), making efforts to keep our presentation approachable to newcomers to both of these fields. Unifying these two fields provides insight that DMs can be seen as a particular kind of AM where Lyapunov stability guarantees are bypassed by intelligently engineering the dynamics of the denoising process. Finally, we present a growing body of evidence that records DMs exhibiting empirical behavior we would expect from AMs, and conclude by discussing research opportunities that are revealed by understanding DMs as a form of energy-based memory.

Previous chapters introduced explicit energy formulations for a transformer whose feedforward inference pass is governed by AM principles, advancing the dissertation's goal of showing that AM provides a reusable language for recognizing memory-like computation across systems that are not usually described as memories. This chapter extends that goal by asking whether the same computational motif is hiding inside one of modern generative modeling's most successful paradigms: Diffusion Models, where memory retrieval appears through engineered denoising dynamics rather than an explicit Lyapunov energy.

5.1 Introduction

Diffusion Models [5, 6, 119, 47] (DMs) have rapidly become the most performant class of generative models on images [120, 121, 122, 123, 124]. However, instead of decoding latent inputs into images using a single forward pass like both GANs [125] and VAEs [126] do, DMs iteratively apply a denoising function that makes some "latent" representation of the image look more and more like the training distribution. Training a DM consists of an unparameterized *forward process* where a known amount of noise is repeatedly added to corrupt an image, and a *reverse process* where the model learns to remove the noise added during the forward process. The computational power of the DM lies entirely in its reverse

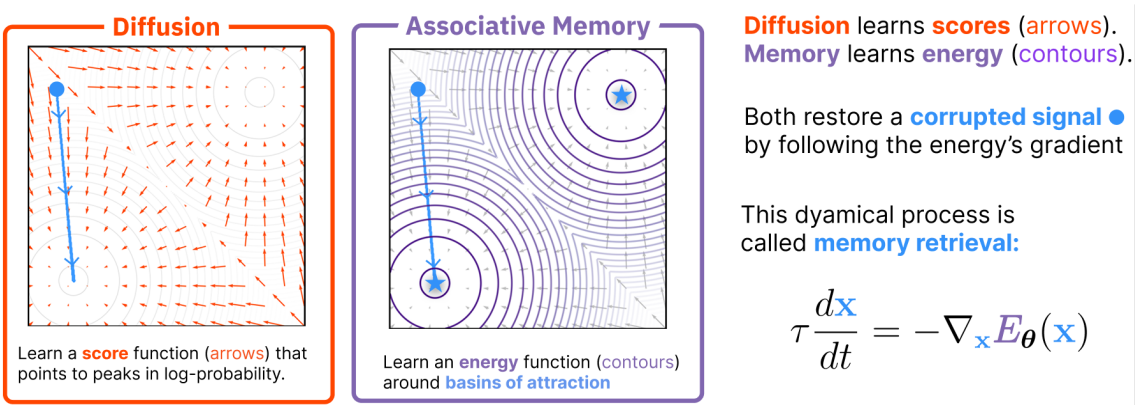

Figure 5.1: MEMORY IN PLAIN SIGHT uncovers the similarities between diffusion-model denoising and associative-memory recall, showing that diffusion models learn scores while associative memories learn energy contours. Both restore corrupted signals by following the energy gradient, but only energy-based AMs expose a Lyapunov energy for memory retrieval.

process, which is a sequence of denoising steps that aims to decrease the *energy* (equivalent to increasing the log-probability, see § 5.3) of some noisy sample, effectively performing *error correction*. The reverse process can generate novel images by "error correcting" pure noise.

In a seemingly unrelated field, energy-based AMs are formalized as dynamical systems that are concerned with the storage and retrieval of data or patterns [9, 10]. Like other Energy Based Models [36], AMs are described by an explicit and learnable energy function that places corrupted patterns at high energy and uncorrupted or "real-looking" patterns at low energy. The ensuing dynamics minimize the energy of an initial corrupted pattern, a process that performs *error correction*. A pattern is said to be "stored" (memorized) if it lives at a local minimum of the energy landscape. These local minima are called *memories* [75].

How do AMs retrieve memories? We can think of the energy landscape as a hilly terrain, where a "ball" is placed at some initial location (i.e., our corrupted pattern acting as a "query"). The ball rolls down the hill (the query evolves, becoming less corrupted), eventually settling at a local minimum (a memory) according to the dynamical equation and diagram in Figure 5.1. This process is formally called *memory retrieval*.

Memory retrieval looks eerily like the reverse process of DMs. Both approaches have the same goal of error correction:

> **Goal of both DMs and AMs**: *Given a corrupted representation of some data, recreate the original uncorrupted data by descending some energy.*

Yet, though DMs have been related to Markovian VAEs [127, 47, 5], Normalizing Flows [128], Neural ODEs [44], and generic Energy Based Models [6, 129], an explicit connection to AMs has not been fully explored, in part due to the lack of awareness about AMs. Such a connection would contribute to a growing body of literature that seeks to use modern AI techniques to unravel the mysteries of memory and cognition [130, 131, 132, 133, 134, 135, 136].

5.1.1 Related Surveys

The popularity of DMs has resulted in surveys that focus on the methods and diverse applications of DMs [137, 6, 138] alongside tutorial-style guides that seek to gently explain them [6, 139, 140]. In all cases, these surveys/guides make no connection to AMs. For an exhaustive reference on DM literature, [141] has collected a (still growing) list of >600 diffusion papers. Other surveys cover AMs and their applications [142, 143, 144, 145, 146], while others acknowledge high-level similarities between recurrent networks and AMs [147, 70]. We are aware of a concurrent work by [148] that is the closest in spirit to this work and whose valuable contributions we discuss in Subsection 5.5.1. This chapter provides a more thorough focus on AMs while emphasizing an approachable introduction to DMs without relying on stochasticity during memory retrieval [44].

5.1.2 Our Contributions

This chapter serves as a survey of the striking similarity between DMs and AMs, straddling a gap between traditional and modern AI research that is rarely bridged. This work then:

1. **Provides an approachable overview** of both AMs and DMs from the perspective of dynamical systems, energy, and Ordinary Differential Equations (ODEs) (§ 5.3 and § 5.4). We raise awareness about AMs by leaning into the popularity of DMs while simultaneously distilling the traditionally complex presentation of DMs using more intuitive descriptions of memory retrieval. We isolate the differences between DMs and AMs (e.g., AM architectures satisfy Lyapunov stability criteria that DMs do not) and discuss evidence that the differences are mitigated through the design and usage of DMs (§ 5.5).
2. **Highlights future research directions** at the intersection of DMs and AMs, made possible by characterizing their resemblance (§ 5.6). We also identify similarities that

AMs have to other modern architectures (e.g., transformers [3, 28, 11]). These similarities highlight mounting evidence that the field of AI is converging to models that strongly resemble AMs, escalating the urgency to understand AMs as an eminent paradigm for computation in AI.

5.2 Mathematical Notations

In this chapter we deviate from notations typically used for DMs and AMs. To minimize visual clutter, we prefer tensor notation over Einstein notation, representing scalars in non-bolded font (e.g., energies E or time t), and tensors in bold (e.g., state vector $\mathbf{x}$ or weight matrices $\boldsymbol{W}$). These distinctions also apply to scalar- and tensor-valued functions (e.g., energy scalar $E(\cdot)$ vs. activation vector $\hat{\mathbf{x}}(\cdot)$). A collection of learnable parameters is expressed through the generic variable $\boldsymbol{\theta}$. Gradients are expressed using "nabla" notation of a scalar valued function, where $\nabla_{\mathbf{x}}(\cdot)$ will be of the same shape as $\mathbf{x}$. The transpose is represented using $\mathbf{x}^\intercal$ notation, whereas $\mathbf{x}^T$ represents the vector $\mathbf{x}$ occurring at time T.

5.3 Diffusion Models

As generative models, Diffusion Models (DMs) seek to synthesize new data by sampling from some learned approximation $p_{\boldsymbol{\theta}}$ of the data's probability density function (p.d.f.) p_{data}. However, rather than learn the likelihood $p_{\boldsymbol{\theta}}$ itself, DMs use their parameters to approximate the gradient of the log-likelihood a.k.a. the **score function** $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$; thus, DMs are considered a class of *score-based models* [6, 44, 149, 150]. Mathematically, any p.d.f. can be expressed in terms of an **energy function** $E_{\boldsymbol{\theta}}(\mathbf{x})$ by the Boltzmann Distribution

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}}{Z_{\boldsymbol{\theta}}}, \tag{5.1}$$

where $Z_{\boldsymbol{\theta}}$ is the partition function (i.e., the normalizing constant enforcing that $\int p_{\boldsymbol{\theta}}(\mathbf{x}) d\mathbf{x} = 1$). Energy $E_{\boldsymbol{\theta}}$ is the negative log likelihood up to the additive constant $\log Z_{\boldsymbol{\theta}}$, and the score function is defined as the negative gradient of the energy:

$$\mathbf{f}_{\boldsymbol{\theta}}(x) \triangleq \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x}). \tag{5.2}$$

Figure 5.1 depicts the score function as vectors pointing to peaks in the log probability. We often think of the score function as predicting the noise we need to remove from a data point

$\mathbf{x}^t$ at some time t, where *adding* the estimated score $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}^t)$ in Equation (5.3) is the same as *removing* the predicted noise. Thus, given a neural network trained to predict the score, the process of generating data using score-based models can be construed (in discrete time) as an iterative procedure that adds the predicted score (subtracts the predicted energy gradient) for some fixed number of steps T on pure noise $\mathbf{x}^0$. The final state $\mathbf{x}^T$ is declared to be a local peak (minimum) of the log-likelihood (energy) and should now look like a sample drawn from the original distribution p_{data}. This process is described in Equation (5.3), where $\alpha \in \mathbb{R}$ is a step size in the direction $F_{\boldsymbol{\theta}}$:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \alpha \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}^t), \qquad t = 0, \ldots, T-1. \tag{5.3}$$

To be more precise, DMs use a score function $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; t)$ and step size $\alpha(t)$ (a.k.a. the "scheduler") that are both conditioned on time t [6]. Also, note that Equation (5.3) uses the convention that *time progresses forward when reconstructing the data*. However, the literature around DMs describes *time in the reconstruction process as going backwards*, denoting $\mathbf{x}^T$ to refer to the sample drawn from pure noise and $\mathbf{x}^0$ to refer to the final reconstructed sample drawn from $p_{\boldsymbol{\theta}}$ [5, 6, 47, 127, 44]. Equation (5.4) rewrites Equation (5.3) using the variable $s \triangleq T - t$ to represent the reverse-time convention used in most DM papers:

$$\mathbf{x}^{s-1} = \mathbf{x}^s + \alpha(s) \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}^s; s), \qquad s = T, \ldots, 1. \tag{5.4}$$

Using score-based models is then conceptually very simple: they seek to maximize (minimize) the log-likelihood (energy) of an initial sample by following the score $\mathbf{f}_{\boldsymbol{\theta}}$. However, DMs require several tricks to train, with the most popular being the technique of *denoising score-matching* [151, 6, 42, 152, 43, 47]. To train with denoising score-matching, samples $\mathbf{x}$ from our original data distribution p are repeatedly perturbed with small amounts of (time-dependent) noise $\eta(s)$. We then train our score-based model $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}^{s+1}; s)$ to remove the added noise. If the noise is small enough everywhere, we can guarantee that our optimal score function (parameterized by optimal weights $\boldsymbol{\theta}^*$) approximates the score of the true data distribution: i.e., $\mathbf{f}_{\boldsymbol{\theta}^*}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ and $\mathbf{f}_{\boldsymbol{\theta}^*}(\mathbf{x}^{s+1}; s) \approx -\eta(s)$.

5.3.1 Diffusion Models are Continuous Neural ODEs

The original DMs [5, 6] relied on a fixed number of discrete and stochastic steps in the forward and reverse processes. This changed when [44] introduced a *probability flow*

Ordinary Differential Equation (PF-ODE) formulation for DMs that formulated DMs in continuous time along deterministic trajectories.

Consider the standard form of a generic ODE under constrained time s

$$\frac{d\mathbf{x}}{ds} = \mu(\mathbf{x}; s), \qquad s \in [T, 0], \tag{5.5}$$

where $\mu(\mathbf{x}; s)$ is an arbitrary *drift function* representing some deterministic change in position of particle $\mathbf{x}^s$ at time s. DMs need to further corrupt the input $\mathbf{x}$, so we add to Equation (5.5) an infinitesimal amount of noise $\frac{d\mathbf{w}}{ds}$ scaled by some real-valued and time-dependent *diffusion coefficient* σ_s. The forward process of PF-ODEs is thus an Itô Stochastic Differential Equation (SDE) [44]:

$$\frac{d\mathbf{x}}{ds} = \mu(\mathbf{x}; s) + \sigma_s \frac{d\mathbf{w}}{ds}. \tag{5.6}$$

[153] argues that the equation above can be further simplified without any loss in performance by assuming constant drift function $\mu(\mathbf{x}; s) = 0$, a convention adapted by [154] to set SOTA one-step generation with DMs. This convention simplifies Equation (5.6) to make the forward process depend only on the diffusion coefficient and the infinitesimal random noise, as in

$$\frac{d\mathbf{x}}{ds} = \sigma_s \frac{d\mathbf{w}}{ds}. \tag{5.7}$$

The reverse process now depends only on the noise scale σ_s and the score $\mathbf{f}_{\boldsymbol{\theta}}$ [44, 154]:

$$\frac{d\mathbf{x}}{ds} = -\frac{1}{2}\sigma_s^2 \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; s). \tag{5.8}$$

We have written the strange "reverse time" convention of DMs using time variable $s \triangleq T - t$. Equation (5.9) rewrites Equation (5.8) using forward time t, collecting the noise scale into a real-valued time-variable $\tau(t) \triangleq \frac{2}{\sigma_t^2}$ to control the rate of change:

$$\tau(t)\frac{d\mathbf{x}}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; t)\,, \qquad t \in [0, T]. \tag{5.9}$$

PF-ODEs generalize previous theories of DMs by removing the mathematical dependency on discrete time and stochasticity in the reverse process. At the same time, the continuous dynamics of PF-ODEs exposes a strong mathematical connection to AMs and deterministic memory retrieval that was difficult to see before.

5.4 Energy-Based AMs

In this chapter, AM means an energy-based associative memory with an explicit Lyapunov energy. These AMs are explicit energy functions that are smooth and bounded from below, ensuring that the retrieval dynamics decrease the energy over time and approach stable fixed points under standard regularity assumptions.

An AM stores memories as local minima of a learnable energy function $E_{\boldsymbol{\theta}} \in \mathbb{R}$. Given an input pattern $\mathbf{x}^0$ at time $t = 0$, retrieval evolves the hidden state toward a fixed point that represents a memory learned from the original data:

$$\tau \frac{d\mathbf{x}}{dt} = -\nabla_{\hat{\mathbf{x}}} E_{\boldsymbol{\theta}}(\hat{\mathbf{x}}) \,, \quad t > 0. \tag{5.10}$$

The activation $\hat{\mathbf{x}}(\mathbf{x})$ is the conjugate variable of $\mathbf{x}$ induced by the Legendre transform of a convex Lagrangian [75, 77], as explained in more detail in § 8.2. Here, it is enough to treat $\hat{\mathbf{x}}$ as a monotonic activation function that makes the energy descent well behaved.

Equation (5.10) can of course be discretized and treated as a neural network that is recurrent through time:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \frac{dt}{\tau} \nabla_{\hat{\mathbf{x}}} E_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^t). \tag{5.11}$$

The energy $E_{\boldsymbol{\theta}}$ acts as a Lyapunov function because retrieval via deterministic energy minimization *cannot increase it*. This is due to the Legendre construction of [75, 77] which makes the activation Jacobian $\frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{x}}$ positive semidefinite,

$$\begin{aligned} \frac{dE_{\boldsymbol{\theta}}}{dt} &= \nabla_{\hat{\mathbf{x}}} E_{\boldsymbol{\theta}}(\hat{\mathbf{x}})^\intercal \frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{x}} \frac{d\mathbf{x}}{dt} \\ &= -\frac{1}{\tau} \nabla_{\hat{\mathbf{x}}} E_{\boldsymbol{\theta}}(\hat{\mathbf{x}})^\intercal \frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{x}} \nabla_{\hat{\mathbf{x}}} E_{\boldsymbol{\theta}}(\hat{\mathbf{x}}) \leq 0. \end{aligned} \tag{5.12}$$

Thus, if the energy is bounded from below, the energy converges to a limiting value. Under the usual AM assumption that the relevant critical points are isolated, the state approaches a fixed point $\mathbf{x}^\star$ asymptotically rather than necessarily reaching it at a finite time:

$$\lim_{t \to \infty} \frac{d\mathbf{x}}{dt} = 0 \,, \quad \text{and} \quad \mathbf{x}^\star = \mathbf{x}^\star - \frac{dt}{\tau} \nabla_{\hat{\mathbf{x}}} E_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^\star) \,.$$

That energy-based AMs have a tractable Lyapunov function is their core distinction from

DM theory and implementations.

Table 5.1: Summarizing the similarities and differences between DMs and AMs. Fields marked with a * indicate caveats. See Subsection 5.5.2 for details.

	Diffusion	**AM**
Parameterizes the ...	Score function $\mathbf{f}_{\boldsymbol{\theta}}$	Energy function $E_{\boldsymbol{\theta}}$
Continuous Update	$\tau \frac{d\mathbf{x}}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})$	$\tau \frac{d\mathbf{x}}{dt} = -\nabla_{\hat{\mathbf{x}}} E_{\boldsymbol{\theta}}(\hat{\mathbf{x}})$
Discrete Update	$\mathbf{x}^{t+1} = \mathbf{x}^t + \alpha \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}^t)$	$\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha \nabla_{\hat{\mathbf{x}}} E_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^t)$
Valid Time Domain	$t \in [0, T]$	$t \geq 0$
Fixed Point Attractor?	No*	Yes
Tractable Energy?	No*	Yes
Undoes Corruption of ...	Noise it was trained on*	Any kind

5.5 The Uncanny Resemblance of AMs and DMs

AMs are different from DMs in that they are not primarily understood as generative models. However, both AMs and DMs can be written as systems that move noisy input patterns "downhill" in an energy-like landscape. Thus, memory retrieval can be easily be construed as a "generation process" that samples from some learned distribution, just as the reverse process of DMs does. This realization makes it possible to directly compare AMs to DMs, and explore what could replace the AM stability guarantee in DMs, if anything.

The important difference is the source of stability. This chapter has emphasized the definition of energy-based AMs as systems with an explicit Lyapunov energy to certify stability. In DMs, the motion is shaped by a learned score, a noise schedule, and a stopping time — techniques that engineer a form of convergence guarantee to avoid inheriting instability from our choice of neural network architecture $\mathbf{f}_{\boldsymbol{\theta}}$. We tabulate these differences in more detail in Table 5.1.

Yet beyond this, both methods share incredible similarities:

- **Both model the energy.** DMs learn a parameterized score function $\mathbf{f}_{\boldsymbol{\theta}}$ to approximate the gradient of some true energy function E such that $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) \approx -\nabla_{\mathbf{x}} E(\mathbf{x}) \; \forall \mathbf{x}$. In AMs, this energy function is explicit and is defined by using architectures that directly model the

energy $E_{\boldsymbol{\theta}}$.

- **Both generate samples by descending the predicted gradient of the energy.** DMs directly output the estimated score $\mathbf{f}_{\boldsymbol{\theta}} \approx -\nabla_{\mathbf{x}} E(\mathbf{x})$, whereas AMs will directly output a smooth energy $E_{\boldsymbol{\theta}}(\hat{\mathbf{x}})$ on which the gradient $-\nabla_{\hat{\mathbf{x}}} E_{\boldsymbol{\theta}}(\hat{\mathbf{x}})$ can be directly calculated and descended. The update rules of both have a formally analogous gradient-descent form as in Equation (5.9) and Equation (5.10).

- **Both produce a solution that lies in the neighborhood of a local energy minimum.** In DMs, this behavior is a consequence of the manner in which it is trained: the final output $\mathbf{x}^T$ exists in a region such that a small perturbation in any direction would increase its energy. In AMs, this statement is a requirement of the Lyapunov function; if the dynamics run for a sufficiently long time, they approach a fixed point $\mathbf{x}^\star$ that lies at a local energy minimum.

5.5.1 Situations of Precise Equivalence

Concurrently to this work, [148] has uncovered a precise mathematical connection between DMs and the Modern Hopfield Network (MHN). Specifically, if we assume N discrete stored patterns and a time dependent temperature $\beta_t^{-1} = (T-t)\sigma_t^2$ that captures both the noise schedule σ_t and a DM's constrained dynamics of $t < T$, an energy function that produces DM dynamics [155] is mathematically equivalent (up to a constant) to a MHN with exponential energy and time dependence:

$$\begin{aligned} E^{\mathrm{DM}}(\mathbf{x}, t) &= -\sigma_t^2 \log \left(\frac{1}{N} \sum_{n=1}^{N} e^{-\frac{||\mathbf{x}-\mathbf{y}_n||_2^2}{2(T-t)\sigma_t^2}} \right) \\ &\iff -\beta_t^{-1} \log \left(\sum_{n=1}^{N} e^{\beta_t \mathbf{x}^\intercal \mathbf{y}_n} \right) + \frac{||\mathbf{x}||_2^2}{2} = E^{\mathrm{MHN}}(\mathbf{x}, t). \end{aligned} \tag{5.13}$$

From this equation, [148] claims that DM theory "can be seen as a wide generalization of the classical theory of energy-based AMs". In this work, we emphasize a different viewpoint: DMs can be seen as a particular extension of AM theory that bypasses the stability guarantees of a Lyapunov energy by intelligently scheduling the noise (temperature) of a generative process that performs memory retrieval. In the single layer MHN, it is possible to find an energy function that will approximate it (e.g., the mixture-of-Gaussian like energy of Equation (5.13)); we believe that there is need to explore the mathematical

equivalence of deeper DM architectures (e.g., U-Nets [156]) and hierarchical AMs [77].

5.5.2 Reconciling the Differences

DMs and AMs are certainly not equivalent methods. However, we discover evidence that the theoretical gaps between DMs and AMs are not so significant in practice.

Observation 1: DMs approximate fixed point attractors. Though the dynamics of DMs have no theoretical guarantees of fixed point attractors, we notice that the design of DMs seems to intelligently engineer the behavior of fixed-point attractors without constraining the architecture itself. We identify two fundamental tricks used by DMs that help approximate stable dynamics:

Trick 1 DMs explicitly halt their reconstruction process at time $t = T$ (i.e., requiring $\mathbf{x}^\star = \mathbf{x}^T$ and added noise $\eta(T) = 0$) and are thus only defined for time $t \in [0, T]$. $\mathbf{x}^T$ then represents a *contrived fixed point* because no further operations change it. We can say that $\mathbf{x}^{t \neq T}$ corresponds to a data point with some corruption and $\mathbf{x}^T$ corresponds to a *memory* in the language of AMs.

Trick 2 We know that $\mathbf{x}^T$ approximates a local energy minimum because of the *noise annealing* trick introduced by [6] and used by all subsequent DMs. During training, datapoints are perturbed with gradually increasing amounts of noise such that small noise is added around the point itself. This leads to a robust approximation of the true energy gradient localized around each data point where the original data point lies at the minimum. [148] proved that this technique is equivalent to temperature annealing a softmax activation function during memory retrieval in the MHN.

We additionally see evidence of DMs storing "memories" that are actual training points. [157] showed that one can retrieve training data almost exactly from publicly available DMs by descending an energy conditioned on prompts from the training dataset. It seems that this behavior is particularly evident for images considered outliers and for images that appear many times. Viewing DMs as AMs, this behavior is not surprising, as the whole function of AMs is to retrieve close approximations to data it has seen before.

Tricks 1 and 2 imply that DMs are inescapably bound to a knowledge of the current time t. The time t defines both the total noise the model should expect in a given pattern (i.e., the standard deviation or the "width" of Gaussian peaks around each data point) and how much

noise the model should expect to remove (i.e., the step size down the energy). Given a pattern with an unknown quantity of noise, a user must either make a "best guess" for the time t corresponding to this amount of noise, or restart the dynamics at time $t = 0$ to reset the "noise scheduler", which causes the model to make large jumps around the energy landscape and likely land it in a distant energy minimum. Currently, AMs have no such dependence between corruption levels and time t, though it is easy to include time-dependence into the memory retrieval dynamics of AMs.

Observation 2: DMs can undo more than Gaussian noise. In order for a DM to behave like an AM, it must be possible to undo any kind of corruption (e.g., inpainting, blur, pixelation, etc.), not just the white- or Gaussian-noise associated with Brownian motion as originally formulated in [5, 6]; this is because all corruptions are a form of error that causes samples to have higher energy. [158, 159] showed that the performance of DMs can actually improve when considering other types of noisy corruption in the forward process. However, it also seems that DMs can learn to reverse any kind of corrupting process. [160] demonstrates that DMs can be trained to invert arbitrary image corruptions that generate samples almost as well as those trained to invert only Gaussian noise introduced by the Brownian motion of the forward process presented in this work. Thus, DMs exhibit AM behavior by following a general score function that can seemingly recover "fixed points" of some energy landscape by undoing arbitrary corruptions.

Observation 3: Unconstrained DMs work best with special architectures. One advantage of DMs over AMs is that they are "unconstrained" and can use any neural network architecture to approximate the score function; that is, the architecture is not required to be the gradient of an actual scalar energy. The only requirement is that the neural network chosen to approximate the score must be *isomorphic* such that the function's output is the same shape as its input (i.e., $\mathbf{f}_{\boldsymbol{\theta}} : \mathbb{R}^d \mapsto \mathbb{R}^d$). However, not all isomorphic architectures are created equal and only select architectures are used for DMs in practice. Both standard feedforward networks [150] and vanilla transformers have struggled to generate quality samples using diffusion [161, 162]. Most applications of DMs use some modification of the U-Net architecture [156] originally used by [47], though the original DM paper [5] used shallow MLPs, and recent work [161] has shown how to engineer Vision Transformers [74] to achieve a similar reconstruction quality as U-Nets on images.

Observation 4: DMs work with explicit energy. Though DMs characterize an energy landscape by modeling its gradient everywhere, they do not inherently have a concept of the energy value itself. However, [44] showed that one can actually compute an exact energy for DMs using the instantaneous change of variables formula from [163], with the caveat that this equation is expensive to compute. Estimations of the energy are preferred over direct computation in practice [128].

Another approach for enforcing an energy on DMs is to choose an architecture that parameterizes an actual energy function, whose score function then describes a conservative gradient field. [150] researched exactly this, exploring whether a generic learnable function $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; t)$ that is constrained to be the true gradient of a parameterized energy function as in Equation (5.14) is able to attain sample quality results similar to those of unconstrained networks.

$$E_{\boldsymbol{\theta}}(\mathbf{x}; t) = \frac{1}{2\sigma(t)} ||\mathbf{x} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; t)||^2 \tag{5.14}$$

The score $\mathbf{f}_{\boldsymbol{\theta}}$ of this energy can be written by computing the analytical gradient

$$\begin{aligned} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; t) = &\frac{\mathbf{x} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; t)}{\sigma(t)} \nabla_{\mathbf{x}} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; t) \\ &- \frac{\mathbf{x} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; t)}{\sigma(t)} . \end{aligned} \tag{5.15}$$

[150] notes that the second term of the equation is the standard DM, while the first term involving $\nabla_{\mathbf{x}} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; t)$ is new and helps guarantee that $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; t)$ is a conservative vector field. They showed that constraining the score function to be the gradient of the energy in Equation (5.14) does not hurt generation performance and provides hope that AMs with constrained energy can one day match the performance of unconstrained DMs.

5.6 Conclusions & Open Challenges

DMs and AMs have remarkable similarities when presented using a unified mathematical notation: both aim to minimize some energy by following its gradient. The output of both approaches represents some sort of *memory* that lies in a local minimum (maximum) of the energy (log-probability). However, these approaches are not identical as evidenced by different validity constraints on architectures and time domains. The training philosophy

behind each approach is also different: DMs assume that the energy function is intractable and learn the gradient using known perturbations of data as the objective, while AMs focus on learning the fixed points of a tractable energy.

5.6.1 Directions for AMs

AMs have not gained the traction of DMs in AI applications. Many researchers in the field focus on shallow Hopfield-style architectures, trying to improve their theoretical memory capacity [164, 165] or apply related models to modern problems [166]. Other researchers are connecting the memory-retrieval capabilities of AMs with existing feed-forward architectures like the transformer [28, 167]; in doing so they discard the idea of global optimization on the energy. In part, these other research directions exist because no pure AM has shown impressive performance on large data until the ENERGY TRANSFORMER [11] (Chapter 3) introduced a true Lyapunov energy that can be trained and used in the same manner as conventional transformers. However, even this AM does not yet show significant performance gain over traditional methods. The empirical success of DMs across many domains should provide hope that modern AM architectures [77, 17] can achieve performance parity on similar tasks. Constrained DMs show no worse performance than unconstrained models [150], and HAM theory [77] can build AMs that resemble U-Nets [17] and transformers [11].

5.6.2 Directions for Diffusion Models

DM researchers should find the theoretical framework of the Lyapunov function from AMs compelling: in the absence of the Lyapunov function, DMs must manufacture fixed points using dynamic noise schedules and step sizes. However, if the score function of DMs is shown to be a conservative vector field as in [150], perhaps DMs have learned fixed point structure and can behave well in all time $t > T$. Viewing DMs as fixed-point attractors like AMs would additionally allow us to theoretically characterize its memory capacity (where "memory capacity" can be seen as a proxy for the "scaling laws", see Subsection 5.6.3). Finally, viewing the sampling procedure from DMs as a form of gradient descent (as in AMs) allows optimization methods like ADAM [168] and L-BFGS [169] to be used to sample from the probability distribution.

5.6.3 Scaling Laws from the Perspective of AMs

The performance of transformer language models is famously characterized by the "scaling laws", which claim that a model's performance will improve as a power-law with model size, dataset size, and the amount of compute used for training [170]. DMs exhibit similar scaling behaviors [171], where larger models perform better than smaller ones in a predictably improving trend. However, the "scaling laws" are empirical and there is no agreed-upon theory to justify why a model's performance would continue to grow with the model size.

AMs offer one possible answer by characterizing large-model performance as one of *memory capacity* (reviewed in Chapter 2), as recently noted by [172, 173]. In the world of AMs, learned parameters can support collections of attractors in the data space; thus, more parameters can mean more local energy minima (memories). Similarly, more data means the parameters can identify more meaningful local minima in the energy. More compute, as measured in terms of model depth or number of iterations down an energy landscape, means more optimal (lower energy) retrievals. These hypotheses are still unexplored research questions that come from intuitively understanding large models as AMs.

5.6.4 Closing Remarks

Very few researchers will observe the rapid advances of AI today and notice a trend towards the dynamical processes of AMs first established by John Hopfield in the 1980s. However, many of the theoretical guarantees of AMs are captured in the design of increasingly popular DMs that have proven themselves fixtures for many applications of generative modeling. This chapter is a first step towards a more comprehensive understanding of the connections between diffusion models and energy-based associative memories. We hope that our work inspires further research into these exciting fields and that it helps to foster a new generation of AI systems that are capable of unlocking the secrets of memory and perception.

Part II

GENERALIZING ASSOCIATIVE MEMORY

Large-capacity energy-based associative memories, commonly called Dense Associative Memories (DenseAMs) [26], solve the storage-capacity problem that limited classical Hopfield Networks. Yet their success introduces two new obstacles: their exponential memory capacity is tied to the size of an explicit synaptic matrix such that storing new patterns requires physically adding new connections, and their ability to retrieve stored memories directly competes with their ability to generate new ones. Part II tackles both obstacles by exploiting the little-known connection between DenseAM energies and kernel-density estimation (KDE). Better tooling for DenseAMs is especially salient since these improvements can propagate to richer AM architectures: DenseAM energies also appear as core components in the Energy Transformer (Part I) and larger hierarchical AMs (HAMs, Part III).

This part begins with DrDAM, which shows that random features enable DenseAMs to compress patterns without adding synaptic slots for every stored pattern. LSRDAM then shows that changing the kernel inside the energy function can create large numbers of "emergent" memories while preserving exact retrieval of the stored memories. Together, these works show that kernel methods can help break the limitations inherent to early formulations of DenseAMs.

Chapter 6

DrDAM: Dense Associative Memory through the Lens of Random Features. Benjamin Hoover, Duen Horng Chau, Hendrik Strobelt, Parikshit Ram, and Dmitry Krotov. *Advances in Neural Information Processing Systems (NeurIPS), 2024*. PDF

Chapter 7

LSRDAM: Dense Associative Memory with Epanechnikov Energy. Benjamin Hoover, Zhaoyang Shi, Krishnakumar Balasubramanian, Dmitry Krotov, and Parikshit Ram. *Advances in Neural Information Processing Systems (NeurIPS), 2025*. PDF

CHAPTER 6
DRDAM: DENSEAM THROUGH THE LENS OF RANDOM FEATURES

Dense Associative Memories (DenseAMs) are high storage capacity variants of the Hopfield networks that are capable of storing a large number of memory patterns in the weights of the network of a given size. Their common formulations typically require storing each pattern in a separate set of synaptic weights, which leads to the increase of the number of synaptic weights when new patterns are introduced. In this work, we propose an alternative formulation of this class of models using random features, commonly used in kernel methods. In this formulation the number of network's parameters remains fixed. At the same time, new memories can be added to the network by modifying existing weights. We show that this novel network closely approximates the energy function and dynamics of conventional DenseAMs and shares their desirable computational properties.

Part I treated energy-based AMs as a way to design modern neural architectures whose inference pass is governed by energy descent. This chapter begins the second movement of the dissertation: treating AMs as algorithmic objects whose memory can be compressed, updated, and analyzed without storing every pattern as a separate parameter block.

6.1 Introduction

The Hopfield Network is an elegant early energy-based AM model that makes it possible to store a set of memory patterns in the synaptic weights of the neural network [9]. For a given prompt $\sigma_i(t = 0)$, which serves as the initial state of that network, the neural update equations drive the dynamical flow towards one of the stored memories. For a system of K memory patterns in the D-dimensional binary space the network's dynamics can be described by the temporal trajectory $\sigma_i(t)$, which descends the energy function

$$E = -\sum_{\mu=1}^{K} \Big(\sum_{i=1}^{D} \xi_i^\mu \sigma_i \Big)^2 \tag{6.1}$$

Here ξ_i^μ (index $\mu = 1...K$, and index $i = 1...D$) represent memory vectors. The neural dynamical equations describe the energy descent on this landscape. In this formulation, which we call the **memory representation**, the geometry of the energy landscape is encoded

in the weights of the network ξ_i^μ, which coincide with the memorised patterns. Thus, in situations when the set of the memories needs to be expanded by introducing new patterns one must introduce additional weights.

Alternatively, one could rewrite the above energy in a different form, which is more commonly used in the literature. Specifically, the sum over the memories can be computed upfront and the energy can be written as

$$E = -\sum_{i,j=1}^{D} T_{ij}\sigma_i\sigma_j, \qquad \text{where} \qquad T_{ij} = \sum_{\mu=1}^{K} \xi_i^\mu \xi_j^\mu \tag{6.2}$$

In this form one can think about weights of the network being the symmetric tensor T_{ij} instead of ξ_i^μ. One advantage of formulating the model this way is that the tensor T_{ij} does not require adding additional parameters when new memories are introduced. Additional memories are stored in the already existing set of weights by redistributing the information about new memories across the already existing network parameters. We refer to this formulation as **distributed representation**. This distinction matters for the thesis-level story because it shows that AM computation is not intrinsically tied to literal per-pattern storage: the same attractor landscape can sometimes be encoded in a fixed synaptic object.

A known problem of the network (Equations (6.1) and (6.2)) is that it has a small memory storage capacity, which scales at best linearly as the size of the network D is increased [9]. This limitation has been resolved with DenseAMs, also known as Modern Hopfield Networks [26]. This is achieved by strengthening the non-linearities (interpreted as neural activation functions) in Equation (6.1), which can lead to the super-linear and even exponentially large memory storage capacity [26, 27]. Using continuous variables $\mathbf{x} \in \mathbb{R}^D$, the energy is defined as[1]

$$E = -Q\Big[\sum_{\mu=1}^{K} F\Big(S\big[\boldsymbol{\xi}^\mu, \mathbf{g}(\mathbf{x})\big]\Big)\Big], \tag{6.3}$$

where the function $\mathbf{g} : \mathbb{R}^D \to \mathbb{R}^D$ is a vector function (e.g., a sigmoid, a linear function, or a layernorm), the function $F(\cdot)$ is a rapidly growing separation function (e.g., power $F(\cdot) = (\cdot)^n$ or exponent), $S[\mathbf{x}, \mathbf{x}']$ is a similarity function (e.g., a dot product or a Euclidean

[1] Throughout this chapter, we use bold symbols for denoting vectors and tensors, e.g., $\boldsymbol{\xi}^\mu$ is a D-dimensional vector in the space of neurons for each value of index μ. Individual elements of those vectors and tensors are denoted with the same symbol, but with plain font. In the example above, these individual elements have an explicit vector index, e.g., ξ_i^μ. Same applies to vectors in the feature space introduced later.

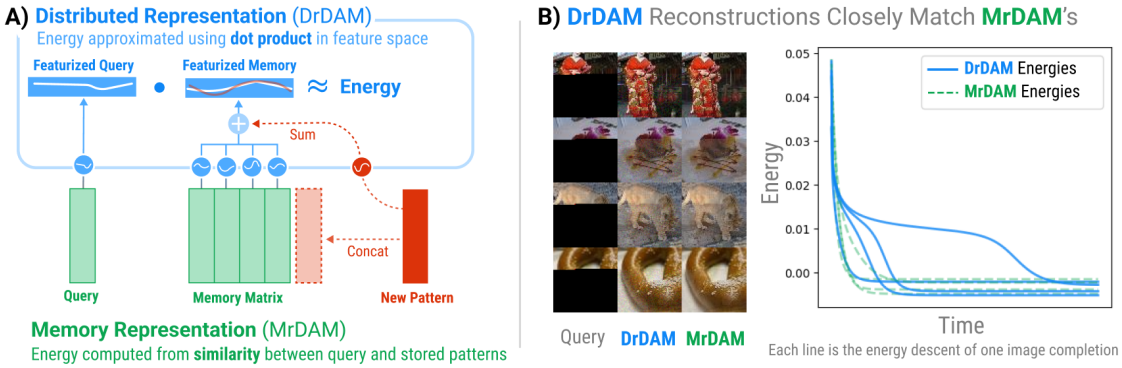

Figure 6.1: The **D**istributed **R**epresentation for **D**ense **A**ssociative **M**emory (**DRDAM**) approximates both the energy and fixed-point dynamics of the traditional **M**emory **R**epresentation for **D**ense **A**ssociative **M**emory (**MRDAM**) while having a parameter space of constant size. A) Diagram of DRDAM using a **basis function** parameterized by random features (e.g., see Equation (6.8)). In the distributed representation, adding new memories does not change the size of the memory tensor. B) Comparing energy descent dynamics between DRDAM and MRDAM on 3x64x64 images from Tiny Imagenet [1]. Both models are initialized on queries where the bottom two-thirds of pixels are occluded with zeros; dynamics are run while clamping the visible pixels and their collective energy traces shown. DRDAM achieves the same fixed points as MRDAM, and these final fixed points have the same energy. The energy decreases with time for both MRDAM and DRDAM, although the dependence of the energy relaxation towards the fixed point is sometimes different between the two representations. Experimental setup is described in Appendix C.4.

distance), and Q is a scalar monotone function (e.g., linear or logarithm). For instance, in order to describe the classical Hopfield network with binary variables (Equation (6.1)) one could take: linear Q, quadratic $F(\cdot) = (\cdot)^2$, dot product S, and a sign function for $g_i = sign(x_i) = \sigma_i$. There are many possible combinations of various functions $\mathbf{g}, F(\cdot), S(\cdot,\cdot)$ that lead to different models from the DenseAM family [26, 27, 28, 75, 164, 165]; many of the resulting models have proven useful for various problems in AI and neuroscience [70]. Diffusion models have been linked to even more sophisticated forms of the energy landscape [13, 148].

From the perspective of the information storage capacity DenseAMs are significantly superior compared to the classical Hopfield networks. At the same time, most[2] of the models from the DenseAM family are typically formulated using the memory representation, and for this reason require introduction of new weights when additional memory patterns are added to the network. The main question that we ask in this work is: *how can we combine superior memory storage properties of DenseAMs with the distributed (across synaptic weights) formulation of these models in the spirit of classical Hopfield networks* (Equation (6.2))? If

[2]This is true for all DenseAMs with the exception of the power model of Krotov and Hopfield [26], which can be written using n-index tensors $T_{i_1,i_2,\ldots,i_n}$ in analogy with the 2-tensor T_{ij} as in Equation (6.2).

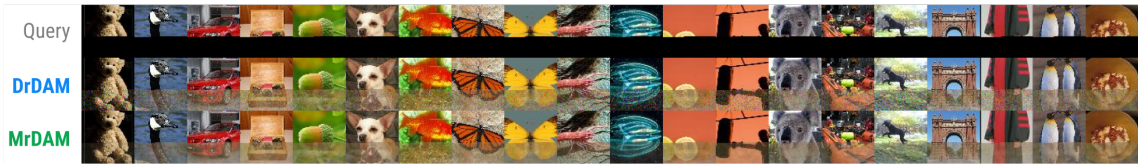

Figure 6.2: DRDAM achieves parameter compression over MRDAM, successfully storing 20 different 64x64x3 images from Tiny ImageNet [1] and retrieving them when occluding the lower 40% of each query. The memory matrix of MRDAM is of shape $(20, 12288)$ while the memory tensor of DRDAM is of shape $Y = 2 \cdot 10^5$, a $\sim$20% reduction in the number of parameters compared to MRDAM; all other configurations for this experiment match those in Appendix C.4. Further compression can be achieved with a higher tolerance for DRDAM's retrieval error, smaller β, and fewer occluded pixels, see § 6.4. **Top:** Occluded query images. **Middle:** Fixed-point retrievals from DRDAM. **Bottom:** (ground truth) Fixed-point retrievals of MRDAM.

such a formulation is found, it would allow us to add memories to the existing network by simply recomputing already existing synaptic weights, without adding new parameters.

A possible answer to this question is offered by the theory of random features and kernel machines. Given an input domain $\mathcal{X}$, kernel machines leverage a positive definite *Mercer kernel function* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ that measures the similarity between pairs of inputs. The renowned "kernel trick" allows one to compute the inner-product

$$k(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\varphi}(\mathbf{x}') \rangle = \sum_{\alpha=1}^{Y} \varphi_\alpha(\mathbf{x}) \varphi_\alpha(\mathbf{x}') \tag{6.4}$$

between two inputs $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ in a rich feature space defined by the feature map $\boldsymbol{\varphi}(\mathbf{x})$ without ever explicitly realizing the feature map $\boldsymbol{\varphi}(\mathbf{x})$. Various machine learning models (such as support vector machines [53], logistic regression, and various others [174, 175]) can be learned with just access to pairwise inner-products, and thus, the kernel trick allows one to learn such models in an extremely expressive feature space. Kernel functions have been developed for various input domains beyond the Euclidean space such as images, documents, strings (such as protein sequences [176]), graphs (molecules [177], brain neuron activation paths) and time series (music, financial data) [178]. Common kernels for Euclidean data are the radial basis function or RBF kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|_2^2)$ and the polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + b)^p$. To appreciate the expressivity of these kernel machines, note that, for input domain $\mathbb{R}^D$, the RBF kernel corresponds to an infinite dimensional feature space ($Y = \infty$) and the polynomial kernel to a $O(D^p)$ dimensional feature space.

Interpreting the composition of the separation and similarity functions in Equation (6.3) as the left hand side of the kernel trick Equation (6.4) we can map the energy into the

feature space, using appropriately chosen feature maps. Subsequently, the order of the summations over memories and features can be swapped, and the sum over memories can be computed explicitly. This makes it possible to encode all the memories in a tensor T_α, which we introduce in section § 6.3, that contains all the necessary information about the memories. The energy function then becomes defined in terms of this tensor only, as opposed to individual memories. This functionality is summarized in Figure 6.1. Additionally, we show examples of retrieved Tiny ImageNet images that have been memorised using the original DenseAM model, which we call MRDAM, and the "featurized" version of the same model, which we call DRDAM (please see the explanations of these names in the caption to Figure 6.1). These examples visually illustrate that mapping the problem into the feature space preserves most of the desirable computational properties of DenseAMs, which normally are defined in the "kernel space". The next chapter uses the same kernel vocabulary for a complementary purpose: DRDAM uses kernels to compress the representation of stored patterns, while LSRDAM uses kernel choice to reshape the energy landscape and control emergent memories.

Contributions:

- We propose a novel approximation of a DenseAM network utilizing random features commonly used in kernel machines. This novel architecture does not require the storage of the original memories, and can incorporate new memories without increasing the size of the network.

- We precisely characterize the approximation introduced in the energy descent dynamics by this architecture, highlighting the different critical factors that drive the difference between the exact energy descent and the proposed approximate one.

- We validate our theoretical guarantee with empirical evaluations.

In the past, kernel trick has been used for optimizing complexity of the attention mechanism in transformers [55], and those results have been recently applied to AM retrieval [179], given the various connections between transformers and DenseAMs [28, 11]. Existing studies [55, 179] focus on settings when attention operation or AM retrieval is done in a single step update. This is different from our goals here, which is to study the recurrent dynamics of AM updates and convergence of that dynamics to the attractor fixed points.

Iatropoulos et al. [56] propose kernel memory networks which are a recurrent form of a kernel support vector machine, and highlight that DenseAM networks are special cases of these kernel memory networks. Making a connection between nonparametric kernel regression and energy-based AMs, Hu et al. [180] propose a family of provably efficient sparse Hopfield networks [57, 58], where the dynamics of any given input are explicitly driven by a subset of the memories due to various entropic regularizations on the energy. DenseAMs have been also used for sequences [181, 182, 58]. To reduce the complexity of computing all the pairs of $F(S[\boldsymbol{\xi}, \mathbf{x}])$ for a given set of memories and queries, Hu et al. [183] leverage a low-rank approximation of this separation-similarity matrix using polynomial expansions. The kernel trick has also recently been used to increase separation between memories (with an additional learning stage to learn the kernel), thereby improving memory capacity [59]. There are also very recent theoretical analysis of the random feature Hopfield networks [184, 185], where their focus in on the construction of memories using random features. Kernels are also related to density estimation [61], and recent works have leveraged a connection between mixtures of Gaussians and DenseAMs for clustering [186, 187]. Lastly, random features have been used for biological implementations of both transformers and DenseAMs [136, 188].

To the best of our knowledge there is no rigorous theoretical and empirical comparison of DenseAMs and their distributed (featurized) variants in recurrent memory storage and retrieval settings, as well as results pertaining to the recovery of the fixed points of the energy descent dynamics. This is the main focus of our work.

6.2 Technical background

Given the energy function in Equation (6.3), a variable $\mathbf{x}$ is updated in the forward pass through the "layers" of this recurrent model such that its energy decreases with each update. If the energy is bounded from below, this ensures that the input will (approximately) converge to a local minimum. This can be achieved by performing a "gradient descent" in the energy landscape. Considering the continuous dynamics, updating the input $\mathbf{x}$ over time with $d\mathbf{x}/dt$, we need to ensure that $dE/dt < 0$. This can be achieved by setting $d\mathbf{x}/dt \propto -\nabla_{\mathbf{x}} E$.

Discretizing the above dynamics, the update of an input $\mathbf{x}$ at the t-th recurrent layer is given by:

$$\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)} - \eta^{(t-1)} \nabla_{\mathbf{x}} E^{(t-1)}, \tag{6.5}$$

where $\eta^{(t)}$ is a (step dependent) step-size for the energy gradient descent, $E^{(t)}$ is the energy of the input after the t-th layer, and the input to the first layer $\mathbf{x}^{(0)} \leftarrow \mathbf{x}$. The final output of the AM after L layers is $\mathbf{x}^{(L)}$.

DenseAMs significantly improve AM capacity by utilizing rapidly growing nonlinearity-based separation-similarity compositions such as $F(S[\mathbf{x}, \boldsymbol{\xi}^\mu]) = \exp(\beta \langle \mathbf{x}, \boldsymbol{\xi}^\mu \rangle)$ or $F(S[\mathbf{x}, \boldsymbol{\xi}^\mu]) = \exp(-\beta/2 \|\mathbf{x} - \boldsymbol{\xi}^\mu\|_2)$ or $F(S[\mathbf{x}, \boldsymbol{\xi}^\mu]) = (\langle \mathbf{x}, \boldsymbol{\xi}^\mu \rangle)^p, p > 2$, among other choices, with $\beta > 0$ corresponding to the *inverse temperature* that controls how rapidly the separation-similarity function grows. However, these separation-similarity compositions do not allow for the straightforward simplifications as in Equation (6.2), except for the power composition. For a general similarity function, the update based on gradient descent over the energy in Equation (6.3) is given by:

$$\nabla_{\mathbf{x}} E = - \left. \frac{dQ(y)}{dy} \right|_{y=\sum_\mu F(S[\boldsymbol{\xi}^\mu, \mathbf{g}(\mathbf{x})])} \cdot \sum_{\mu=1}^{K} \left(\left. \frac{dF(s)}{ds} \right|_{s=S[\boldsymbol{\xi}^\mu, \mathbf{g}(\mathbf{x})]} \cdot \left. \frac{dS(\boldsymbol{\xi}^\mu, \mathbf{z})}{d\mathbf{z}} \right|_{\mathbf{z}=\mathbf{g}(\mathbf{x})} \cdot \frac{d\mathbf{g}(\mathbf{x})}{d\mathbf{x}} \right) \tag{6.6}$$

For example, with $Q(\cdot) = (1/\beta)\log(\cdot)$, $F(\cdot) = \exp(\beta \cdot)$, $S[\boldsymbol{\xi}^\mu, \mathbf{x}] = \langle \boldsymbol{\xi}^\mu, \mathbf{x} \rangle$ and $\mathbf{g}(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|_2$, the energy function and the corresponding update[3] are:

$$E(\mathbf{x}) = -\frac{1}{\beta} \log \sum_{\mu=1}^{K} \exp(\beta \langle \boldsymbol{\xi}^\mu, \mathbf{g}(\mathbf{x}) \rangle), \ \nabla_{\mathbf{x}} E(\mathbf{x}) = -\frac{\sum_{\mu=1}^{K} \exp(\beta \langle \boldsymbol{\xi}^\mu, \mathbf{g}(\mathbf{x}) \rangle) \boldsymbol{\xi}^\mu}{\sum_{\mu=1}^{K} \exp(\beta \langle \boldsymbol{\xi}^\mu, \mathbf{g}(\mathbf{x}) \rangle)} \cdot \frac{d\mathbf{g}(\mathbf{x})}{d\mathbf{x}}. \tag{6.7}$$

This form does not directly admit itself to a distributed storage of memories as in Equation (6.2), and thus, in order to perform the gradient descent on the energy, it is necessary to keep all the memories in their original form. We will try to address this issue by taking inspiration from the area of *kernel machines* [189].

6.2.1 Random Features for Kernel Machines

The expressivity of kernel learning usually comes with increased computational complexity both during training and inference, taking time quadratic and linear in the size of the training set respectively. The groundbreaking work of Rahimi and Recht [54] introduced random features to generate explicit feature maps $\boldsymbol{\varphi} : \mathbb{R}^D \to \mathbb{R}^Y$ for the RBF and other *shift-invariant* kernels[4] that approximate the true kernel function – that is $\langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\varphi}(\mathbf{x}') \rangle \approx$

[3] We are eliding the $d\mathbf{g}(\mathbf{x})/d\mathbf{x} = (1/\|\mathbf{x}\|_2)[I_D - (1/\|\mathbf{x}\|_2^{3/2})\mathbf{x}\mathbf{x}^\top]$ term for the ease of exposition.

[4] Kernel functions that only depend on $(\mathbf{x} - \mathbf{x}')$ and not individually on $\mathbf{x}$ and $\mathbf{x}'$.

$k(\mathbf{x}, \mathbf{x}')$. Various such random maps have been developed for shift-invariant kernels [55, 179, 190] and polynomials kernels [191, 192, 193].

For the RBF kernel and the exponentiated dot-product or EDP kernel $k(\mathbf{x}, \mathbf{x}') = \exp(\langle \mathbf{x}, \mathbf{x}' \rangle)$, there are usually two classes of random features – trigonometric features and exponential features. For the RBF kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|_2^2/2)$, the trigonometric features [54] are given on the left and the exponential features [55] are on the right:

$$\boldsymbol{\varphi}(\mathbf{x}) = \frac{1}{\sqrt{Y}} \begin{bmatrix} \cos(\langle \boldsymbol{\omega}^1, \mathbf{x} \rangle) \\ \sin(\langle \boldsymbol{\omega}^1, \mathbf{x} \rangle) \\ \dots, \\ \cos(\langle \boldsymbol{\omega}^Y, \mathbf{x} \rangle) \\ \sin(\langle \boldsymbol{\omega}^Y, \mathbf{x} \rangle) \end{bmatrix}, \quad \boldsymbol{\varphi}(\mathbf{x}) = \frac{\exp(-\|\mathbf{x}\|_2^2)}{\sqrt{2Y}} \begin{bmatrix} \exp(+\langle \boldsymbol{\omega}^1, \mathbf{x} \rangle) \\ \exp(-\langle \boldsymbol{\omega}^1, \mathbf{x} \rangle) \\ \dots, \\ \exp(+\langle \boldsymbol{\omega}^Y, \mathbf{x} \rangle) \\ \exp(-\langle \boldsymbol{\omega}^Y, \mathbf{x} \rangle) \end{bmatrix}, \tag{6.8}$$

where $\boldsymbol{\omega}^\alpha \sim \mathcal{N}(0, I_D) \forall \alpha \in \{1, \dots, Y\}$ are the random projection vectors.[5] A random feature map $\boldsymbol{\varphi}$ for the RBF kernel can be used for the EDP kernel by scaling $\boldsymbol{\varphi}(\mathbf{x})$ with $\exp(\|\mathbf{x}\|_2^2/2)$. While the trigonometric features ensure that $k(\mathbf{x}, \mathbf{x}) = \langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\varphi}(\mathbf{x}) \rangle = 1$, the exponential features ensure that $\boldsymbol{\varphi}(\mathbf{x}) \in \mathbb{R}_+^{2Y}$, which is essential in certain applications as in transformers [55, 179]. Furthermore, while the random samples $\boldsymbol{\omega}^\alpha \sim \mathcal{N}(0, I_D)$ are supposed to be independent, Choromanski et al. [194] show that the $\{\boldsymbol{\omega}^1, \dots, \boldsymbol{\omega}^Y\}$ can be entangled to be exactly orthogonal to further reduce the variance of the approximation while maintaining unbiasedness. In general, the approximation of the random feature map is $O(\sqrt{D/Y})$, implying that a feature space with $Y \sim O(D/\epsilon^2)$ random features will ensure, with high probability, for any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^D$, $|k(\mathbf{x}, \mathbf{x}') - \langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\varphi}(\mathbf{x}') \rangle| \leq \epsilon$. Scaling in the kernel functions such as $\exp(-\beta\|\mathbf{x} - \mathbf{x}'\|_2^2/2)$ or $\exp(\beta \langle \mathbf{x}, \mathbf{x}' \rangle)$ can be handled with the aforementioned random feature maps $\boldsymbol{\varphi}$ by applying them to $\sqrt{\beta}\mathbf{x}$ with $\langle \boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}), \boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}') \rangle \approx \exp(-\beta\|\mathbf{x} - \mathbf{x}'\|_2^2/2)$.

[5] A technical detail here is that while we are using Y random samples, we are actually developing a $2Y$-dimensional feature map $\boldsymbol{\varphi} : \mathbb{R}^D \to \mathbb{R}^{2Y}$ – we can get a Y dimensional feature map by dropping the $\sin(\cdot)$ terms in the trigonometric features (and add a random rotation term $b^\alpha, \alpha \in [\![Y]\!]$ to the $\cos(\langle \boldsymbol{\omega}^\alpha, \mathbf{x} \rangle + b^\alpha)$ term), and the $\exp(-\cdot)$ term in the exponential features. This modification (using $2Y$ features instead of Y) reduces the variance of the kernel function approximation [55, Lemma 1, 2].

6.3 DRDAM with Random Features

Revisiting the general energy function in Equation (6.3), if we have available an explicit mapping $\boldsymbol{\varphi} : \mathbb{R}^D \to \mathbb{R}^Y$ such that $\langle \boldsymbol{\varphi}(\boldsymbol{\xi}^\mu), \boldsymbol{\varphi}(\mathbf{x}) \rangle \approx F(S[\boldsymbol{\xi}^\mu, \mathbf{x}])$, then we can simplify the general energy function in Equation (6.3) to

$$E(\mathbf{x}) \approx \hat{E}(\mathbf{x}) = -Q\left(\sum_{\mu=1}^{K} \langle \boldsymbol{\varphi}(\boldsymbol{\xi}^\mu), \boldsymbol{\varphi}(\mathbf{g}(\mathbf{x})) \rangle\right) = -Q\left(\left\langle \sum_{\mu=1}^{K} \boldsymbol{\varphi}(\boldsymbol{\xi}^\mu), \boldsymbol{\varphi}(\mathbf{g}(\mathbf{x})) \right\rangle\right). \tag{6.9}$$

Denoting $\mathbf{T} = \sum_\mu \boldsymbol{\varphi}(\boldsymbol{\xi}^\mu)$, we can write a simplified general update step for any input $\mathbf{x}$ as:

$$\nabla_{\mathbf{x}} \hat{E} = -\left.\frac{dQ(s)}{ds}\right|_{s=\langle \boldsymbol{\varphi}(\mathbf{g}(\mathbf{x})), \mathbf{T} \rangle} \cdot \left(\left.\frac{\boldsymbol{\varphi}(\mathbf{z})}{d\mathbf{z}}\right|_{\mathbf{z}=\mathbf{g}(\mathbf{x})}^{\top} \mathbf{T}\right) \cdot \frac{d\mathbf{g}(\mathbf{x})}{d\mathbf{x}} \tag{6.10}$$

where $d\boldsymbol{\varphi}(\mathbf{x})/d\mathbf{x} \in \mathbb{R}^{Y \times D}$ is the gradient of the feature map with respect to its input. In the presence of such an explicit map $\boldsymbol{\varphi}$, we can distribute the memory in a MRDAM into the single Y-dimensional vector $\mathbf{T}$, and be able to apply the update in Equation (6.10). We can then use the random feature based energy gradient $\nabla_{\mathbf{x}} \hat{E}(\mathbf{x})$ instead of the true energy gradient $\nabla_{\mathbf{x}} E(\mathbf{x})$ in the energy gradient descent step in Equation (6.5).[6]

We name this scheme "**D**istributed **r**epresentation for **D**ense **A**ssociative **M**emory" or DRDAM, and we compare the computational costs of DRDAM with the "**M**emory **r**epresentation of **D**ense **A**ssociative **M**emory" or MRDAM in the following:

Proposition 6.1. *With access to the K memories $\{\boldsymbol{\xi}^\mu \in \mathbb{R}^D, \mu \in [\![K]\!]\}$,* MRDAM *takes $O(LKD)$ time and $O(KD)$ peak memory for L energy gradient descent steps (or layers) as defined in Equation (6.5) with the true energy gradient $\nabla_{\mathbf{x}} E(\mathbf{x})$.*

Naively, the random feature based DRDAM would require $O(DY)$ memory to store the random vectors and the $\nabla_{\mathbf{x}} \boldsymbol{\varphi}(\mathbf{x})$ matrix. However, we can generate the random vectors on demand to reduce the overall peak memory to $O(Y)$. Algorithm 1 summarizes the three operations needed by DRDAM: RF recreates the same random feature map from a seed, ProcMems compresses all stored patterns into a single vector $\mathbf{T} \in \mathbb{R}^Y$, and GradComp evaluates the approximate energy gradient $\nabla_{\mathbf{x}} \hat{E}$ used in energy descent.

[6] If $Q(\cdot) = \log(\cdot)$ in Equation (6.9), note that the inner product between unconstrained choices of $\boldsymbol{\varphi}$ can be negative but the argument to $\log$ must not be; thus, we clip the value to the $\log$ to some small $\varepsilon > 0$.

Algorithm 1: Procedures for DRDAM with random features.

Input: Feature dimension Y, random seed τ, stored patterns $\{\boldsymbol{\xi}^\mu\}_{\mu=1}^K$, input query $\mathbf{x}$, feature map $\boldsymbol{\varphi}$, input transform $\mathbf{g}$, and scalar energy nonlinearity Q.

Output: Distributed memory vector $\mathbf{T} = \sum_{\mu=1}^K \boldsymbol{\varphi}(\boldsymbol{\xi}^\mu)$ and approx. gradient $\nabla_{\mathbf{x}} \hat{E}(\mathbf{x})$ from Equation (6.10).

RF(seed τ, pattern $\mathbf{u}$)
- Create empty feature vector $\mathbf{p} \leftarrow 0_Y$ for pattern $\mathbf{u}$
- Reset random number generator to seed τ
- **for** *feature index* $\alpha = 1, \ldots, Y$ **do**
  - Recreate the αth random feature function $\varphi_\alpha : \mathbb{R}^D \to \mathbb{R}$
  - Store its value on the pattern, $p_\alpha \leftarrow \varphi_\alpha(\mathbf{u})$
- **end**
- **return** $\mathbf{p}$

end

ProcMems(seed τ, stored patterns $\{\boldsymbol{\xi}^\mu\}_{\mu=1}^K$)
- Initialize distributed memory vector $\mathbf{T} \leftarrow 0_Y$
- **for** *stored pattern* $\boldsymbol{\xi}^\mu$, $\mu = 1, \ldots, K$ **do**
  - Encode the pattern with RF and add it to memory, $\mathbf{T} \leftarrow \mathbf{T} + \mathrm{RF}(\tau, \boldsymbol{\xi}^\mu)$
- **end**
- **return** $\mathbf{T}$

end

GradComp(seed τ, distributed memory $\mathbf{T}$, input query $\mathbf{x}$)
- Project query $\mathbf{x}$ into feature space, $\mathbf{p} \leftarrow \mathrm{RF}(\tau, \mathbf{g}(\mathbf{x}))$
- Compute energy argument $s \leftarrow \langle \mathbf{T}, \mathbf{p} \rangle$
- Initialize $\mathbf{z} \leftarrow 0_D$ to hold gradient w.r.t. transformed query $\mathbf{y} = \mathbf{g}(\mathbf{x})$
- **for** *coordinate* $i = 1, \ldots, D$ *of* $\mathbf{y}$ **do**
  - Recreate derivative of all features along this coordinate, $\mathbf{u} \leftarrow d\boldsymbol{\varphi}(\mathbf{y})/dy_i|_{\mathbf{y}=\mathbf{g}(\mathbf{x})}$
  - Accumulate the memory-weighted feature derivative, $z_i \leftarrow \langle \mathbf{u}, \mathbf{T} \rangle$
- **end**
- Initialize $\mathbf{z}' \leftarrow 0_D$ to hold gradient w.r.t. the original query $\mathbf{x}$
- **for** *coordinate* $i = 1, \ldots, D$ *of* $\mathbf{x}$ **do**
  - Compute how the transform $\mathbf{g}$ changes with coordinate x_i, $\mathbf{j}_i \leftarrow d\mathbf{g}(\mathbf{x})/dx_i$
  - Apply the chain rule through $\mathbf{g}$, $z'_i \leftarrow \langle \mathbf{j}_i, \mathbf{z} \rangle$
- **end**
- Compute the scalar energy factor $q \leftarrow -Q'(s)$
- **return** $q\mathbf{z}'$

end

The following are the computational complexities of these procedures:

Proposition 6.2. *RF in Algorithm 1 takes $O(DY)$ time and $O(D + Y)$ peak memory.*

Proposition 6.3. *ProcMems in Algorithm 1 takes $O(DYK)$ time and $O(D+Y)$ peak memory.*

Proposition 6.4. *GradComp in Algorithm 1 takes $O(D(Y+D))$ time and $O(D+Y)$ peak memory.*

Combining one ProcMems pass with L GradComp calls gives the end-to-end cost of running DrDAM energy descent.

Theorem 6.1. *With a random feature map $\boldsymbol{\varphi}$ utilizing Y random projections $\{\varphi_\alpha, \alpha \in \{1,\dots,Y\}\}$ and K memories $\{\boldsymbol{\xi}^\mu \in \mathbb{R}^D, \mu \in \{1,\dots,K\}\}$, the random-feature based* DrDAM *takes $O\left(D\left(YK + L(Y+D)\right)\right)$ time and $O(Y+D)$ peak memory for L energy gradient descent steps (or layers) as defined in Equation (6.5) with the random feature based approximation gradient $\nabla_{\mathbf{x}}\hat{E}(\mathbf{x})$ defined in Equation (6.10).*

The proof and computational comparison are provided in Appendix C.6.1.

The memory encoding only needs to be done once, so the same $\mathbf{T}$ can be reused across L energy-gradient steps and across multiple inputs. In this setting, the cost of ProcMems is amortized over those inputs. Adding a new memory is also independent of the number of patterns already stored:

Proposition 6.5. *The inclusion of a new memory $\boldsymbol{\xi} \in \mathbb{R}^D$ to a* DrDAM *with K memories distributed in $\mathbf{T} \in \mathbb{R}^Y$ takes $O(DY)$ time and $O(D+Y)$ peak memory.*

The above result shows that inclusion of new memories correspond to constant time and memory irrespective of the number of memories in the current DenseAM. Next, we study the divergence between the output of a L-layered MrDAM using the energy descent in Equation (6.5) with the true gradient in Equation (6.6) and that of DrDAM using the random feature based gradient in Equation (6.10).

Theorem 6.2. *Consider the following energy function with K memories $\{\boldsymbol{\xi}^\mu \in \mathbb{R}^D, \mu \in \{1,\dots,K\}\}$ and inverse temperature $\beta > 0$:*

$$E(\mathbf{x}) = -\frac{1}{\beta}\log\left(\sum_{\mu=1}^{K}\exp(-\beta/2\|\boldsymbol{\xi}^\mu - \mathbf{x}\|_2^2)\right). \tag{6.11}$$

We further make the following assumptions: (A1) All memories $\boldsymbol{\xi}^\mu$ and inputs $\mathbf{x}$ lie in $\mathcal{X} = [0, 1/\sqrt{D}]^D$. (A2) Using a random feature map $\boldsymbol{\varphi} : \mathbb{R}^D \to \mathbb{R}^Y$ with Y random features, for any

$\mathbf{x}, \mathbf{x}' \in \mathbb{R}^D$ *there is a constant* $C_1 > 0$ *such that* $|\exp(-\beta\|\mathbf{x} - \mathbf{x}'\|_2^2/2) - \langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\varphi}(\mathbf{x}')\rangle| \leq C_1\sqrt{D/Y}$. *Given an input* $\mathbf{x} \in \mathcal{X}$, *let* $\mathbf{x}^{(L)}$ *be the output of the* MRDAM *defined by the energy function in Equation (6.11) using the true energy gradient in Equation (6.6) and* $\hat{\mathbf{x}}^{(L)}$ *be the output of* DRDAM *with approximate gradient in Equation (6.10) using the random feature map* $\boldsymbol{\varphi}$ *with a constant step-size of* $\eta > 0$ *in Equation (6.5). Then*

$$\left\|\mathbf{x}^{(L)} - \hat{\mathbf{x}}^{(L)}\right\|_2 \leq 2\eta L C_1 K e^{\beta E(\mathbf{x})}\sqrt{D/Y}\left(\frac{1 - \left(\eta L(1 + 2K\beta e^{\beta/2})\right)^L}{1 - \eta L(1 + 2K\beta e^{\beta/2})}\right) \tag{6.12}$$

Assumption (A1) just ensures that all the memories and inputs have bounded norm, and can be achieved via translating and scaling the memories and inputs. Assumption (A2) pertains to the approximation introduced in the kernel function evaluation with the random feature map, and is satisfied (with high probability) based on results such as Rahimi and Recht [54, Claim 1] and Choromanski et al. [55, Theorem 4]. The above result precisely characterizes the effect on the divergence $\|\mathbf{x}^{(L)} - \hat{\mathbf{x}}^{(L)}\|$ of the (i) initial energy of the input $E(\mathbf{x})$ – lower is better, (ii) the inverse temperature β – lower is better, (iii) the number of memories K – lower is better, (iv) the ambient data dimensionality D – lower is better, (v) the number of random features Y – higher is better, and (vi) the number of layers L – lower is better. The proof and further discussion are provided in Appendix C.6.2. Note that Theorem 6.2 analyzes the discretized system, but as the step-size $\eta \to 0$, we approach the fully contracting continuous model. An appropriate choice for the energy descent step-size η simplifies the above result, bounding the divergence to $O(\sqrt{D/Y})$:

Corollary 6.1. *Under the conditions and definitions of Theorem 6.2, if we set the step size* $\eta = \frac{C_2}{L(1+2K\beta e^{\beta/2})}$ *with* $C_2 < 1$, *the divergence is bounded as:*

$$\left\|\mathbf{x}^{(L)} - \hat{\mathbf{x}}^{(L)}\right\|_2 \leq \frac{C_1 C_2 e^{\beta(E(\mathbf{x}) - 1/2)}}{\beta(1 - C_2)}\sqrt{D/Y}. \tag{6.13}$$

The same proof technique extends these results to the EDP-based energy function

$$E(\mathbf{x}) = -\frac{1}{\beta}\log\sum_{\mu}\exp(\beta\langle\boldsymbol{\xi}^{\mu}, \mathbf{x}\rangle) + \frac{1}{2}\|\mathbf{x}\|_2^2. \tag{6.14}$$

6.4 Empirical evaluation

To be an accurate approximation of the traditional MRDAM, DRDAM must empirically satisfy the following desiderata for all possible queries and at all configurations for inverse temperature β and pattern dimension D:

(D1) for the same query, DRDAM must predict similar energies and energy gradients as MRDAM; and

(D2) for the same initial query, DRDAM must retrieve similar fixed points as MRDAM.

However, in our experiments we observed that the approximation quality of DRDAM is strongly affected by the choice of β and that the approximation quality decreases the further the query patterns are from the stored memory patterns, as predicted by Theorem 6.2. This regime makes DRDAM especially natural for retrieval and inpainting, where queries are expected to retain enough structure to remain near the stored pattern manifold, rather than for unconstrained generation from arbitrary initial states. We characterize this behavior in the following experiments using the trigonometric "SinCos" basis function, which performed best in our ablation experiments (see Appendix C.3), but note that the choice of the random features do play a significant role in the interpretations of these results.

6.4.1 (D1) How accurate are the energies and gradients of DRDAM?

Figure 6.3 evaluates how well DRDAM, configured at different feature sizes Y, approximates the energy and energy gradients of MRDAM configured with different inverse temperatures β and storing random binary patterns of dimension D. The experimental setup is described below.

We generated $2K = 1000$ unique, binary patterns (where each value is normalized to be in $\{0, \frac{1}{\sqrt{D}}\}$) and stored $K = 500$ of them into the memory matrix Ξ of MRDAM. We denote these stored patterns as $\boldsymbol{\xi}^\mu \in \{0, \frac{1}{\sqrt{D}}\}^D$, $\mu \in [\![K]\!]$, where D is a hyperparameter controlled by the experiment. For a given β, the memory matrix is converted into the featurized memory vector $T_\alpha := \sum_\mu \varphi_\alpha(\boldsymbol{\xi}^\mu)$ from Equation (6.9), where $\alpha \in [\![2Y]\!]$. The remaining patterns are treated as the "random queries" $\mathbf{x}^b_{\text{far}}$, $b \in [\![K]\!]$ (i.e., queries that are far from the stored patterns). Finally, in addition to evaluating the energy at these random queries and at the stored patterns, we also want to evaluate the energy at queries $\mathbf{x}^b_{\text{near}}$ that

are "near" the stored patterns; thus, we take each stored pattern $\boldsymbol{\xi}^\mu$ and perform bit-flips on $0.1D$ of its entries.

For each set of queries $\mathbf{x}^b \in \{\boldsymbol{\xi}^b, \mathbf{x}^b_{\text{near}}, \mathbf{x}^b_{\text{far}}\}$, $b \in [\![K]\!]$, and choice of β, Y, and D, we compute the **M**ean **A**pproximation **E**rror (MAE) between MRDAM's energy $E_b := E(\mathbf{x}^b; \beta, \Xi)$ (whose gradient matrix is denoted $\nabla_{\mathbf{x}} E_b$) and DRDAM's energy $\hat{E}_b := \hat{E}(\mathbf{x}^b; \beta, \mathbf{T})$ (whose gradient is denoted $\nabla_{\mathbf{x}} \hat{E}_b$).

$$\text{MAE}_{\text{Energy}} = \frac{1}{K} \sum_{b \in [\![K]\!]} \left| E_b - \hat{E}_b \right|, \text{ and } \quad \text{MAE}_{\text{Gradient}} = \frac{1}{K} \sum_{b \in [\![K]\!]} \left\| \nabla_{\mathbf{x}} E_b - \nabla_{\mathbf{x}} \hat{E}_b \right\|_2 \tag{6.15}$$

We found it useful to visualize the results using log-scale and to compare the errors against the expected error of a "random guess" of the energy/gradients (horizontal red dashed line in each plot of Figure 6.3). The "random guess error" was empirically computed by sampling a new set of random queries $\mathbf{x}^b_{\text{guess}}$, $b \in [\![K]\!]$ (independent of the reference queries) and computing the MAE between the standard energy on the reference queries vs. the approximate energies on the random queries. This error was averaged across Y for each β; the highest average error across all βs is plotted.

Observation 1: DRDAM approximations are best for queries near stored patterns DRDAM approximations for both the energy and energy gradients are better the closer the query patterns are to the stored patterns. In this regime, approximation accuracy predictably improves when increasing the value for Y within "reasonable" values (i.e., values corresponding into sizes of featurized queries and memories that can operate within 46GB of GPU memory).

Observation 2: DRDAM approximations worsen as inverse temperature β increases Across nearly all experiments, DRDAM approximations worsen as β increases. At queries near the stored patterns, $\beta = 50$ has an energy error approximately $10\times$ that of $\beta = 30$ and $100\times$ that of $\beta = 10$ across all Y. At high D and when queries are far from the patterns, the error of $\beta = 50$ approaches $1000\times$ the error of $\beta = 10$. This observation similarly holds for the errors of corresponding gradients, corroborating the statement of Theorem 6.2.

Observation 3: DRDAM approximations break at sufficiently high values of D and β In general, DRDAM's approximation errors remain the same across choices for D, especially when the queries are near the stored patterns. However, when both β and D

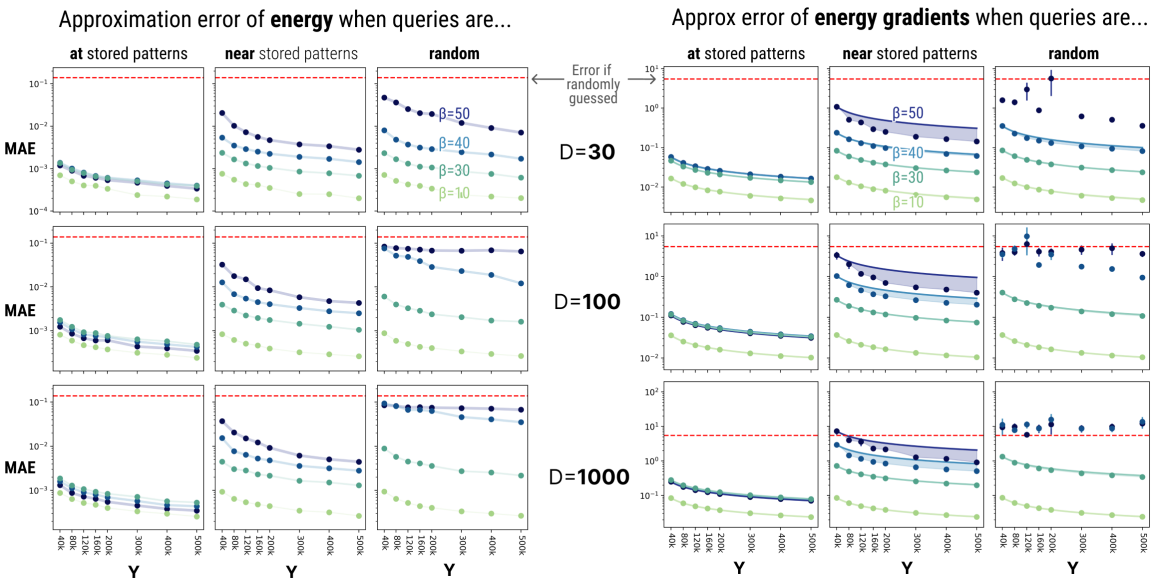

Figure 6.3: DRDAM produces better approximations to the energies and gradients of MRDAM when the queries are closer to the stored patterns. Approximation quality improves with larger feature dimension Y, but decreases with higher β and higher pattern dimension D. Approximation error is computed on 500 stored binary patterns normalized between $\{0, \frac{1}{\sqrt{D}}\}$. The **M**ean **A**pproximation **E**rrors (**MAE**, Equation (6.15)) is taken over 500 queries initialized: **at** stored patterns (i.e., queries equal the stored patterns), **near** stored patterns (i.e., queries equal the stored patterns where 10% of the bits have been flipped), and **randomly** (i.e., queries are random and **far** from stored patterns). Error bars represent the standard error of the mean but are visible only at poor approximations. Red horizontal lines represent the expected error of random energies and gradients. The theoretical error upper bounds of Equation (6.13) (dark curves on the gradient errors in the *right plot only*) show a tight fit to empirical results at low β and D and are only shown if predictions are "better than random". The shaded area shows the difference between the theoretical bound and the empirical results.

are sufficiently large (e.g., $\beta \geq 40$ and $D \geq 100$ in Figure 6.3), increasing the value of Y does not improve the approximation quality: DRDAM continues to return almost random gradients and energies. We explore this phenomenon more in Subsection 6.4.2 in the context of the retrievability of stored patterns.

6.4.2 (D2) How accurate are the memory retrievals using DRDAM?

Memory retrieval is the process by which an initial query $\mathbf{x}^{(0)}$ descends the energy function and is transformed into a fixed point of the energy dynamics. This process can be described by the discrete update rule in Equation (6.5), where E can represent either MRDAM's energy or the approximate energy of DRDAM. A memory is said to be "retrieved" when $|E(\mathbf{x}^{(L)}) - E(\mathbf{x}^{(L-1)})| < \varepsilon$ for some small $\varepsilon > 0$, at which point $\mathbf{x}^{(L-1)} \approx \mathbf{x}^{(L)} =: \mathbf{x}^\star$ is declared to be the retrieved *memory* after L iterations because $\mathbf{x}^\star$ lives at a local minimum

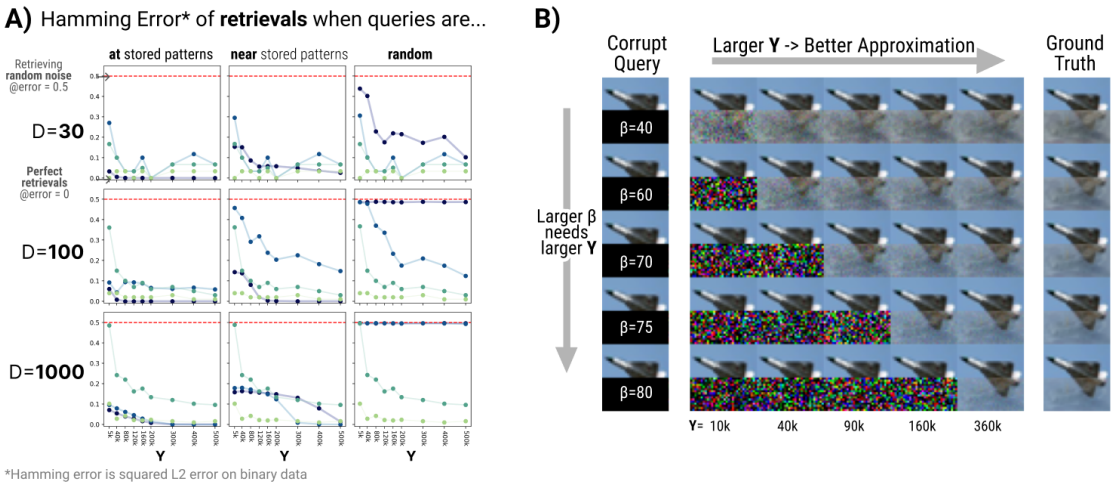

Figure 6.4: **A)** Retrieval errors predictably follow the approximation quality of Figure 6.3. Error is lowest at/near stored patterns but is completely random when energy and gradient approximations are poor, i.e., at high values of β and D. Note that error improves across Y but follows a different (and noisier) trace than the corresponding approximations for energy and gradient in Figure 6.3 due to error accumulating over multiple update steps. **B)** DrDAM's approximation quality improves as Y increases (visible at low β), but larger Y's are needed for good approximations to the DAM's fixed points at higher β's. (Left) The same corrupted query from CIFAR-10 where bottom 50% is masked is presented to DAM's with different β's. (Middle) The fixed points of DrDAM for each β at different sizes Y of the feature space. (Right) The "ground truth" fixed point of MrDAM. The top 50% of pixels are clamped throughout the dynamics.

of the energy function E.

Quantifying retrieval error Given the same initial queries $\mathbf{x}^{(0)} \in \{0, \frac{1}{\sqrt{D}}\}^D$, we want to quantify the difference between the fixed points $\hat{\mathbf{x}}^\star$ retrieved by descending DrDAM's approximate energy and the fixed points $\mathbf{x}^\star$ retrieved by descending the energy of MrDAM. We follow the experimental setup of Subsection 6.4.1, only this time we run full memory retrieval dynamics until convergence.

Note that since energy uses an L2-similarity kernel, memory retrieval is not guaranteed to return binary values. Thus, we binarize $\mathbf{x}^\star$ by assigning each entry to its nearest binary value before computing the normalized Hamming approximation error Δ_H, i.e.,

$$\lceil x \rfloor := \begin{cases} \frac{1}{\sqrt{D}}, & x \geq \frac{1}{2\sqrt{D}} \\ 0, & \text{otherwise} \end{cases}, \text{and} \qquad \Delta_H := \frac{1}{\sqrt{D}} \sum_{i \in [\![D]\!]} \left| \lceil \mathbf{x}_i^\star \rfloor - \lceil \hat{\mathbf{x}}_i^\star \rfloor \right|. \tag{6.16}$$

The choice of normalized Hamming approximation error Δ_H on our binary data is

equivalent to the squared L2 error on the left side of our bound in Equation (6.13) (up to a linear scaling of $\frac{1}{\sqrt{D}}$).

Figure 6.4A shows the results of this experiment. Many observations from Subsection 6.4.1 translate to these experiments: we notice that retrieval is random at high β and D, and that retrievals are of generally higher accuracy nearer the stored patterns. However, we notice that high β values can retrieve better approximations than lower values of β when the queries are at or near stored patterns. Additionally, for sufficiently high β (e.g., see $D = 1000$, $\beta = 50$ near stored patterns), this accompanies an interesting "thresholding" behavior for Y where retrieval error starts to improve rapidly once Y reaches a minimal threshold. This behavior is corroborated in the high D regime in Figure 6.4B.

Visualizing retrieval error Figure 6.4B shows what retrieval errors look like qualitatively. We stored $K = 10$ random images from CIFAR10 [195] into the memory matrix of MRDAM, resulting in patterns of size $D = 3 \times 32 \times 32 = 3072$, and compared retrievals using βs that produced meaningful image results with MRDAM. To keep β values consistent with our previous experiments, each pixel was normalized to the continuous range between 0 and $\frac{1}{\sqrt{D}}$ s.t. $\xi_i^\mu \in [0, \frac{1}{\sqrt{D}}]$, with $\mu \in [\![K]\!]$ and $i \in [\![D]\!]$.

From Subsection 6.4.1 and Figure 6.4A, we know that approximate retrievals are inaccurate at high β and high D if the query is far from the stored patterns. However, this is exactly the regime we test when retrieving images in Figure 6.4B. The visible pixels (top half of the image) are clamped while running the dynamics until convergence. Retrieved memories at different configurations for DRDAM are plotted against their corresponding MRDAM retrievals in Figure 6.4B.

As β increases, insufficiently large values of Y fail to retrieve meaningful approximations to the dynamics of MRDAM. We observe that image completions generally become less noisy as Y increases, but with diminishing improvement in perceptible quality after some threshold where DRDAM goes from predicting noise to predicting meaningful image completions.

6.5 Conclusion

Our study is explicitly designed to characterize where DRDAM is a good approximation to the energies and dynamics of MRDAM. In pushing the limits of the distributed representation, we discovered that DRDAM is most accurate when: (1) query patterns are nearer to

the stored patterns; (2) β is lower; and (3) Y is large. Error bounds for these situations are explicitly derived in Theorem 6.2 and empirically tested in § 6.4.

We have explored the use of distributed representations via random feature maps in DenseAMs. We have demonstrated how this can be done efficiently, and we precisely characterized how it performs the neural dynamics relative to the memory representation DenseAMs. Our theoretical results highlight the factors playing a role in the approximation introduced by the distributed representations, and our experiments validate these theoretical insights. As future work, these distributed memories should be studied as reusable modules inside hierarchical AMs and HAMUX-style systems [77, 17], where compression must coexist with inductive biases such as convolutions, attention, and multiple hidden layers.

CHAPTER 7
DENSE ASSOCIATIVE MEMORY WITH EPANECHNIKOV ENERGY

We propose a novel energy function for Dense Associative Memory (DenseAM) networks, the Log-Sum-ReLU (LSR), inspired by optimal kernel density estimation. Unlike the common log-sum-exponential (LSE) function, LSR is based on the Epanechnikov kernel and enables exact memory retrieval with exponential capacity without requiring exponential separation functions. Moreover, it introduces abundant additional *emergent* local minima while preserving perfect pattern recovery — a characteristic previously unseen in DenseAM literature. Empirical results show that LSR energy has significantly more local minima (memories) that have comparable log-likelihood to LSE-based models. Analysis of LSR's emergent memories on image datasets reveals a degree of creativity and novelty, hinting at this method's potential for both large-scale memory storage and generative tasks.

7.1 Associative Memories and Energy Landscapes

Energy-based associative memory networks or AMs are models parameterized with M "memories" in d dimensions, $\boldsymbol{\Xi} = \{\boldsymbol{\xi}_\mu \in \mathbb{R}^d, \mu \in [\![M]\!]\}$. A popular class of models from this family can be described by an energy function defined on the state vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$:

$$E_\beta(\mathbf{x}; \boldsymbol{\Xi}) = -Q\left[\sum_{\mu=1}^{M} F\left(\beta S\left(g(\mathbf{x}), \boldsymbol{\xi}_\mu\right)\right)\right], \tag{7.1}$$

where $g : \mathbb{R}^d \to \mathbb{R}^d$ is a vector operation (such as binarization, (layer) normalization), $S : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a similarity function (e.g., dot-product, negative Euclidean distance), $\beta > 0$ denotes the inverse temperature, $F : \mathbb{R} \to \mathbb{R}$ is a rapidly growing separation function (power, exponential) and Q is a monotonic scaling function (logarithm, linear) [26, 70, 14]. With g as the sign-function, $\boldsymbol{\xi}_\mu \in \{-1, +1\}^d$, $S(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ and F as the quadratic function, and Q as a linear function, we recover the classical Hopfield model [9]. The output of an AM corresponds to one of the local minima of its energy function. A memory $\boldsymbol{\xi}_\mu$ is said to be retrieved if $\mathbf{x} \approx \boldsymbol{\xi}_\mu$ corresponds to such a local minimum, with exact retrieval occurring when $\mathbf{x} = \boldsymbol{\xi}_\mu$. The memory capacity of the AM is defined as the maximum number $M^\star$ of correctly retrieved memories. For classical AMs, the capacity scales as

$M^\star = O(d)$. With the introduction of power-law separation functions — that is, $F(x) = x^p$ for $p > 2$ — modern *Dense Associative Memories* (DenseAMs) achieve a significantly higher capacity of $M^\star = O(d^p)$ [26, 196].

The use of an exponential separation function combined with a logarithmic scaling function — $F(x) = \exp(x)$ and $Q(x) = \log x$ — leads to the widely studied log-sum-exp (LSE) energy function [27, 28, 75], yielding exponential memory capacity $M^\star \sim \exp(d)$ [197]. Hierarchical organization of memories has also been explored in Krotov [77] and Hoover et al. [17]. Given that the gradient of the LSE energy corresponds to a softmax over all stored patterns, recent works have proposed sparsified variants to improve scalability. In particular, Hu et al. [57] and Santos et al. [58] consider sparsified softmax-based gradients, effectively projecting the full gradient onto a reduced support.[1] Alternatively, Wu et al. [59] propose learning new representations for the memories to increase capacity, while continuing to use the LSE energy in the transformed representation space.

In this work, we consider the following motivating question: *can we achieve simultaneous perfect memorization and generalization in energy-based associative memory models?* While the exponential separation function enables DenseAMs to achieve high memory capacity, capacity alone is not the only desideratum. In standard supervised learning, it was long believed that exactly interpolating or memorizing the training data — achieving zero training loss — would harm generalization. However, recent advances, particularly in deep learning, have challenged this belief: models that perfectly fit the training data can still generalize well. Although this phenomenon gained prominence with deep networks, it has earlier roots in kernel methods and boosting [198, 199, 200].

An analogous question arises in the context of AMs. Traditionally, AMs focus on storing a fixed set of patterns. But from a broader machine learning perspective, the goal extends beyond memorization to include the generation of new, meaningful patterns. Prior work using LSE-type energy functions has shown that generating such novel patterns typically requires sacrificing perfect recall of the original patterns. This trade-off highlights a core tension between memorization and generalization. To address this, we explore alternative separation functions that can preserve exact memorization while enabling the emergence of new patterns — pushing toward models that truly unify memory and generalization.

Our approach is also motivated by the well-established connection between the energy and probability density function. An energy function $E : \mathbb{R}^d \to \mathbb{R}$ induces a

[1] Sparsified softmax-based gradients can be interpreted as specific projections of the original gradient.

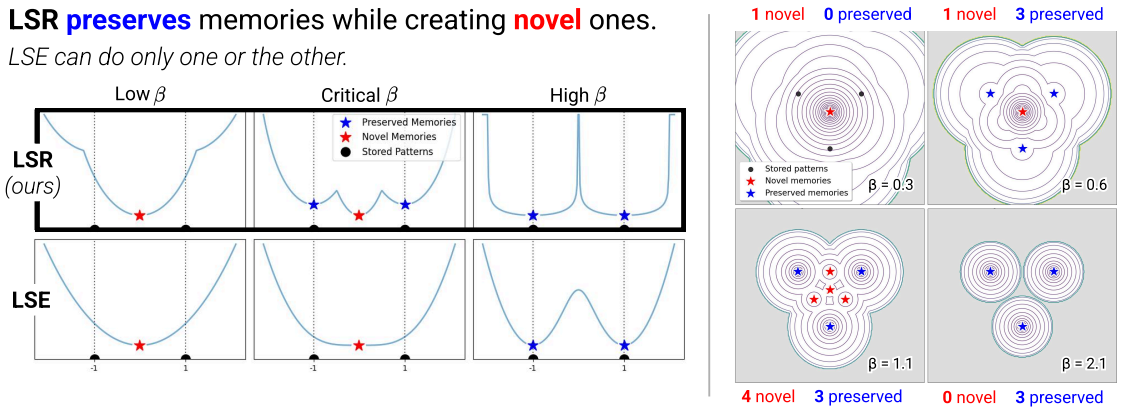

Figure 7.1: LSR energy can create more memories than there are stored patterns under critical regimes of β. Left: 1D LSR vs LSE energy landscape. Note that LSE is never capable of having more local minima than the number of stored patterns. Right: 2D LSR energy landscape, where increasing β creates novel local minima where basins intersect. Unsupported regions are shaded gray.

probability density function $p : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ with $p(\mathbf{x}) = \exp[-E(\mathbf{x})] / \int_{\mathbf{z}} \exp[-E(\mathbf{z})] d\mathbf{z}$. Conversely, given a density p, we have an energy $E(\mathbf{x}) \propto -\log p(\mathbf{x})$, the negative log-likelihood. Minimizing the energy corresponds to maximizing the log-likelihood (with respect to the corresponding density). Based on this connection, with $Q(\cdot) = \log(\cdot)$, the $\exp[-E_\beta(\mathbf{x}; \boldsymbol{\Xi})] = \sum_\mu F(\beta S(\mathbf{x}, \boldsymbol{\xi}_\mu))$ in Equation (7.1) (assuming g is identity) is the corresponding (unnormalized) density at $\mathbf{x}$. Assuming that the memories $\boldsymbol{\xi}_\mu \sim p$ are sampled from an unknown ground truth density p, the $\exp[-E_\beta(\mathbf{x}; \boldsymbol{\Xi})]$ is an unnormalized **kernel density estimate** or KDE of p at $\mathbf{x}$ with the *kernel* F and bandwidth $1/\beta$ [60]. Thus, the LSE energy with $F(x) = \exp(x)$ and $S(\mathbf{x}, \mathbf{x}') = -1/2 \|\mathbf{x} - \mathbf{x}'\|^2$ corresponds to the KDE of p with the Gaussian kernel.

KDE is well studied in nonparametric statistics [60, 201], and various forms of kernels have been explored. The quality of the estimates are well characterized in terms of properties on the kernels; we will elaborate on this in the sequel. While the Gaussian kernel is extremely popular for KDE (much like LSE in AM literature), there are various other kernels which have better estimation abilities than the Gaussian kernel. Among the commonly used kernels, the Epanechnikov kernel has the most favorable estimation quality (see § 7.2). In our notation, this corresponds to a kernel $F(x) = \max(1 + x, 0) = \text{ReLU}(1 + x)$, a shifted ReLU operation (again with $S(\mathbf{x}, \mathbf{x}') = -1/2 \|\mathbf{x} - \mathbf{x}'\|^2$). This results in a novel energy function that we name Log-Sum-ReLU or LSR (see Equation (7.3)). *Surprisingly, we show that this energy function is capable of both exactly memorizing all M original patterns*

(with $M \sim \exp(d)$) and simultaneously generating new, meaningful emergent memories (see Definition 7.2). This defies the conventional tradeoff seen in prior AM models — where improving generalization (ability to create emergent local minima) typically requires compromising exact memorization — and reveals that precise memory storage and creative pattern generation are not inherently at odds with one another. To summarize, we make the following contributions in this work:

- **Novel ReLU-based energy function with exponential memory capacity.** We propose a LSR energy function for DenseAM using the popular ReLU activation, built upon the connection between energy functions and densities. In Theorems 7.1 and 7.2 respectively, we demonstrate exact retrieval and exponential memory capacity of LSR energy, without the use of $\exp(\cdot)$ separation function.

- **Simultaneous memorization and emergence.** We show that this LSR energy has a unique property of *simultaneously* being able to *exactly* retrieve all original memories (training data points) while also creating many *emergent* memories (additional local minima). The total number of memories of LSR can exceed the number of stored patterns, a property absent with LSE (see Proposition 7.1). When applied to images, LSR can generate novel and seemingly creative memories that are not present in the training dataset.

7.2 Kernel Density Estimation and the Choice of Kernels

We now provide a brief overview of Kernel Density Estimation (KDE) considering the univariate setting for simplicity; similar conclusions also hold in higher dimensions. Given a sample $\boldsymbol{\Xi} = \{\xi_\mu \in \mathbb{R}, \mu \in [\![M]\!]\}$ drawn from an unknown density f, the KDE is defined as $\hat{f}_h(\xi) = (Mh)^{-1} \sum_{\mu=1}^{M} K\left(\frac{\xi - \xi_\mu}{h}\right)$, where $K(\cdot)$ is the kernel function and $h > 0$ is the bandwidth parameter. The kernel function is assumed to satisfy: (i) symmetry (i.e., $K(-x) = K(x)$, for all $x \in \mathbb{R}$), (ii) positivity (i.e., $K(x) \geq 0$, for all $x \in \mathbb{R}$) and (iii) normalization (i.e., $\int_x K(x)\, dx = 1$). Note that for the purpose of KDE, the scale of the kernel function is not unique. That is, for a given $K(\cdot)$, we can define $\tilde{K}(\cdot) = b^{-1}K(\cdot/b)$, for some $b > 0$. Then, one obtains the same KDE by rescaling the choice of h. Hence, the shape of the kernel function plays a more important role in determining the choice of the kernel. We now introduce two parameters associated with the kernel, $\mu_K := \int_x x^2 K(x)\, dx$ and

$\sigma_K := \int_x K^2(x)\,dx$ that correspond to the *scale* and *regularity* of the kernel. We will discuss below how the *generalization error* of KDE depends on the aforementioned parameters.

The *generalization error* of $\hat{f}_h(\xi)$ is measured by the Mean Integrated Squared Error (MISE), given by $\text{MISE}(h) = \mathbb{E}\left[\int_\xi (\hat{f}_h(\xi) - f(\xi))^2 d\xi\right]$. Assuming that the ground-truth density $f(\xi)$ is twice continuously differentiable, a second-order Taylor expansion gives the leading terms of the $\text{MISE}(h)$, which decomposes into squared bias and variance terms: $\mathsf{MISE}(h) \approx \frac{\mu_K^2}{4} h^4 \int_\xi |f''(\xi)|^2 d\xi + \frac{\sigma_K}{Mh}$; see Wand and Jones [60, Section 2.5] for details. This result shows that reducing h decreases bias but increases variance, while increasing h smooths the estimate but introduces bias, highlighting the bias-variance trade-off. The optimal mean-square is obtained by minimizing $\text{MISE}(h)$ with respect to h. We thus obtain the optimal choice of h and the optimal generalization accuracy as

$$h_* := \left(\frac{\sigma_K}{M\mu_K^2} \frac{4}{\int_\xi |f''(\xi)|^2 d\xi}\right)^{1/5} \quad \text{and} \quad \mathsf{MISE}(h_*) \approx \frac{5}{4}\left(\frac{\sqrt{\mu_K}\sigma_K \int_\xi |f''(\xi)|^2 d\xi}{M}\right)^{4/5}, \tag{7.2}$$

respectively. From this, we see that the choice of the kernel K in the KDE, controls the generalization error via the term $\sqrt{\mu_K}\sigma_K$.

Thus, a natural question is to find the choice of kernel $K(\cdot)$ that results in the minimum $\mathsf{MISE}(h_*)$. As discussed above, the scale of the kernel function is non-unique. Hence, the problem boils down to minimizing σ_K (which is a regularity parameter of the kernel, determining the shape), subjected to $\mu_K = 1$ (without loss of generality), over the class of normalized, symmetric, and positive kernels. This problem is well-studied (see, for example, [62], [202], Wand and Jones [60, Section 2.7]), and, as it turns out, the Epanechnikov kernel $K_{\mathsf{epan}}(x) = \max\{1 - x^2, 0\} = \text{ReLU}\,(1 - x^2)$ achieves the optimal *generalization error*. The quantity, $\mathsf{Eff}(K) := \sigma_K / \sigma_{K_{\mathsf{epan}}}$ is hence referred to as the efficiency of any kernel with respect to the Epanechnikov kernel. We discuss other compact-support kernels and their emergence behavior in Appendix D.4.

The number of modes of the KDE has also been examined in the literature, mostly when the target is unimodal. Assuming that the target is unimodal, a direct consequence of Mammen [63, Theorem 1] on the number of modes of the KDE when $d = 1$ (also see Geshkovski et al. [64, Theorem 1.1]) is that the number of modes of KDE with a Gaussian kernel with bandwidth h is $\widetilde{\Theta}(1/\sqrt{h})$; see Geshkovski et al. [64, Section 1.2] for extensions to dimension $d > 1$.

7.3 A New Energy Function with Emergent Memory Capabilities

So far, we have seen the relationship between the LSE energy and the KDE, i.e., $\exp[-E_\beta(\mathbf{x}; \boldsymbol{\Xi})]$ is an unnormalized kernel density estimate with the Gaussian kernel and the bandwidth $1/\beta$, and the optimality of using the Epanechnikov kernel in KDE. Given these observations, we will explore the use of the corresponding shifted-ReLU separation function $\mathrm{ReLU}\,(1 + x)$ in the energy function instead of the widely used exponentiation. Before we state the precise energy functions, we compare and contrast the shapes of these separation functions $F(\beta x)$ in Figure 7.2 for varying values of the inverse temperature β. Note that, as the β increases, both these separation functions decay faster. However, as expected, the shifted-ReLU separation linearly decays and then zeroes out.

Recall that LSE ENERGY is given by $E_\beta^{\mathrm{LSE}}(\mathbf{x}; \boldsymbol{\Xi}) = -\frac{1}{\beta} \log \sum_{\mu=1}^{M} \exp(-\frac{\beta}{2} \|\mathbf{x} - \boldsymbol{\xi}_\mu\|^2)$. Based on the discussion on separation functions, our proposed LSR ENERGY (which we also refer to as Epanechnikov energy) is given by

$$E_\beta^{\mathrm{LSR}}(\mathbf{x}; \boldsymbol{\Xi}) = -\frac{1}{\beta} \log \left(\epsilon + \sum_{\mu=1}^{M} \mathrm{ReLU} \left(1 - \frac{\beta}{2} \|\mathbf{x} - \boldsymbol{\xi}_\mu\|^2 \right) \right), \tag{7.3}$$

where $\|\cdot\|$ describes the Euclidean norm and β is an inverse temperature.

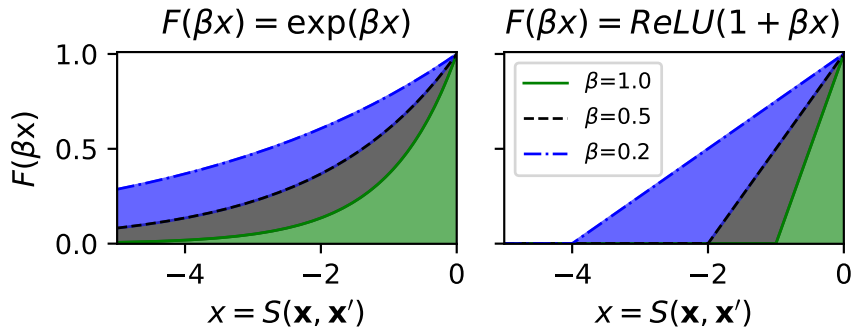

Figure 7.2: Visualizing the separation functions $F(\beta x) = \exp(\beta x)$ (LSE) and $F(\beta x) = \mathrm{ReLU}\,(1 + \beta x)$ (LSR) with $x = S(\mathbf{x}, \mathbf{x}')$ for varying values of β. We focus on $S(\mathbf{x}, \mathbf{x}') = -1/2 \|\mathbf{x} - \mathbf{x}'\|^2$.

The factor $\epsilon \geq 0$ in the LSR energy is a small nonnegative constant, where an $\epsilon > 0$ ensures that every point in the space has finite (albeit extremely large $O(\log(1/\epsilon))$) energy for all values of β. Indeed, with $\epsilon = 0$, defining $\mathsf{S}_\mu \triangleq \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \boldsymbol{\xi}_\mu\| \leq \sqrt{2/\beta}\}$, it is easy to see that $\forall \mathbf{x} \in \mathcal{X} \setminus \cup_{\mu=1}^{M} \mathsf{S}_\mu$, $E_\beta^{\mathrm{LSR}}(\mathbf{x}) = \infty$. This is a result of the finiteness of the

ReLU separation function. Regions of infinite energy imply zero probability density, which matches the finite support of the density estimate with the Epanechnikov kernel. Based on the introduced LSR energy, we next highlight the following favourable properties.

Theorem 7.1. *Let* $r = \min_{\mu,\nu\in[\![M]\!],\mu\neq\nu} \|\boldsymbol{\xi}_\mu - \boldsymbol{\xi}_\nu\|$ *be the minimum Euclidean distance between any two memories. Let* $S_\mu(\Delta) = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \boldsymbol{\xi}_\mu\| \leq \Delta\}$ *be a basin around the* μ^{th} *memory for some basin radius* $\Delta \in (0, r)$*. Then, with* $\beta = 2/(r-\Delta)^2$*, for any* $\mu \in [\![M]\!]$ *and any input* $\mathbf{x} \in S_\mu(\Delta)$*, the output of the DenseAM via LSR energy gradient descent is exactly* $\boldsymbol{\xi}_\mu$*, implying that all memories* $\boldsymbol{\xi}_\mu, \mu \in [\![M]\!]$ *are retrievable. Furthermore, if the learning rate of the energy gradient descent is set appropriately, then for any* $\mu \in [\![M]\!]$ *and any* $\mathbf{x} \in S_\mu(\Delta)$*, the memory is exactly retrieved with a single energy gradient descent step (single step retrieval).*

The above result states that, given a set of memories, and an appropriately selected β, there is a distinct basin of attraction $S_\mu(\Delta)$ around each memory $\boldsymbol{\xi}_\mu$, and any input $\mathbf{x}$ from within that basin exactly retrieves the memory as the output of the DenseAM.

Remark 7.1. *For a finite but appropriately large* β*, the LSR energy gradient* $\nabla E_\beta^{LSR}(\boldsymbol{\xi}_\mu; \boldsymbol{\Xi})$ *at any memory* $\boldsymbol{\xi}_\mu$ *is exactly zero, implying exact retrieval of the memory. The LSE energy gradient* $\nabla E_\beta^{LSE}(\boldsymbol{\xi}_\mu; \boldsymbol{\Xi})$ *is only approximately zero, and the retrieved point is approximately equal to an original memory [28, Theorem 3]. However, if* $\beta = \infty$ *then the LSE energy gradient is exactly zero at the memory.*

The striking phenomenon that we observe with LSR energy is that the DenseAM can simultaneously create local energy minima around the original memories as well as additional local minima around points that are not part of the set of original memories; see Figure 7.1. We formalize this concept below through the notion of *global emergence*.

Definition 7.1 (Novel local minima)**.** *Consider a DenseAM parameterized with* M *memories* $\boldsymbol{\Xi} = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M\}, \boldsymbol{\xi}_\mu \in \mathcal{X}$*, and equipped with an energy function* $E_\beta(\mathbf{x}; \boldsymbol{\Xi})$ *at any state* $\mathbf{x} \in \mathcal{X}$ *for a specific inverse temperature* $\beta > 0$*. For some* $\varepsilon > 0$*, we define* $\mathcal{M}_\varepsilon$ *as the (possibly empty) set of* **novel local minima** $\tilde{\boldsymbol{\xi}} \in \mathcal{X}$ *such that* $\forall \tilde{\boldsymbol{\xi}} \in \mathcal{M}_\varepsilon$*,*

(a) $\tilde{\boldsymbol{\xi}}$ *is a local energy minimum with* $\nabla E_\beta(\tilde{\boldsymbol{\xi}}; \boldsymbol{\Xi}) = 0$ *and* $\nabla^2 E_\beta(\tilde{\boldsymbol{\xi}}; \boldsymbol{\Xi}) \succ 0$*,*

(b) $\tilde{\boldsymbol{\xi}}$ *is novel with respect to the original memories, that is,* $\min_{\mu\in[\![M]\!]} \left\|\tilde{\boldsymbol{\xi}} - \boldsymbol{\xi}_\mu\right\| \geq \varepsilon$*.*

Definition 7.2 (Global emergence)**.** *For the DenseAM in Definition 7.1 and for some $\varepsilon > 0$, let $\mathcal{M}_\varepsilon$ be the (possibly empty) set of novel local minima. For some $\beta \subset (0, \infty)$, we claim that this system exhibits ε-***global emergence** *if (i) each original memory $\boldsymbol{\xi}_\mu, \mu \in [\![M]\!]$ is a local energy minimum with $\nabla E_\beta(\boldsymbol{\xi}_\mu; \boldsymbol{\Xi}) = 0$ and $\nabla^2 E_\beta(\boldsymbol{\xi}_\mu; \boldsymbol{\Xi}) \succ 0$ (positive definite), and (ii) the set $\mathcal{M}_\varepsilon$ is non-empty. We term $\mathcal{M}_\varepsilon$ as the set of ε-***globally emergent memories**.

The notion of ε-global emergence specifically refers to those new patterns that arise *after* all original memories have been exactly stored. Definition 7.2 characterizes emergence as a property of the global energy function at a specific inverse temperature, requiring simultaneous exact recovery of all original memories and the presence of at least one novel local minimum (parameterized with ε). It is instructive to start by understanding the above definition for DenseAMs equipped with LSE energy. According to Ramsauer et al. [28], any point $\mathbf{x}^*$ such that $\nabla E_\beta^{\text{LSE}}(\mathbf{x}^*; \boldsymbol{\Xi}) = 0$ is defined via the softmax corresponding to the transformer attention as follows: $\mathbf{x}^* = \sum_{\mu=1}^{M} \text{softmax}(\beta(\mathbf{x}^*)^\top \boldsymbol{\xi}_\mu)\boldsymbol{\xi}_\mu$, and the softmax can be highly peaked if all $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^M$ are well separated and $\mathbf{x}^*$ is near a stored pattern $\boldsymbol{\xi}_\mu$. If no stored pattern $\boldsymbol{\xi}_\mu$ is well separated from the others, then $\mathbf{x}^*$ is close to a global fixed point, which is the arithmetic mean of all the stored patterns. Based on this, we can make the following observations:

- Case I: *All LSE memories are novel*. With a large enough but finite β, there is a minimum close to each of the original memories. However, each of these local minima will be considered novel local minima as these are distinct from the original memories, thus condition (ii) in Definition 7.2 will be satisfied. However, then the condition (i) in Definition 7.2 would not be satisfied.

- Case II: *No LSE memories are novel*. If we do consider the case $\beta = \infty$, then the original memories would exactly be the local energy minima, and condition (i) will be satisfied. But then the set of novel local minima $\mathcal{M}_\varepsilon$ for a strictly positive ε would be empty, violating condition (ii).

- Case III: *Novel LSE memories form only when basins merge*. For a moderate β, LSE can form novel local minima by merging the basins of attractions of the original memories, thereby giving us a non-empty $\mathcal{M}_\varepsilon$ and satisfying condition (ii) in Definition 7.2. However, condition (i) will be violated as the memories whose basins are merged would no longer be local minima.

Thus, we can make this more formal in the following:

Proposition 7.1. *Assume $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^M$ are i.i.d. from any density fully supported on $\mathcal{X}$. Note that they are linearly independent with probability 1, as otherwise they lie in a lower dimensional space. Then, for any $\beta > 0$, the LSE energy, $E_\beta^{LSE}(\cdot)$, does not satisfy the ε-global emergence in Definition 7.2.*

One can argue that global emergence as in Definition 7.2 is too restrictive; we also want to characterize an individual local minimum as "emergent", or not. Thus we present a relaxed *local* notion of emergence in the following, noting that LSE does not satisfy this weaker form of emergence either:

Definition 7.3 (Locally emergent memory)**.** *Consider the DenseAM in Definition 7.1 with a non-empty set of novel local minima $\mathcal{M}_\varepsilon$ for a $\varepsilon > 0$. For any $\tilde{\boldsymbol{\xi}} \in \mathcal{M}_\varepsilon$, let $\mathcal{S}(\tilde{\boldsymbol{\xi}}) \subseteq \boldsymbol{\Xi}$ be the minimal non-empty subset of $\boldsymbol{\Xi}$ such that, for each $\boldsymbol{\xi}_\mu \in \mathcal{S}(\tilde{\boldsymbol{\xi}})$, $\tilde{\boldsymbol{\xi}}$ is no longer a local minimum of the energy $E_\beta(\cdot, \boldsymbol{\Xi} \setminus \{\boldsymbol{\xi}_\mu\})$ that excludes the memory. Then we define $\tilde{\boldsymbol{\xi}} \in \mathcal{M}_\varepsilon$ as a ε-**locally emergent memory** if there is some original memory $\boldsymbol{\xi}_\mu \in \mathcal{S}(\tilde{\boldsymbol{\xi}})$ which still is a local minimum of the energy $E_\beta(\cdot; \boldsymbol{\Xi})$. If every original memory $\boldsymbol{\xi}_\mu \in \mathcal{S}(\tilde{\boldsymbol{\xi}})$ is a local minimum of $E_\beta(\cdot; \boldsymbol{\Xi})$, we call $\tilde{\boldsymbol{\xi}} \in \mathcal{M}_\varepsilon$ a ε-**local strongly emergent memory**.*

While Definition 7.2 discusses emergence at a global energy level, Definition 7.3 characterizes emergence locally for each of the novel local minima. This distinction is important as we see emergence as a general property of the energy function that can be driven by a subset of the memories. Definition 7.2 requires all original memories to be retrievable, while in Definition 7.3 we allow for emergence due to the interaction of stored patterns in a system even if not all original memories are retrievable, so long as a critical subset of the original memories are. Global emergence implies the existence of at least one local strongly emergent memory; every globally emergent memory is a local strongly emergent one. However, the existence of a local strongly emergent memory does not imply global emergence. As with global emergence, we can see that a DenseAM with LSE energy does not also have locally emergent memories. First note that, for a finite β, all local minima are novel local minima, and the minimal set $\mathcal{S}(\tilde{\boldsymbol{\xi}})$ for a novel local minimum $\tilde{\boldsymbol{\xi}}$ is the whole set $\boldsymbol{\Xi}$ given the infinite support of the exponential function, with none of them being a local minimum. For $\beta = \infty$, the set of novel local minima $\mathcal{M}_\varepsilon$ is empty. So in both cases, the required conditions are not satisfied and there are no locally emergent memories with LSE energy.

Note that these novel local minima are different from the well-studied spurious memories or parasitic memories [203, 204]. In classical AMs, spurious memories start appearing when the AM is packed with memories beyond its memory capacity. In contrast, the appearance of emergence (novel local minima) does not seem to be related to whether the DenseAM is over or under capacity — as we show in Figure 7.1, a locally emergent memory can appear even with just 2 stored patterns of any dimension. Spin-glass states [205] do not occur in either the LSE or LSR energy due to our use of Euclidean similarity over the dot product in both energies.

The next result provides explicit characterization of the form of novel memories in LSR energy.

Proposition 7.2. *Consider the LSR energy in Equation (7.3). For any* $\mathbf{x} \in$ *interior*$(\mathcal{X})$*, letting* $B(\mathbf{x}) \triangleq \{\mu \in [\![M]\!] : \|\mathbf{x} - \boldsymbol{\xi}_\mu\| \leq \sqrt{2/\beta}\}$*, there is a local minima of the LSR energy which is given by* $\frac{1}{|B(\mathbf{x})|} \sum_{\mu \in B(\mathbf{x})} \boldsymbol{\xi}_\mu$*.*

Note that when $|B(\mathbf{x})| = 1$, the local minima in Proposition 7.2 is exactly the stored memory $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^M$. With $|B(\mathbf{x})| > 1$, it is not equal to any of the original memories $\{\boldsymbol{\xi}_\mu, \mu \in [\![M]\!]\}$ (with probability 1). The region $\{\mathbf{x} \in \mathcal{X} : |B(\mathbf{x})| > 1\} \subset \mathcal{X}$ is precisely characterized as $\left(\cup_{\mu \in [\![M]\!]} \mathsf{S}_\mu\right) \setminus \left(\cup_{\mu \in [\![M]\!]} S_\mu(\Delta)\right)$ where S_μ is the region of finite energy around the μ^{th} memory and $S_\mu(\Delta)$ (defined in Theorem 7.1) is the distinct attracting basin for the μ^{th} memory. The following theorem shows that this LSR based DenseAM is capable of simultaneously retrieving all (up to exponentially many) memories while also creating many novel local minima, and quantifies this phenomenon precisely.

Theorem 7.2. *Consider a DenseAM parameterized with* M *memories* $\boldsymbol{\Xi} = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M\}$ *sampled uniformly from* $\mathcal{X}$ *with* $vol(\mathcal{X}) = V < \infty$ *and the LSR energy* $E_\beta^{LSR}(\cdot; \boldsymbol{\Xi})$ *defined in Equation (7.3). For each novel local minimum* $\mathbf{x}^* = \frac{1}{|B(\mathbf{x}^*)|} \sum_{\mu \in B(\mathbf{x}^*)} \boldsymbol{\xi}_\mu$ *given in Proposition 7.2, define* $D_{\max}(\mathbf{x}^*) := \max_\mu \|\mathbf{x}^* - \boldsymbol{\xi}_\mu\|$*,*

$$\delta_{\min}(\mathbf{x}^*) := \min_{\mu \in B(\mathbf{x}^*)} \left(\frac{2}{\beta} - \|\mathbf{x}^* - \boldsymbol{\xi}_\mu\|^2\right), \ \gamma_{\min}(\mathbf{x}^*) := \min_{\nu \notin B(\mathbf{x}^*)} \left(\|\mathbf{x}^* - \boldsymbol{\xi}_\nu\|^2 - \frac{2}{\beta}\right).$$

Then, for all $\beta > 0$ *such that* $\max_{\mathbf{x}^*} \delta_{\min}(\mathbf{x}^*) > 0$ *and* $\min_{\mathbf{x}^*} \gamma_{\min}(\mathbf{x}^*) > 0$*, there exists an* $\varepsilon := \min_{\mathbf{x}^*} \left(\sqrt{D_{\max}^2(\mathbf{x}^*) + \min\{\delta_{\min}(\mathbf{x}^*), \gamma_{\min}(\mathbf{x}^*)\}} - D_{\max}(\mathbf{x}^*)\right) > 0$ *such that* $E_\beta^{LSR}(\cdot)$ *satisfies the* ε*-global emergence condition (Definition 7.2) with high probability, as:*

(a) With probability at least $\delta \in (0, 1)$*, and* $M = \Theta\left(\sqrt{1-\delta}\exp(\alpha d)\right)$ *for a positive*

α, all memories are retrievable as per Theorem 7.1 with the value of the minimum pairwise distance $r = \min_{\mu,\nu\in[\![M]\!],\mu\neq\nu} \|\boldsymbol{\xi}_\mu - \boldsymbol{\xi}_\nu\| \geq (V_d/V)^{-1/d} e^{-2\alpha}$ and per-memory basin radius $\Delta \in (0, (V_d/V)^{-1/d} e^{-2\alpha})$ with a $\beta \leq 2/((V_d/V)^{-1/d} e^{-2\alpha} - \Delta)^2$, where V_d is the volume of the unit ball in $\mathbb{R}^d$.

(b) For each novel local minimum $\mathbf{x}^*$, there exists a radius

$$r^* := \sqrt{D_{\max}^2(\mathbf{x}^*) + \min\{\delta_{\min}(\mathbf{x}^*), \gamma_{\min}(\mathbf{x}^*)\}} - D_{\max}(\mathbf{x}^*) > 0, \tag{7.4}$$

such that $S_{\mathbf{x}^*}(r^*) = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \mathbf{x}^*\| \leq r^*\}$ forms a basin around the novel memory $\mathbf{x}^*$, and for any $\mathbf{x} \in S_{\mathbf{x}^*}(r^*)$, the output of the DenseAM via energy gradient descent is exactly $\mathbf{x}^*$, implying the novel memories are retrievable.

Furthermore, with probability at least $1 - M^{-2}$, the number of ε-globally emergent memories is

$$O\left(\exp\left(M \frac{V_d}{V} \left(\frac{2}{\beta}\right)^{\frac{d}{2}} \log\left(\frac{eV}{V_d}\left(\frac{\beta}{2}\right)^{d/2}\right)\right)\right). \tag{7.5}$$

In particular, for fixed $\beta > 0$ and d, the bound grows with M whenever $\beta > 0$ satisfies $|B(\mathbf{x})| \in (1, M]$ due to Proposition 7.2. Here, for any $\mathbf{x} \in \mathcal{X}$, a large β leads to a small $|B(\mathbf{x})|$, and $|B(\mathbf{x})| > 1$ is the required condition for a novel local minima; a small β leads to a large $|B(\mathbf{x})|$, and $|B(\mathbf{x})| \leq M$ stands for the case $B(\mathbf{x})$ at most covers the entire domain $\mathcal{X}$.

The above result demonstrates that with the LSR energy, it is possible to exactly memorize all original patterns (with high probability) and still generate new patterns — what we term emergent memories (Definition 7.2). This behavior is surprising in the same way interpolating models in deep learning generalize unexpectedly well: both challenge the classical bias-variance intuition [198, 199]. While LSE-based models also produce novel memories, they typically do so at the expense of perfect recall of the original patterns. In Proposition 7.1 we show that LSE based DenseAMs do not have the global emergence property. This distinction highlights a key contribution of our work: *new memory creation need not come at the cost of perfect memorization.*

Next, we provide an exact order (i.e., upper and lower bounds) of the number of emergent memories under a grid design assumption.

Proposition 7.3. *If* $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^M$ *form a grid over* $\mathcal{X}$ *of equal size with* $Vol(\mathcal{X}) = V < \infty$*, the number of emergent memories is of order* $\Theta\left(\left(M^{1/d} - \lambda^{1/d} + 1\right)^d\right)$*, where* $\lambda = \Theta\left(MV^{-1}\left(8/\beta\right)^{\frac{d}{2}}\right)$ *and for* $\beta > 0$ *such that* $1 < \lambda \leq M$*.*

Note that we showed an explicit form of the emergent memories $\mathbf{x}^* = \frac{1}{|B(\mathbf{x}^*)|}\sum_{\mu \in B(\mathbf{x}^*)} \boldsymbol{\xi}_\mu$, where $B(\mathbf{x}^*) = \left\{\mu : \|\mathbf{x}^* - \boldsymbol{\xi}_\mu\| < \sqrt{2/\beta}\right\} \subset \{1, \ldots, M\}$. Equation (7.5) in Theorem 7.2 is stated under the uniform sampling regime, and Proposition 7.3 is stated under a fixed grid setting. In general, the number of emergent memories varies according to the specific geometry of the stored patterns $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^M$, i.e., whether such a subset of $\{1, \ldots, M\}$ can be realized by a ball $\left\{\|\mathbf{x} - \boldsymbol{\xi}_\mu\| < \sqrt{2/\beta}\right\}$. This can grow much faster than a linear order of M, and is naively bounded by 2^M.

7.4 Experiments

7.4.1 Quantifying the scaling of emergent memories

How many local minima do we see in practice as we: (a) vary the number of stored patterns, (b) change the dimensionality of those patterns, and (c) vary the inverse temperature β? We observe that, at critical values of β, we can create *orders of magnitude* more emergent memories than stored patterns. These results are shown in Figure 7.3 (left).

To quantify the number of local minima induced by the LSR energy, we uniformly sample M patterns from the d-dimensional unit hypercube to serve as memories $\boldsymbol{\Xi}$. We enumerate all possible local minima of the LSR energy by computing the centroid $\bar{\boldsymbol{\xi}}_{\mathcal{K}} := |\mathcal{K}|^{-1}\sum_{\mu \in \mathcal{K}} \boldsymbol{\xi}_\mu$ for every possible subset of stored patterns $\mathcal{K} \subseteq [\![M]\!]$ (there are 2^M possible subsets if we allow for singleton sets). For each subset, we first check that its centroid is supported (i.e., that $E_\beta^{\text{LSR}}(\bar{\boldsymbol{\xi}}_{\mathcal{K}}; \boldsymbol{\Xi}) < \infty$ at $\epsilon = 0$), and then declare that $\bar{\boldsymbol{\xi}}_{\mathcal{K}}$ is a local minimum of the LSR energy if $\left\|\nabla E_\beta^{\text{LSR}}(\bar{\boldsymbol{\xi}}_{\mathcal{K}}; \boldsymbol{\Xi})\right\| < \delta$ for small $\delta > 0$. β values are varied across the "interesting" regime between fully overlapping support regions (a single local minimum in the unit hypercube) to fully disjoint support regions around each memory. Further experimental details are provided in Appendix D.1.1.

Certain values of β yield particularly interesting behavior. For example, we observe that LSR can create orders of magnitude more emergent memories under ranges of β where: (i) a majority ($>60\%$) of stored patterns are also recoverable, and (ii) around 20 percent of the unit hypercube is still supported. Note that in each experiment there are choices of β such that the LSR energy does not exhibit *global emergence* (Definition 7.2, i.e., at low β

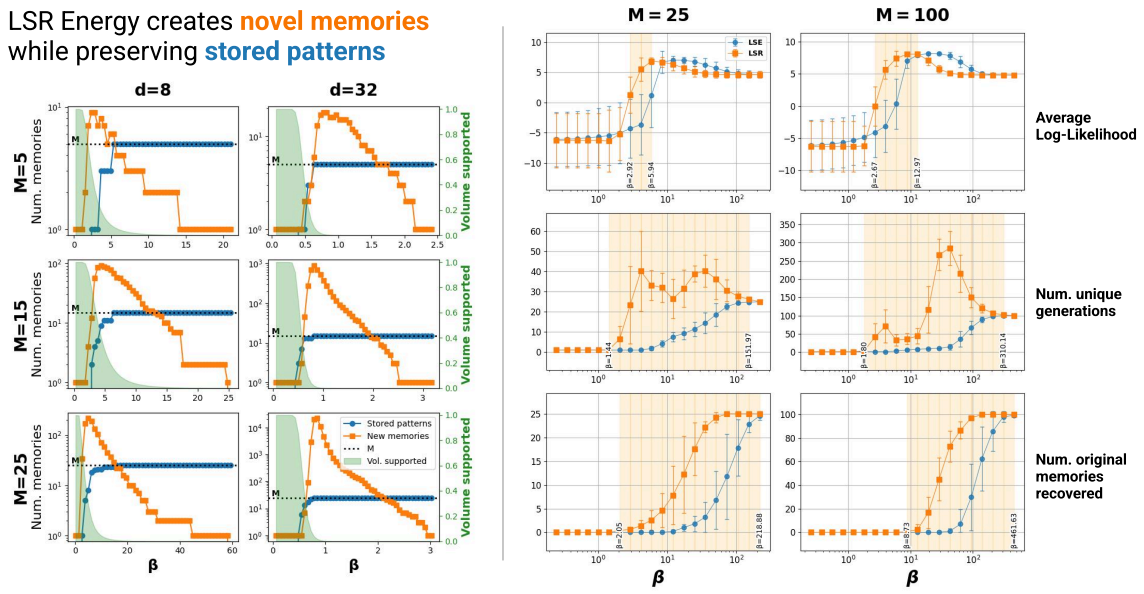

Figure 7.3: (Left) Analyzing local minima in LSR energy reveals a number of **novel memories** several orders of magnitude larger than M, the number of stored patterns, at critical values of β (note that the y-axes are logscale). These emergent memories occur even while still preserving the **stored patterns** as memories. Smaller values of β have a larger **region of support** on the unit hypercube. (Right) Given samples from some known true density function (in this case, a $k = 10$ mixture of 8-dim Gaussians with means drawn uniformly from the unit hypercube and $\sigma = 0.1$), memories from LSR energy have a log-likelihood comparable to, and occasionally slightly higher than, LSE under the true density function. Note that LSR achieves comparable log-likelihood while having more unique samples than LSE, even when both are seeded with the same $N = 500$ queries. Regions of β where LSR outperforms LSE on a metric are specified by the orange regions. Error bars indicate the standard error across 5 different seeds for sampling stored patterns and initial queries.

where novel memories are forming but not all stored patterns are yet retrievable). However, in these regions *local emergence* (Definition 7.3) of the novel memories still holds (see Figure 7.1 for intuition).

7.4.2 Generative quality of emergent memories

LSR memories are certainly more diverse than those of LSE, but do they represent more "meaningful" samples from a true, underlying density function $p(\mathbf{x})$ (as measured by their log-likelihood)? The experimental setup is as follows: Let $p(\mathbf{x})$ be a mixture of k Gaussians whose means $\boldsymbol{\mu}_i \sim \mathcal{U}([0,1]^d)$ for $i \in [\![k]\!]$ are uniformly sampled from the d-dimensional unit hypercube with scalar ($\sigma = 0.1$) covariances such that $p(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \sigma^2 \mathbf{I}_d)$. We sample M points $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M\}, \boldsymbol{\xi}_\mu \sim p(\mathbf{x})$ to serve as the stored patterns $\boldsymbol{\Xi}$ used to parameterize both the LSE and LSR energies from Equation (7.3). Define a thin *support*

boundary induced by pattern $\boldsymbol{\xi}_\mu$ to be $\text{supp}[\boldsymbol{\xi}_\mu; \delta] = \{\mathbf{x} : 2\beta^{-1} - \delta \leq \|\mathbf{x} - \boldsymbol{\xi}_\mu\|^2 < 2\beta^{-1}\}$ for some small $\delta > 0$. Then, for initial points $\mathbf{x}_n^{(0)}$, $n \in [\![N]\!]$ sampled from the support boundary around each stored pattern,[2] LSE memories can be found using gradient descent

$$\mathbf{x}_n^{(t)} = \mathbf{x}_n^{(t-1)} - \alpha \nabla E_\beta^{\text{LSE}}(\mathbf{x}_n^{(t-1)}; \boldsymbol{\Xi}), \tag{7.6}$$

until convergence to a *memory* $\mathbf{x}_n^\star$. We use Algorithm 2 to efficiently find the LSR memory corresponding to each initial point. Thus we have N "samples" (memories) from both LSE and LSR on which we compare three metrics of interest in Figure 7.3 (right):

1. **Average Log-Likelihood**. Do LSR memories $\mathbf{x}_{\text{LSR}}^\star$ have higher $\log p(\mathbf{x}_{\text{LSR}}^\star)$ than LSE memories?

2. **Number of Unique Samples**. Does high $\log p(\mathbf{x}_{\text{LSR}}^\star)$ occur alongside many emergent memories?

3. **Number of Original Memories Recoverable**. Does high $\log p(\mathbf{x}_{\text{LSR}}^\star)$ occur alongside high numbers of preserved memories? How does this trend compare with LSE memory performance?

The results tell a consistent story. Despite LSE energy being a more natural choice to model the underlying Mixture of Gaussians' density $p(\mathbf{x})$ (LSE has a Gaussian kernel that makes it ideal for the modeling task), LSR can match LSE in log-likelihood while simultaneously generating more diverse samples and preserving the stored patterns. See Appendix D.1.2 for more experimental results and extended discussion.

7.4.3 Emergent memories in latent space

What do emergent memories look like when LSR is applied to real-world datasets? To study this behavior, we use a VAE to encode MNIST and Tiny ImageNet [1] images into latent vectors that serve as the stored patterns for LSR and LSE energies (see Figure 7.4). Using a carefully chosen β, we compute *all* memories (both preserved and novel) for each energy. The emergent memories of LSR in principle are simply the centroids of small subsets of the stored patterns, yet when decoded they appear as plausible and creative generations.

[2] We use the same initial points to seed the dynamics of both E^{LSR} and E^{LSE}. See Appendix D.1.2 for details.

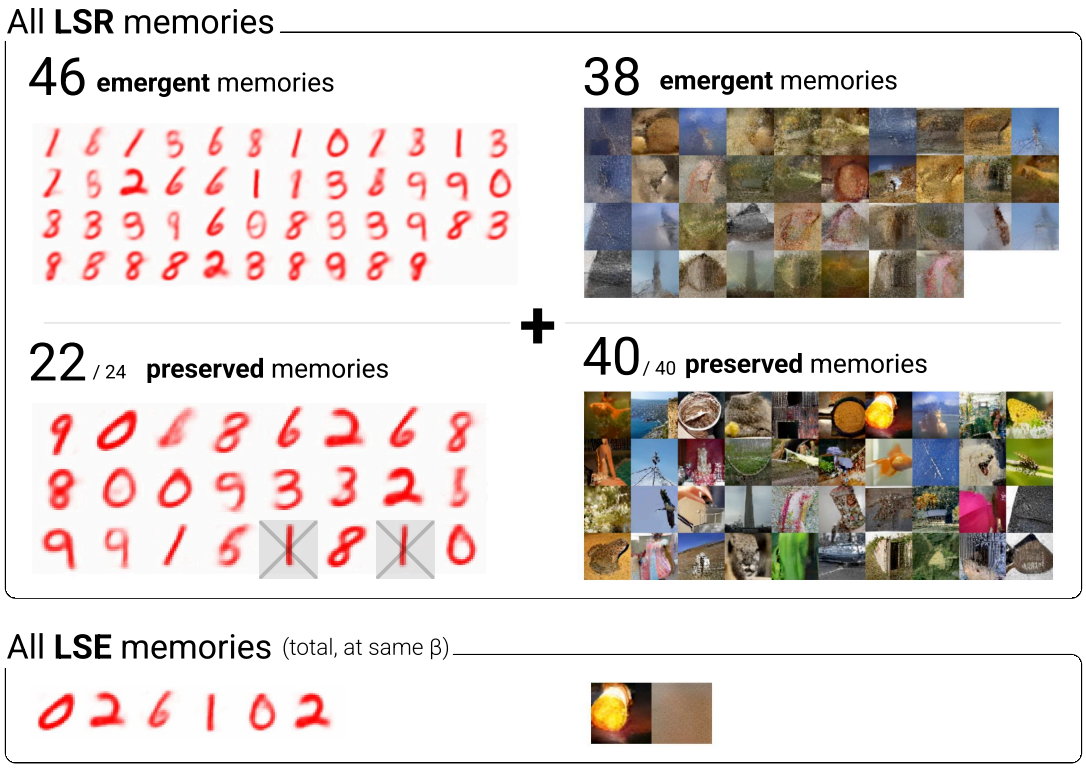

Figure 7.4: LSR's emergent memories appear as novel, creative generations when the energy is applied to a semantically meaningful latent space. **(Left)** 24 randomly-selected MNIST images are encoded into 10-dim VAE latents and stored into an LSR- and LSE-energy using a carefully chosen β (see Algorithm 3). Gray boxes indicate which stored patterns were not preserved at the chosen β. **(Right)** 40 Tiny ImageNet [1] images are encoded into 256-dim latents using a pretrained VAE [2] and stored into an LSR- and LSE-energy using a carefully chosen β. Note that in this Tiny ImageNet example the LSR energy is, by definition, *globally emergent* since all stored patterns are recoverable, while the MNIST example is not. See Appendix D.1.3 for experiment details.

With the same β value, LSR generates an order of magnitude more total memories than LSE. While this choice of β is somewhat arbitrary and could be tuned separately for each energy, LSE would only ever be able to retrieve up to M memories (where M is the total number of stored patterns). See Appendix D.1.3 for full experiment details.

Emergent memories are mechanistically simple: they are simply the centroids of small subsets of the stored patterns. The semantic novelty of the emergent memories in Figure 7.4 occurs because the latent space is structured to be semantically meaningful, where averaging two or more stored patterns produces seemingly novel semantics.

7.4.4 Emergent memories in pixel space

When the stored patterns live in a semantically structured latent space, as is done in Figure 7.4, the energy landscape inherits the structure such that centroids of subsets of stored patterns appear semantically novel. However, the latent space visually obscures the mechanistic simplicity of emergent memories. Thus, we repeat the experiment in pixel space to reinforce how emergent memories work.

We store 8 randomly selected MNIST images as rasterized pixels (normalized between 0 and 1) into the LSR energy. The resulting emergent minima and stored patterns are shown in Figure 7.5, where the β is chosen to balance the number of emergent minima and the retrievability of the stored patterns (*global emergence* Definition 7.2).

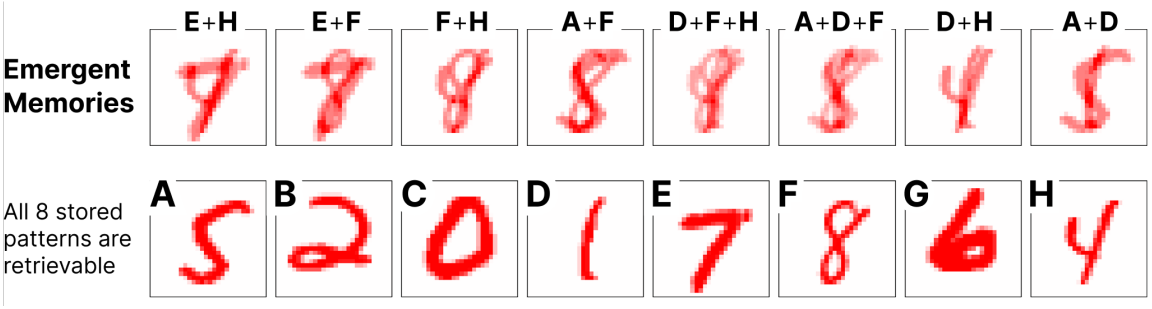

Figure 7.5: **Emergent memories** are centroids of subsets of stored patterns, shown clearly when 8 stored images are visualized in pixel space alongside their induced emergent memories. Stored patterns (bottom, indexed **A**-**H**) merge to form emergent memories (top, labeled by the stored patterns that merged to form the emergent memory). β is chosen such that the number of emergent memories is approximately the same as the number of stored patterns.

7.5 Discussion

Emergent memories are a powerful tool for creating novel samples, but the "meaningfulness" of these samples is a nuanced question that depends heavily on the specific application domain and task requirements. For example, in Figure 7.3 (right) we show that the LSR energy approximates an unknown p.d.f. better than LSE's energy while simultaneously generating diverse samples. This represents a desirable behavior of emergence, since high log-likelihood samples from the LSE energy are quite homogeneous, causing 500 initial queries to converge to the same $\sim$10 memories. In this density estimation context, the emergent memories serve a clear functional purpose: they capture meaningful interpolations within the data distribution that improve generalization.

However, consider the novel memories from Tiny Imagenet in Figure 7.4. Though visually plausible, many of the emergent generations appear blurry and would be considered "undesirable" from the perspective of a high-fidelity image generation model. We discuss the limitations of emergent memories further in Appendix D.2.

We note that there are potential parallels between emergent memories and "hallucinations" as observed in LLMs. We discuss the philosophical similarities between emergent memories and hallucinations in Appendix D.3.1. It is also interesting to note that the LSR Energy presented in this work can have a feasible biological implementation using bipartite neurons. See further discussion in Appendix D.3.2.

Finally, we reiterate that there are many choices for alternative kernels, and not just the Epanechnikov kernel. We discover that many kernels with compact support are capable of producing emergent memories and even manifolds, and the energy landscapes they produce can look quite different from each other. To this end, we show the "basin merging" behavior across different kernels in 1D in Figure D.5 and we include an extended discussion on these other kernels in Appendix D.4.

7.6 Conclusion

Our work introduces the LSR energy function which achieves the surprising combination of exact memorization of exponentially many patterns and the emergence of new, meaningful memories, thereby providing a powerful alternative to traditional AM formulations. The properties of the LSR energy define a novel class of *emergent memories* in Dense Associative Memory systems — a phenomenon not observed in prior DenseAM formulations with simultaneous exact retrieval. Unlike conventional models (e.g., LSE-based energies) where generalization (formation of local minima different from training data) typically comes at the cost of perfect memorization, LSR demonstrates that these objectives can coexist harmoniously in a single energy function. We additionally demonstrated that the diverse memories created by LSR achieve log-likelihood comparable to LSE when sampling from a true density function, while generating an order of magnitude more unique memories. Finally, we showed that when applied to latent representations of real-world image datasets, LSR's emergent memories represent plausible and creative generations.

Part III

UNIFYING ASSOCIATIVE MEMORY

The preceding parts develop several ways to unlock the capabilities of AMs, but they also expose their fragmented development. Each model is often written with its own notation, energy function, and architectural assumptions, making it difficult to compare designs or compose their useful pieces. Part III asks how these models can be expressed as instances of a common design language that could enable even more powerful latent memory models.

This part introduces HAMUX, a universal framework for describing AM energies using a toolbox familiar to deep learning practitioners. Rather than treating each AM as a separate historical architecture, HAMUX decomposes energy-based systems into reusable *neuron layers* that carry dynamic variables and *hypersynapses* that encode interactions among them. The total energy of any HAMUX graph can be minimized using *local neuron interactions* that require *no differentiation through non-linearities*, properties that are highly desirable for physical computation. In doing so, this part serves as a kind of glue that turns the dissertation's individual contributions into a broader framework for developing new energy-based memory systems.

Chapter 8

HAMUX: A Universal Abstraction for Hierarchical Hopfield Networks. Benjamin Hoover, Duen Horng Chau, Hendrik Strobelt, and Dmitry Krotov. *The Symbiosis of Deep Learning and Differential Equations II, 2022*. PDF

Modern Methods in Associative Memory. Dmitry Krotov, Benjamin Hoover, Parikshit Ram, and Bao Pham. *arXiv preprint arXiv:2507.06211, 2025*. PDF

CHAPTER 8
HAMUX

While Dense Associative Memories (DenseAMs) are mathematically aesthetic and simple to analyze, they have a strong limitation. They do not have a hierarchical structure of representations, a crucial component for deep learning that limits their ability to handle complex patterns from real-world datasets. Their energy is rigidly tied to a single synaptic weight matrix that constrains what type of patterns and relationships can be learned by the network.

Modern AM architectures need a compositional language for hierarchy, heterogeneous non-linearities, and reusable interactions. Krotov's Hierarchical Associative Memory paper [77] showed that the Hopfield framework extends beyond the classical two-layer setting to systems with many neuron layers, local connectivity, native top-down feedback, and a global energy that decreases along the dynamics. However, designing models within this family from scratch makes it difficult to compare architectures, reuse components, or separate the role of dynamic variables from the interactions that connect them. The goal of HAMUX is to make AM designs resemble the composition of neural-network modules while preserving the energy descent and convergence structure that distinguish AMs from ordinary feedforward computation.

HAMUX represents an AM as a hypergraph whose nodes are dynamic states with equipped non-linearities and whose weighted hyperedges describe synaptic interactions between neurons. This idea of representing a global objective through local factors has a long history in factor graphs, graphical models, and energy-based learning [206, 207, 36]. The contribution of HAMUX is to specialize this perspective to energy-based memories whose neuron states evolve by Lyapunov-stable dynamics.

The modular energy perspective of HAMUX decomposes the energy of any AM into standardized components: **neuron layers** that encode dynamic variables and **hypersynapses** that encode their interactions. *The total energy of the system is the sum of each individual component energy.* This framework clarifies how existing methods relate to each other and provides a systematic language for designing architectures such as Hierarchical Associative Memories [77], Energy Transformers [11], neuron-astrocyte networks [188], and many others.

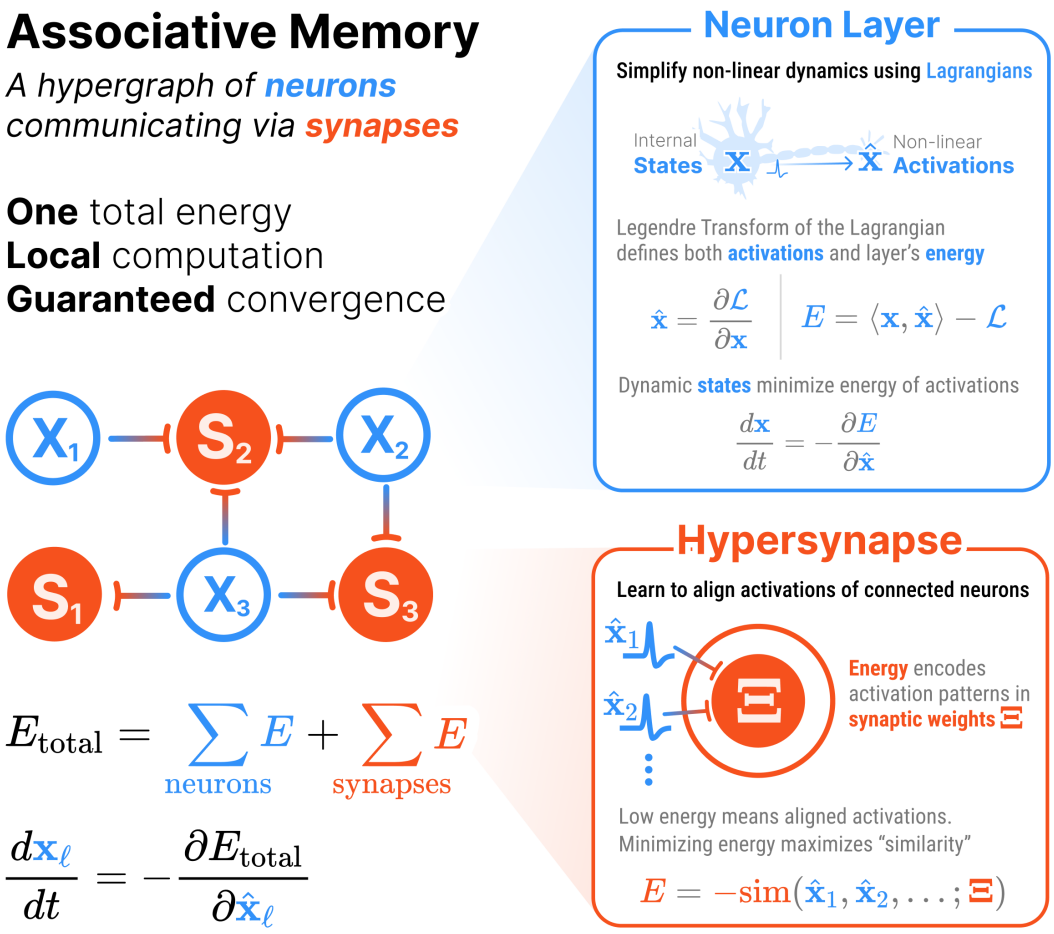

Figure 8.1: **HAMUX hypergraph diagrams are a graphical depiction of an AM** whose total energy is the sum of the neuron layer (node) and hypersynapse (hyperedge) energies. Inference is done recurrently, modeled by a system of differential equations where each neuron layer's hidden state updates to minimize the total energy. When all non-linearities are captured in the dynamic neurons, inference becomes a local computation that avoids differentiating through non-linearities.

We refer to this generalized abstraction of AMs as **HAMUX** after the software library that introduced it [17]. Here, HAMUX stands for "**H**ierarchical **A**ssociative **M**emory **U**ser e**X**perience". The abstraction, however, is more fundamental than its specific software implementation.

8.1 Overview of HAMUX

The HAMUX hypergraph has two component types whose energies compose into the total energy of the AM.

A **neuron layer** is a *node* of the energy hypergraph that captures a non-linearity in the network, such as `ReLU`, `sigmoid`, `tanh`, `softmax`, or `layernorm`. We call these

non-linearities *activations*, and their inputs or *pre-activations* serve as the dynamic variables of the system. For example, a neuron layer can capture the computation $\hat{\mathbf{x}} = \texttt{ReLU}(\mathbf{x})$, which has activations $\hat{\mathbf{x}}$ and pre-activations $\mathbf{x}$ that serve as the dynamic *internal state* for this neuron layer. See § 8.2 for more details on designing neurons.

A **hypersynapse** is an *edge* (or more accurately, a *hyperedge*) of the energy hypergraph that parameterizes how similar the activations of its connected neuron layers are. For example, a simple dense hypersynapse may take the form $E_{xy}(\hat{\mathbf{x}}, \hat{\mathbf{y}}; \boldsymbol{W}) = -\hat{\mathbf{y}}^\intercal \boldsymbol{W} \hat{\mathbf{x}}$, where $\boldsymbol{W}$ is a synaptic weight matrix. If we assume L2-normalized $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$, minimizing E_{xy} with respect to the activations maximizes their cosine similarity or *alignment* as modulated by the synaptic matrix $\boldsymbol{W}$. The negative gradient of this energy with respect to $\hat{\mathbf{x}}$ or $\hat{\mathbf{y}}$ looks like a `Dense` linear transformation, though more complex synaptic energies can be chosen to look like `Conv`, `Pooling`, or even `Attention` layers. See § 8.3 for more details on designing hypersynapses.

For a system of L neuron layers and S hypersynapses, the total energy of the system is

$$E_{\text{total}} = \sum_{\ell=1}^{L} E_\ell^{\text{neuron}} + \sum_{s=1}^{S} E_s^{\text{synapse}}. \tag{8.1}$$

The total energy is structured such that the activations of a neuron layer affect only connected hypersynapses and itself. Let $\hat{\mathbf{x}}_\ell$ and $\mathbf{x}_\ell$ represent the activations and internal states of neuron layer ℓ, and let $\texttt{N}(\ell)$ represent the set of hypersynapses that connect to neuron layer ℓ. The following update rule describes how neuron internal states $\mathbf{x}_\ell$ minimize the total energy using only local signals:

$$\tau_\ell \frac{d\mathbf{x}_\ell}{dt} = -\frac{\partial E_{\text{total}}}{\partial \hat{\mathbf{x}}_\ell} = -\left(\sum_{s \in \texttt{N}(\ell)} \frac{\partial E_s^{\text{synapse}}}{\partial \hat{\mathbf{x}}_\ell} \right) - \frac{\partial E_\ell^{\text{neuron}}}{\partial \hat{\mathbf{x}}_\ell} = \mathcal{I}_{x_\ell} - \mathbf{x}_\ell, \tag{8.2}$$

where $\mathcal{I}_{x_\ell} := -\sum_{s \in \texttt{N}(\ell)} \nabla_{\hat{\mathbf{x}}_\ell} E_s^{\text{synapse}}$ is the *total synaptic input current* into neuron layer ℓ. This current is fundamentally local and serves to minimize the energy of connected hypersynapses. The time constant for neurons in layer ℓ is denoted by τ_ℓ. When the activations $\hat{\mathbf{x}}_\ell$ are bounded, the above system is guaranteed not to increase the energy, and it converges to fixed points when the boundedness conditions discussed below are met.

8.2 Dynamical Neurons and their Lagrangians

A *neuron layer* describes the dynamical variables in an AM and represents a *node* of the computational hypergraph. Each neuron layer has an *internal state* $\mathbf{x}$ which evolves over time and an *activation* $\hat{\mathbf{x}}$ that forwards a signal to the rest of the network. Neurons can be understood as analogous to the activation functions of standard neural networks, where $\mathbf{x}$ are the "pre-activations" and $\hat{\mathbf{x}}$ are the outputs, e.g., $\hat{\mathbf{x}} = \texttt{ReLU}(\mathbf{x})$.

In order to define a neuron layer's energy, AMs employ two mathematical tools from physics: *convex Lagrangian functions* and the *Legendre transform*. For each neuron layer, we define a convex, scalar-valued Lagrangian $\mathcal{L}_x(\mathbf{x})$. The Legendre transform $\mathcal{T}$ of this Lagrangian produces the dual variable $\hat{\mathbf{x}}$, which is the activation, and the dual function $E_x(\hat{\mathbf{x}})$, which is the neuron energy:

$$\begin{aligned} \hat{\mathbf{x}} &= \nabla \mathcal{L}_x(\mathbf{x}) \quad \text{(activation function)} \\ E_x(\hat{\mathbf{x}}) = \mathcal{T}[\mathcal{L}_x] &= \langle \mathbf{x}, \hat{\mathbf{x}} \rangle - \mathcal{L}_x(\mathbf{x}) \quad \text{(dual energy)}. \end{aligned} \tag{8.3}$$

Here, $\langle \cdot, \cdot \rangle$ is the element-wise inner product. Because $\mathcal{L}_x$ is convex, the Jacobian of the activations, $\frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{x}} = \nabla^2 \mathcal{L}_x(\mathbf{x})$, is positive definite. This important point is summarized in Figure 8.1.

The energy $E_x(\hat{\mathbf{x}})$ has another useful property: *its gradient equals the hidden states*. Thus, when we minimize the energy of neurons in the absence of any other signal, we observe exponential decay. This keeps the dynamic behavior of the system bounded and well-behaved, especially for very large values of $\mathbf{x}$:

$$\frac{d\mathbf{x}}{dt} = -\nabla_{\hat{\mathbf{x}}} E_x(\hat{\mathbf{x}}) = -\mathbf{x}. \tag{8.4}$$

Summary A neuron layer is just a convex function $\mathcal{L}_x$, the Lagrangian, applied to an internal state $\mathbf{x}$. The Legendre transform of this Lagrangian then automatically provides two things: the activation function $\hat{\mathbf{x}} = \nabla \mathcal{L}_x(\mathbf{x})$ and the dual energy representation $E_x(\hat{\mathbf{x}})$. This mathematical machinery abstracts away much of the complexity of non-linearities and gives us a simpler system to work with.

Proof: Neuron energy gradient equals hidden states

Show that $\frac{\partial E_x(\hat{\mathbf{x}})}{\partial \hat{\mathbf{x}}} = \mathbf{x}$. This identity matters because it turns neuron energy contributions into simple decay terms $-\mathbf{x}$, leaving synaptic input currents as the only nontrivial signals a neuron must receive.

$$\begin{aligned}
\frac{\partial E_x(\hat{\mathbf{x}})}{\partial \hat{\mathbf{x}}} &= \frac{\partial}{\partial \hat{\mathbf{x}}} \left(\langle \mathbf{x}, \hat{\mathbf{x}} \rangle - \mathcal{L}_x(\mathbf{x}) \right) \\
&= \mathbf{x} + \hat{\mathbf{x}} \frac{\partial \mathbf{x}}{\partial \hat{\mathbf{x}}} - \frac{\partial \mathcal{L}_x(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \hat{\mathbf{x}}} \\
&= \mathbf{x} + \hat{\mathbf{x}} \frac{\partial \mathbf{x}}{\partial \hat{\mathbf{x}}} - \hat{\mathbf{x}} \frac{\partial \mathbf{x}}{\partial \hat{\mathbf{x}}} \\
&= \mathbf{x}.
\end{aligned}$$

8.3 Hypersynapses

The activations of one neuron layer are sent to other neurons via communication channels called *hypersynapses*. At its most general, a hypersynapse is a scalar-valued energy function defined on top of the activations of connected neuron layers. For example, a hypersynapse connecting neuron layers X and Y has an *interaction energy* $E_{xy}(\hat{\mathbf{x}}, \hat{\mathbf{y}}; \boldsymbol{\Xi})$, where $\boldsymbol{\Xi}$ represents the *synaptic weights* or learnable parameters. $E_{xy}(\hat{\mathbf{x}}, \hat{\mathbf{y}}; \boldsymbol{\Xi})$ encodes the desired relationship between activations $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$. When this energy is low, the activations satisfy the relationship encoded by the synaptic weights $\boldsymbol{\Xi}$. During energy minimization, the system adjusts the activations to reduce all energy terms, which means synapses effectively "pull" the connected neuron layers toward configurations encoded in the parameters that minimize their interaction energy.

Hypersynapses are closely related in spirit to factors in factor graphs: both represent local terms whose composition defines a global function [206]. This perspective also appears in graphical models and Gibbs distributions, where local clique potentials define a global energy over an undirected graph [207, 36]. The distinction in HAMUX is that the variables are dynamical neuron states, the factors are differentiable interaction energies, and inference is specified as local energy descent with convergence guarantees.

Hypersynapses in the HAMUX framework differ from biological synapses in two fundamental ways.

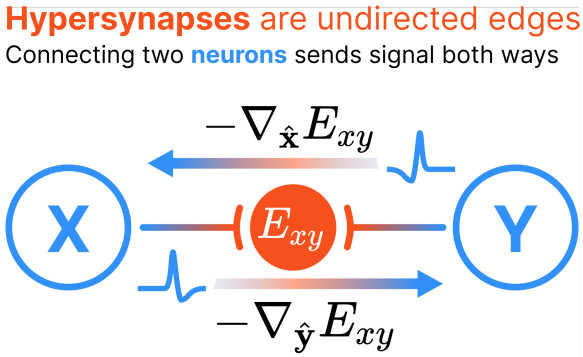

Figure 8.2: **Hypersynapses are represented as undirected hyperedges in a hypergraph.** Shown is an example pairwise synapse, which is a single energy function $E_{xy}(\hat{\mathbf{x}}, \hat{\mathbf{y}}; \Xi)$ defined on the activations $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ from connected nodes, which necessarily propagates signal to both connected nodes. Here, *signal* is defined as the negative gradient of the interaction energy with respect to the connected layer's activations. For example, layer X receives signal $\mathcal{I}_x = -\nabla_{\hat{\mathbf{x}}} E_{xy}(\hat{\mathbf{x}}, \hat{\mathbf{y}}; \Xi)$ while layer Y receives signal $\mathcal{I}_y = -\nabla_{\hat{\mathbf{y}}} E_{xy}(\hat{\mathbf{x}}, \hat{\mathbf{y}}; \Xi)$. This is in contrast to biological synapses, which are directional and only propagate signal in one direction from layer X to Y, needing a separate synapse to bring information back from Y to X.

1. **Hypersynapses can connect any number of layers simultaneously**. Meanwhile, biological synapses connect only two neurons. This officially makes each hypersynapse a *hyperedge* in graph theory terms, and the multi-neuron behavior is evident in the flexibility we have when defining the synaptic energy's signature $E_{xyz}(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$.

2. **Hypersynapses are undirected**. All connected layers influence each other bidirectionally during energy minimization. Meanwhile, biological synapses are unidirectional, meaning signal flows from a presynaptic to postsynaptic neuron. See Figure 8.2 for explanation.

Because of these differences, we choose the distinct term "hypersynapses" to distinguish them from biological synapses.

Hypersynapse notation conventions

For synapses connecting multiple layers, we subscript with the identifiers of all connected layers. For example:

- E_{xy}: synapse connecting layers X and Y.
- E_{xyz}: synapse connecting layers X, Y, and Z.

- $E_{xyz\ldots}$: synapses connecting more than three layers are possible, but rare.

However, synapses can also connect a layer to itself. To avoid confusion with neuron layer energy E_x, we use curly brackets for synaptic self-connections. For example, $E_{\{x\}}$ represents the interaction energy of a synapse that connects layer X to itself.

Because almost every interaction energy is parameterized in some way, we generally omit Ξ from the notation in subsequent sections when it is not central to the discussion.

The undirected nature of hypersynapses fundamentally distinguishes AM from traditional neural networks. Whereas feedforward networks follow a directed computational graph with clear input-to-output flow, AMs have no inherent concept of "forward" or "backward" directions. All connected layers influence each other bidirectionally during energy minimization, with information propagating from deeper layers to shallower layers as readily as the other way around. Figure 8.1 illustrates this bidirectional hypergraph view.

Unlike neuron-layer energies, the interaction energies of hypersynapses are only constrained to be differentiable scalar functions of activations. Such interaction energies are admissible in the energy-descent proof, but convergence still depends on the total energy and neuron activations satisfying the boundedness and convexity assumptions above.[1] The interaction energy of a synapse may choose to introduce its own non-linearities beyond those handled by the neuron layers. When this occurs, the energy minimization dynamics must compute gradients through these "synaptic non-linearities", unlike the case where all non-linearities are abstracted into the neuron-layer Lagrangians.

8.4 Energy Descent Dynamics

The central result is that the dynamical equations in Equation (8.2) decrease the global energy of the network in Equation (8.1). To demonstrate this, consider the total time derivative of the energy:

$$\frac{dE_{\text{total}}}{dt} = \sum_{\ell=1}^{L} \frac{\partial E_{\text{total}}}{\partial \hat{\mathbf{x}}_\ell} \frac{\partial \hat{\mathbf{x}}_\ell}{\partial \mathbf{x}_\ell} \frac{d\mathbf{x}_\ell}{dt} = -\sum_{\ell=1}^{L} \tau_\ell \frac{d\mathbf{x}_\ell}{dt} \frac{\partial^2 \mathcal{L}_x}{\partial \mathbf{x}_\ell \partial \mathbf{x}_\ell} \frac{d\mathbf{x}_\ell}{dt} \leq 0. \tag{8.5}$$

[1] Some energies could be more meaningful than others.

Here, we expressed the partial derivative of the energy with respect to the activations through the velocity of the neuron's internal states in Equation (8.2). The Hessian matrix $\frac{\partial^2 \mathcal{L}_x}{\partial \mathbf{x}_\ell \partial \mathbf{x}_\ell}$ has size equal to the number of neurons in layer ℓ by the number of neurons in layer ℓ. As long as this matrix is positive semi-definite, a property resulting from the convexity of the Lagrangian, the total energy of the network is guaranteed to either decrease or stay constant. The energy cannot increase.

Additionally, if the energy of the network is bounded from below, the dynamics in Equation (8.2) are guaranteed to lead trajectories to fixed manifolds corresponding to local minima of the energy. If the fixed manifolds have zero dimension, they are fixed *point attractors*, and the velocity field will vanish once the network arrives at the local minimum. This corresponds to Hessians being strictly positive definite. Alternatively, if the Lagrangians have zero modes, resulting in zero eigenvalues of the Hessian matrices, the network may converge to fixed manifolds of one or more dimensions, but the velocity fields may stay non-zero while the network's state moves along that manifold.

8.5 Locality and Physical Computation

The energy descent result is especially useful because global energy minimization can be implemented through local neuron-hypersynapse interactions. This is the property needed for physical, analog, neuromorphic, or distributed implementations: *each neuron only needs the currents generated by the hypersynapses that touch it*, while the system as a whole still descends a single global energy. In this sense, HAMUX gives a modular design language for architectures whose computation is both globally coordinated and locally executable.

The locality follows directly from the decomposition of the total energy. For any neuron ℓ, the derivative of the full energy with respect to that neuron's activation is

$$\frac{\partial E_{\text{total}}}{\partial \hat{\mathbf{x}}_\ell} = \frac{\partial E_\ell^{\text{neuron}}}{\partial \hat{\mathbf{x}}_\ell} + \sum_{s \in \mathtt{N}(\ell)} \frac{\partial E_s^{\text{synapse}}}{\partial \hat{\mathbf{x}}_\ell}. \tag{8.6}$$

All non-incident neuron and hypersynapse components vanish from this derivative by virtue of the "composition of modular energies" structure of the hypergraph. Thus, the full global gradient with respect to a neuron's activation equals the partial gradient of the local components that touch that neuron.

The same construction explains why the local update avoids differentiating through non-linearities when those non-linearities are captured by neuron Lagrangians. The vector

field in Equation (8.2) is written in terms of gradients with respect to activations, while the non-linearity enters the energy-descent proof through the Hessian $\nabla^2 \mathcal{L}_\ell(\mathbf{x}_\ell)$:

$$\frac{dE_{\text{total}}}{dt} = -\sum_{\ell=1}^{L} \tau_\ell \frac{d\mathbf{x}_\ell^\intercal}{dt} \nabla^2 \mathcal{L}_\ell(\mathbf{x}_\ell) \frac{d\mathbf{x}_\ell}{dt} \leq 0. \tag{8.7}$$

If this Hessian is positive semi-definite, as it is for convex Lagrangians, the local dynamics cannot increase the total energy. The non-linearity therefore certifies descent through a positive semi-definite metric rather than appearing as a differentiated operation inside every synaptic update.

8.6 Implementing AMs using HAMUX

We have established how the computational graph is built and the rules for how neuron layers and hypersynapses are constructed. We now discuss how the above mathematical framework can be used to recreate some commonly used AM models.

8.6.1 A Minimal Bipartite AM

A minimal bipartite AM provides the simplest example of how the HAMUX primitives recover both classical Hopfield Networks and DenseAMs. It consists of a visible neuron, a hidden neuron, and a single synapse connecting them. We call these neurons "visible" and "hidden" to match the standard language of neural-network architectures. The visible neuron has internal state $\mathbf{v}$, activation $\hat{\mathbf{v}}$, and Lagrangian $\mathcal{L}_v$; the hidden neuron has internal state $\mathbf{h}$, activation $\hat{\mathbf{h}}$, and Lagrangian $\mathcal{L}_h$. Their component energies are

$$\begin{aligned} E_v^{\text{neuron}} &= \hat{\mathbf{v}}^\intercal \mathbf{v} - \mathcal{L}_v(\mathbf{v}), \\ E_h^{\text{neuron}} &= \hat{\mathbf{h}}^\intercal \mathbf{h} - \mathcal{L}_h(\mathbf{h}), \\ E_{vh}^{\text{synapse}} &= -\hat{\mathbf{h}}^\intercal \boldsymbol{W} \hat{\mathbf{v}}. \end{aligned} \tag{8.8}$$

The total energy is the sum of these component energies:

$$E_{\text{total}} = E_v^{\text{neuron}} + E_h^{\text{neuron}} + E_{vh}^{\text{synapse}}. \tag{8.9}$$

Applying the local update rule from Equation (8.2) to the two neuron layers gives

$$\begin{cases} \tau_v \dfrac{d\mathbf{v}}{dt} & = -\nabla_{\hat{\mathbf{v}}} E_{\text{total}} = \boldsymbol{W}^\intercal \hat{\mathbf{h}} - \mathbf{v}, \\ \tau_h \dfrac{d\mathbf{h}}{dt} & = -\nabla_{\hat{\mathbf{h}}} E_{\text{total}} = \boldsymbol{W} \hat{\mathbf{v}} - \mathbf{h}. \end{cases} \tag{8.10}$$

Thus $\boldsymbol{W}^\intercal \hat{\mathbf{h}}$ and $\boldsymbol{W}\hat{\mathbf{v}}$ are the input currents to the visible and hidden neurons, while $-\mathbf{v}$ and $-\mathbf{h}$ are the exponential-decay terms contributed by the neuron energies. Under suitable choices of the hidden-neuron activation $\hat{\mathbf{h}}$, this two-neuron sandbox recovers the classical Hopfield Network and its dense modern variants [9, 10, 75]. The exercises below take this two-layer construction as their starting point and show how to recover both the classical Hopfield Network and a discrete-variable DenseAM.

Exercise 8.1: Recovering the classical Hopfield Network

Problem Starting from the minimal bipartite AM in Subsection 8.6.1, show how the classical Hopfield Network appears when the hidden neuron is linear and relaxes faster than the visible neuron.

Solution

Choose the hidden-neuron activation to be the identity by setting

$$\mathcal{L}_h(\mathbf{h}) = \frac{1}{2}\|\mathbf{h}\|^2, \qquad \text{so that} \qquad \hat{\mathbf{h}} = \nabla \mathcal{L}_h(\mathbf{h}) = \mathbf{h}. \tag{8.11}$$

Let the rows of $\boldsymbol{W}$ store M memory patterns with normalization $W_{\mu i} = \xi_i^\mu / \sqrt{N}$. In the fast-hidden limit $\tau_h \to 0$, the hidden update in Equation (8.10) reaches its fixed point:

$$\mathbf{h}^\star(\hat{\mathbf{v}}; \boldsymbol{W}) = \hat{\mathbf{h}}^\star(\hat{\mathbf{v}}; \boldsymbol{W}) = \boldsymbol{W}\hat{\mathbf{v}}. \tag{8.12}$$

Substituting this fixed point into the visible update gives

$$\tau_v \frac{d\mathbf{v}}{dt} = \boldsymbol{W}^\intercal \boldsymbol{W}\hat{\mathbf{v}} - \mathbf{v} = \boldsymbol{J}\hat{\mathbf{v}} - \mathbf{v}, \qquad J_{ij} = \frac{1}{N}\sum_{\mu=1}^{M} \xi_i^\mu \xi_j^\mu. \tag{8.13}$$

Thus, eliminating the hidden neuron recovers the Hebbian synaptic matrix of the classical Hopfield Network. The corresponding effective energy is obtained by

substituting $\mathbf{h}^\star = \boldsymbol{W}\hat{\mathbf{v}}$ into the two-layer energy:

$$E_{\text{eff}} = E_v^{\text{neuron}} - \frac{1}{2}\hat{\mathbf{v}}^\intercal \boldsymbol{J}\hat{\mathbf{v}}. \tag{8.14}$$

Finally, choose a saturating visible activation $\hat{v}_i = \tanh(\beta v_i)$ and take the limit $\beta \to \infty$, so that $\hat{v}_i = \text{Sign}(v_i) = \sigma_i$. In this hard-threshold limit, the visible neuron energy E_v^{neuron} vanishes because $\frac{1}{\beta}\log\cosh(\beta v_i) \to |v_i|$ and $v_i\,\text{Sign}(v_i) = |v_i|$. Discretizing the visible update then yields the binary Hopfield update:

$$\sigma_i^{(t+1)} = \text{Sign}\left(\sum_{j=1}^{N} J_{ij}\sigma_j^{(t)}\right). \tag{8.15}$$

Exercise 8.2: Designing the energy for a custom DenseAM

Problem Consider a DenseAM model consisting of D neurons with activation function $\hat{x}_i = \tanh(\beta x_i)$. Design the synaptic energy and the global energy to recreate the discrete-variable DenseAM in the limit $\beta \to \infty$.

Solution

First, define the Lagrangian for this network so that its partial derivative gives the desired activation:

$$\mathcal{L} = \frac{1}{\beta}\sum_{i=1}^{D}\log\Big(\cosh(\beta x_i)\Big), \qquad \text{resulting in} \qquad \hat{x}_i = \tanh(\beta x_i). \tag{8.16}$$

The synapse connects the neuron layer to itself, and its synaptic energy is given by

$$E^{\text{synapse}} = -\sum_{\mu=1}^{K} F\Big(\sum_{j=1}^{D}\xi_j^\mu \hat{x}_j\Big). \tag{8.17}$$

The total energy of the network is

$$\begin{aligned} E^{\text{total}} &= E^{\text{neuron}} + E^{\text{synapse}} \\ &= \Big[\sum_{i=1}^{D} \hat{x}_i x_i - \mathcal{L}\Big] - \sum_{\mu=1}^{K} F\Big(\sum_{j=1}^{D} \xi_j^\mu \hat{x}_j\Big) \\ &= \sum_{i=1}^{D} \Big[\tanh(\beta x_i) x_i - \frac{1}{\beta}\log\Big(\cosh(\beta x_i)\Big)\Big] - \sum_{\mu=1}^{K} F\Big(\sum_{j=1}^{D} \xi_j^\mu \hat{x}_j\Big). \end{aligned} \tag{8.18}$$

The dynamical update equation in Equation (8.2) is given by

$$\tau \frac{dx_i}{dt} = \sum_{\mu=1}^{K} \xi_i^\mu f\Big(\sum_{j=1}^{D} \xi_j^\mu \hat{x}_j\Big) - x_i, \tag{8.19}$$

where $f := F'$ is the derivative of the DenseAM's *separation function* F. Now, discretize time. Set $\tau = 1$ and write the above equation in finite differences, with $dt = 1$:

$$\frac{x_i^{t+1} - x_i^t}{dt} = \sum_{\mu=1}^{K} \xi_i^\mu f\Big(\sum_{j=1}^{D} \xi_j^\mu \hat{x}_j^t\Big) - x_i^t. \tag{8.20}$$

This leads to

$$x_i^{t+1} = \sum_{\mu=1}^{K} \xi_i^\mu f\Big(\sum_{j=1}^{D} \xi_j^\mu \hat{x}_j^t\Big). \tag{8.21}$$

Finally, express everything through the activations $\hat{x}_i$ and take the limit $\beta \to \infty$. In this limit, $\hat{x}_i = \text{Sign}(x_i) = \sigma_i$ and the neuron energy vanishes, resulting in the total energy

$$E^{\text{total}} = -\sum_{\mu=1}^{K} F\Big(\sum_{j=1}^{D} \xi_j^\mu \sigma_j^t\Big). \tag{8.22}$$

Acting with the $\text{Sign}(\cdot)$ function on both sides of Equation (8.21) yields the discrete DenseAM update:

$$\sigma_i^{(t+1)} = \text{Sign}\left[\sum_{\mu=1}^{K} \xi_i^\mu f\Big(\sum_{j=1}^{D} \xi_j^\mu \sigma_j^{(t)}\Big)\right]. \tag{8.23}$$

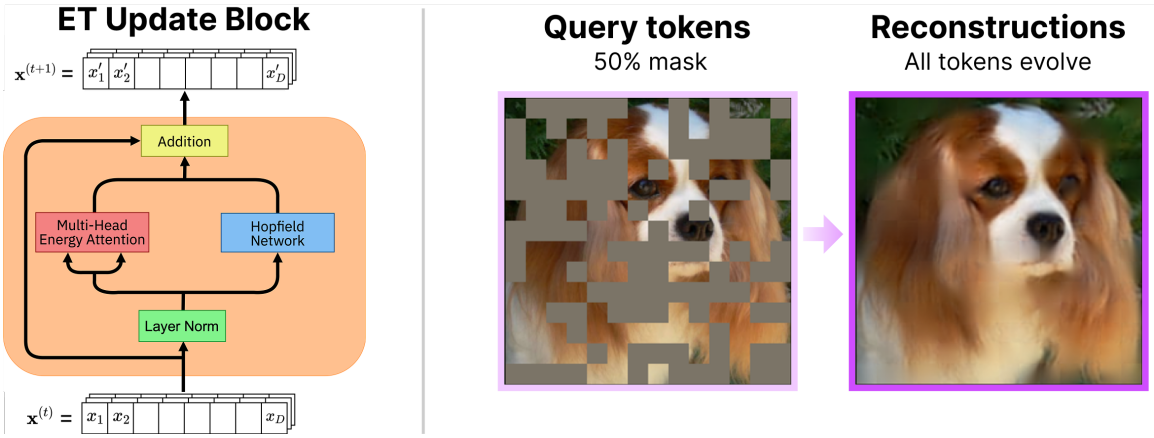

Figure 8.3: ENERGY TRANSFORMER (ET, Chapter 3) describes an energy-based AM whose gradient looks like a transformer block. From the HAMUX perspective, ET combines a LayerNorm neuron with two hypersynapses: an attention energy and a Hopfield Network energy.

8.6.2 Energy Transformer

The ENERGY TRANSFORMER (ET) is the central example showing that HAMUX can express a modern transformer-style architecture [11]. Chapter 3 develops ET as a full architecture for masked-token prediction — this section shows how ET can be written in HAMUX decomposed energies, where `LayerNorm` is a neuron activation, and both attention and the Hopfield network are hypersynapse energies. We repeat equations in this section as necessary for clarity of exposition.

An ET token is a dynamic neuron state $\mathbf{x}_A \in \mathbb{R}^D$, where $A = 1, \dots, N$ indexes tokens and $i = 1, \dots, D$ indexes vector elements. The token activation is a layer-normalized representation $\hat{\mathbf{x}}_A$. ET then couples these token neurons through attention and Hopfield Network hypersynapses.

LayerNorm as a Neuron Lagrangian

LayerNorm is the neuron activation used by ET. For a token state $\mathbf{x}$, the activation is

$$\hat{x}_i = \gamma \frac{x_i - \bar{x}}{\sqrt{\frac{1}{D}\sum_j \left(x_j - \bar{x}\right)^2 + \varepsilon}} + \delta_i, \qquad \text{where} \qquad \bar{x} = \frac{1}{D}\sum_{k=1}^{D} x_k. \tag{8.24}$$

The scalar γ and the vector elements δ_i are learnable parameters, and ε is a small regularization constant. Following the general recipe of HAMUX, this operation can be viewed as a

neuron activation derived as a partial derivative of the Lagrangian function:

$$\mathcal{L}(\mathbf{x}) = D\gamma\sqrt{\frac{1}{D}\sum_j \left(x_j - \bar{x}\right)^2 + \varepsilon} + \sum_j \delta_j x_j, \qquad \text{so that} \qquad \hat{x}_i = \frac{\partial \mathcal{L}(\mathbf{x})}{\partial x_i}. \tag{8.25}$$

See [75, 76, 77] for discussion of this property.

Attention as a Hypersynapse Energy

Energy attention is a hypersynapse that couples token activations through query-key alignments. Below, index $\alpha = 1, \ldots, Y$ denotes elements of the attention feature space, and index $h = 1, \ldots, H$ denotes different heads. The attention hypersynapse is

$$E^{\text{ATT}} = -\frac{1}{\beta}\sum_{h=1}^{H}\sum_{C=1}^{N} \log\left(\sum_{B\neq C} \exp\left(\beta A_{hBC}\right)\right). \tag{8.26}$$

The attention matrix A_{hBC} is computed from query and key tensors as follows:

$$\begin{aligned} A_{hBC} &= \sum_\alpha K_{\alpha hB} Q_{\alpha hC}, \qquad && \mathbf{A} \in \mathbb{R}^{H\times N\times N}, \\ K_{\alpha hB} &= \sum_j W^K_{\alpha hj}\hat{x}_{jB}, \qquad && \mathbf{K} \in \mathbb{R}^{Y\times H\times N}, \\ Q_{\alpha hC} &= \sum_j W^Q_{\alpha hj}\hat{x}_{jC}, \qquad && \mathbf{Q} \in \mathbb{R}^{Y\times H\times N}. \end{aligned} \tag{8.27}$$

The tensors $\mathbf{W}^K \in \mathbb{R}^{Y\times H\times D}$ and $\mathbf{W}^Q \in \mathbb{R}^{Y\times H\times D}$ are learnable parameters. From the perspective of HAMUX, Equation (8.26) is the energy of a hypersynapse that mixes token neurons. Its negative gradient provides the attention-like current that updates each token state.

Hopfield Network as a Hypersynapse Energy

The Hopfield Network module is a second hypersynapse that acts independently on each token activation. Its energy is

$$E^{\text{HN}} = -\sum_{B=1}^{N}\sum_{\mu=1}^{K} G\Big(\sum_{j=1}^{D}\xi_{\mu j}\hat{x}_{jB}\Big), \qquad \boldsymbol{\xi} \in \mathbb{R}^{K\times D}. \tag{8.28}$$

Here, $\xi_{\mu j}$ is a set of learnable weights, or memories in the Hopfield Network, and $G(\cdot)$ is an integral of the activation function $r(\cdot)$, so that $G(\cdot)' = r(\cdot)$. This formula is identical to the energy in Equation (8.17). Depending on the choice of the activation function, this step can be viewed either as a classical continuous Hopfield Network [10], if the activation function grows slowly, or as a Dense Associative Memory [26, 75], if the activation function is sharply peaked around the memories. Examples of slowly growing and sharply peaked activations include $r(\cdot) =$ `ReLU` and $r(\cdot) =$ `power` or `softmax`, respectively. The HN sub-block is analogous to the feedforward MLP step in a conventional transformer block, but requires the weights of the projection from token space to hidden-neuron space to be the same, as a transposed matrix, as the weights of the subsequent projection from hidden space to token space. Thus, the HN module here is an MLP with shared weights that is *applied recurrently*. The energy contribution of this hypersynapse is low when token activations align with rows of $\boldsymbol{\xi}$, which represent memories.

Dynamics of Token Updates

ET composes these pieces by summing the neuron energy, attention hypersynapse energy, and Hopfield Network hypersynapse energy. For token states x_{iA} and activations $\hat{x}_{iA}$, the total energy is

$$\begin{aligned} E_{\text{total}} &= E^{\text{neuron}} + \sum_{s \in \{\text{ATT, HN}\}} E_s^{\text{synapse}} \\ &= \Big[\sum_{A=1}^{N} \sum_{i=1}^{D} x_{iA} \hat{x}_{iA} - \sum_{A=1}^{N} \mathcal{L}(\mathbf{x}_A) \Big] + E^{\text{ATT}} + E^{\text{HN}} \\ &\approx E^{\text{ATT}} + E^{\text{HN}} + O(\varepsilon). \end{aligned} \tag{8.29}$$

When the parameter ε in the LayerNorm Lagrangian is small, the neuron energy contributes only the regularization term $O(\varepsilon)$. In this limit, the total HAMUX energy is approximately the sum of E^{ATT} and E^{HN}:

$$\tau \frac{dx_{iA}}{dt} = -\frac{\partial E_{\text{total}}}{\partial \hat{x}_{iA}}, \qquad \text{where} \qquad E_{\text{total}} = E^{\text{ATT}} + E^{\text{HN}}. \tag{8.30}$$

Here, x_{iA} is the token representation, which is the input and output from the ET block, and $\hat{x}_{iA}$ is its layer-normalized version. The attention energy is low when token queries align with compatible keys, and the Hopfield Network energy is low when token activations align

with memory slots. Equation (8.30) therefore expresses the ET block as a HAMUX energy descent system rather than as a directed sequence of transformer operations.

8.6.3 Neuron-Astrocyte Associative Memory

Neuron-Astrocyte Associative Memory is a biological example where neurons, synapses, and astrocyte processes are all dynamical variables [188]. Even this form of AM is expressible in the HAMUX framework. The model begins from the observation that astrocytes modulate synapses through tripartite synapses, and that communication among astrocyte processes can make distant synapses interact through an additional dynamical layer. In the energy-based limit studied by Kozachkov et al. [188], these coupled neuron-synapse-astrocyte dynamics admit a global Lyapunov function and can behave as a high-capacity AM.

From the HAMUX perspective, the model is naturally represented as three neuron layers and three interaction hypersynapses. The neural states x_i have activations $\hat{x}_i$ derived from a neural Lagrangian $\mathcal{L}^{[n]}$, the synaptic states s_{ij} are now a dynamical state like neurons and have their own activations $\hat{s}_{ij}$ derived from a synaptic Lagrangian $\mathcal{L}^{[s]}$, and the astrocyte-process dynamic states p_{ij} have activations $\hat{p}_{ij}$ derived from an astrocyte-process Lagrangian $\mathcal{L}^{[p]}$. We write both s_{ij} and p_{ij} with pair indices under the fully indexed pair model, where each synaptic variable has a corresponding astrocyte-process variable for the same neuron pair. The public Neuron-Astrocyte Associative Memory tutorial instantiates these activations as tanh nonlinearities, $\hat{x}_i = \tanh(\beta_n x_i)$, $\hat{s}_{ij} = \tanh(\beta_s s_{ij})$, and $\hat{p}_{ij} = \tanh(\beta_p p_{ij})$. Equivalently, each neuron layer can use the same Lagrangian form, with separate inverse-temperature parameters for neural, synaptic, and astrocyte-process states:

$$\begin{aligned}
\mathcal{L}^{[n]}(\mathbf{x}) &= \frac{1}{\beta_n} \sum_i \log\left(\cosh(\beta_n x_i)\right), \\
\mathcal{L}^{[s]}(\mathbf{s}) &= \frac{1}{\beta_s} \sum_{ij} \log\left(\cosh(\beta_s s_{ij})\right), \\
\mathcal{L}^{[p]}(\mathbf{p}) &= \frac{1}{\beta_p} \sum_{ij} \log\left(\cosh(\beta_p p_{ij})\right).
\end{aligned} \tag{8.31}$$

The parameters β_n, β_s, and β_p may differ. The neuron energies are

$$\begin{aligned} E^{[n]} &= \lambda \left(\sum_i x_i \hat{x}_i - \mathcal{L}^{[n]} \right), \\ E^{[s]} &= \frac{\alpha}{2} \left(\sum_{ij} s_{ij} \hat{s}_{ij} - \mathcal{L}^{[s]} \right), \\ E^{[p]} &= \frac{\gamma}{2} \left(\sum_{ij} p_{ij} \hat{p}_{ij} - \mathcal{L}^{[p]} \right). \end{aligned} \tag{8.32}$$

The interaction hypersynapses are *exclusively functions of the activations* of each dynamical state, not the raw internal states.

$$E^{[ns]} = -\frac{1}{2} \sum_{ij} \hat{s}_{ij} \hat{x}_i \hat{x}_j, \qquad E^{[ps]} = -\frac{1}{2} \sum_{ij} \hat{p}_{ij} \hat{s}_{ij}, \qquad E^{[pp]} = -\frac{1}{4} \sum_{ijkl} T_{ijkl} \hat{p}_{ij} \hat{p}_{kl}. \tag{8.33}$$

Thus, the total energy can be written as

$$E_{\text{total}} = E^{[n]} + E^{[s]} + E^{[p]} + E^{[ns]} + E^{[ps]} + E^{[pp]}. \tag{8.34}$$

This energy-based form assumes the symmetry conditions required by Kozachkov et al., including symmetric neuron-synapse interactions and astrocyte-process coupling satisfying $T_{ijkl} = T_{klij}$. The $E^{[pp]}$ term is the HAMUX hypersynapse that encodes communication among astrocyte processes, and after eliminating the synaptic and astrocytic variables the model yields an effective quartic DenseAM-like energy.

The corresponding update equations appear as negative partial gradients of the total energy with respect to the activations of each dynamical state:

$$\begin{cases} \tau_n \dfrac{dx_i}{dt} = -\dfrac{\partial E_{\text{total}}}{\partial \hat{x}_i} = -\lambda x_i + \displaystyle\sum_j \hat{s}_{ij} \hat{x}_j, \\ \tau_s \dfrac{ds_{ij}}{dt} = -2 \dfrac{\partial E_{\text{total}}}{\partial \hat{s}_{ij}} = -\alpha s_{ij} + \hat{x}_i \hat{x}_j + \hat{p}_{ij}, \\ \tau_p \dfrac{dp_{ij}}{dt} = -2 \dfrac{\partial E_{\text{total}}}{\partial \hat{p}_{ij}} = -\gamma p_{ij} + \displaystyle\sum_{kl} T_{ijkl} \hat{p}_{kl} + \hat{s}_{ij}. \end{cases} \tag{8.35}$$

This example shows that HAMUX can express biologically motivated AMs in this energy-based form by treating non-neural variables as dynamical neuron layers and their couplings

as reusable hypersynapse energies.

8.7 Conclusion

This chapter presents HAMUX as a compositional language for energy-based AMs. The core idea is that AM architectures can be written as sums of neuron energies and hypersynapse energies, where neurons carry dynamic states, hypersynapses encode interactions, and inference is the relaxation of the total energy. This view keeps the energy-minimization structure explicit while making it easier to compare architectures that otherwise appear to use different notation or design principles.

The main benefit of this abstraction is that it separates the design choices of an AM into reusable pieces. To build or analyze an AM, one chooses which variables should be dynamic, what non-linearities those variables should use, which interactions should connect them, and what energy should be minimized during inference. In this way, classical Hopfield Networks, DenseAMs, Energy Transformers, and even Neuron-Astrocyte Networks, can be understood as different compositions of the same basic primitives.

Part IV
CONCLUSIONS

CHAPTER 9
CONCLUSION

This dissertation shows that *memory can be a design language for the next generation of AI architectures*, reviving energy-based associative memories (AMs) into a practical foundation for modern AI by uniting a historical theory of neural computation with technical advances in both modern AI and classical ML. The form of computational memory studied here treats inference as iterative error correction: corrupted or incomplete states are moved through an energy landscape toward more coherent states. From this perspective, memory retrieval, denoising, and generation are not separate phenomena, but related forms of energy-based computation.

The chapters of this thesis develop AMs into an alternative to purely feedforward deep learning. Part I shows that recent advances in transformers and diffusion models can inspire competitive AM architectures: ENERGY TRANSFORMER casts image inpainting, graph anomaly detection, and graph classification as energy descent, NRGPT extends this view to causal language modeling, and MEMORY IN PLAIN SIGHT formally connects generative denoising in diffusion models to memory retrieval in AMs. Part II extends AMs using classical ML methods, showing that kernels can equip AMs with compression and creativity: DRDAM uses random features to compress DenseAMs, while LSRDAM uses the KDE-optimal Epanechnikov kernel to create emergent memories without sacrificing exact retrieval. Part III organizes these advances into a compositional design language: HAMUX turns AMs into a cohesive framework of hierarchical architectures built from operations familiar to deep learning. Under the framework's convexity and boundedness assumptions, HAMUX computational graphs come with guarantees around Lyapunov stability, local computation, and simple differentiation properties that make AMs attractive models of physical computation.

9.1 Research Contributions

This thesis makes AMs *useful* for modern AI by showing how AMs can be used to design practical modern architectures, reinterpret generative models, compress and extend memory systems with classical machine-learning tools, and build a broader research program around

energy-based memory.

Groundbreaking algorithms that unlock practical utility for memory

- Pioneered novel algorithmic capabilities to make generative memories practical by ***uniting AMs and kernel theory*** from classical ML. For example, DRDAM equips dense memory with ***pattern compression*** capabilities and a ***one-shot memorization*** learning rule that avoids expensive optimization (Chapter 6). Meanwhile, LSRDAM solves the simultaneous memorization and generalization tradeoff of dense memories by leveraging theoretically optimal kernels (Chapter 7). LSRDAM received a ***Spotlight Paper*** award at NeurIPS (top 3% of 21k+ submissions) and an ***Oral Poster*** at the ICCV'25 workshop on Memory and Vision.

- Introduced AM algorithms that exemplify principled design for ***efficient architectures***, with ET using an order-of-magnitude ***fewer parameters*** than comparable transformers and emphasizing ***reusable weights*** and ***adaptable compute*** in its forward pass (Chapter 3). This architecture was awarded a hardware gift worth ∼$60k by NVIDIA's Academic Grant Program for advancing Robotics and Edge AI.

Fundamental theory to elevate physical computation to modern intelligent systems

- Developed a grounded theory for ***large, fixed-point computational systems***. Specifically, HAMUX describes a cohesive framework for constructing dynamical architectures with convergence guarantees (Chapter 8), while ET and NRGPT show that those architectures can scale to the performance and flexibility expected from large vision and language models (Chapters 3 and 4). ET has been covered in both ***Nature Reviews*** and ***Quanta Magazine***, and was a ***Spotlight Paper*** at the NeurIPS'23 workshop on Associative Memories & Hopfield Networks. HAMUX was a ***Spotlight Paper*** at the NeurIPS'22 workshop on Deep Learning & Differential Equations.

- Established a formal connection between denoising generation and memory recall, advancing a ***human-centric and physically grounded*** theory of memorization and creativity in generative models (Chapter 5). This work has been featured in two ***Quanta Magazine*** articles on the **physics of AI** and **diffusion models and creativity**.

Unified research agenda for accelerating innovations in generative memory

- This thesis brings together a suite of works that bridges performant AI architectures and physics by showing how transformer models, diffusion denoising, compressed memories, and hierarchical memories can all be described through energy-based memory dynamics. In particular, HAMUX establishes an explicit ***blueprint for scaling*** memory architectures using reusable components under physical computation constraints (Chapter 8). My research has ***catalyzed momentum*** for a vibrant AM community through ***two full tutorials*** and ***four dedicated workshops*** across flagship AI conferences such as ICML, AAAI, and NeurIPS. ET has been presented at over 100 conferences and lectures, including Los Alamos National Labs, Harvard CMSA, and the Aspen Center for Physics.

9.2 Impact

My research is making a significant impact on the research community and beyond:

- ET (Chapter 3) was **prominently featured in a *Nature Reviews*** article examining the renaissance of energy-based associative memory in AI, and has been presented at over **100 invited lectures**, including prestigious academic institutions such as Los Alamos National Labs, the Aspen Center for Physics, and Harvard CMSA. Both ET (Chapter 3) and MEMORY IN PLAIN SIGHT (Chapter 5) were **featured in a Quanta Magazine** cover story on the enduring legacy and transformative potential of Hopfield Networks.

- My research on memory has directly convened **over 1,500** students and researchers worldwide at flagship AI conferences, including **4 dedicated workshops** on AMs at NeurIPS, ICLR, and ICCV and **2 full tutorials** at ICML and AAAI.

- My work has been recognized by spotlight awards across top-tier AI venues: LSR-DAM (Chapter 7) received the distinguished **Spotlight Paper Award** (top 3% of >21,000 submissions) at NeurIPS, the largest and most prestigious AI conference. ET was a **Spotlight Paper** at the NeurIPS'23 workshop on Associative Memory & Hopfield Networks, while HAMUX received a **Spotlight Paper** at the NeurIPS'22 workshop on Deep Learning & Differential Equations.

- My research on efficient AM algorithms was awarded a hardware gift worth ∼$60k by **NVIDIA's Academic Grant Program** for advancing Robotics and Edge AI.

9.3 Open Challenges & Research Directions

The results in this thesis make AMs substantially more expressive than the classical models that motivated them. Still, this thesis is only the beginning, and several problems remain before AMs can become a preferred architecture class for deployed systems.

These open problems are easiest to see as the missing pieces of a complete AM pipeline across architectures, training, inference, and applications. This thesis primarily tackles how we can *scale the architectures* of AMs. However, a complete AM pipeline also needs *inference* procedures that retrieve or sample memories efficiently and diversely, *training rules* that write information into energy landscapes, *interpretability tools* that expose the geometry of those landscapes, *applications* where fixed-point computation is a core advantage, and *hardware* that can implement relaxation directly. These all remain part of a larger research program toward reimagining the next generation of AI.

9.3.1 Architectures and Scaling

HAMUX provides guidelines for composing AMs, but we still need larger architectures that make those rules consequential among the current landscape of models. Future work should scale AM systems to larger models, longer contexts, richer modalities, and deeper hierarchies with meaningful latent variables.

There is also considerable theoretical opportunity in this area, since the energy function of AMs gives us a tool to study scaling laws [170] from the perspective of *memory capacity* used to precisely study the capacity of classical Hopfield Networks and DenseAMs [9, 25, 26, 27]. Recent efforts have started connecting associative memory to modern scaling laws and transformer performance [172, 173], but they do not yet address the kind of compositional AMs developed in this thesis.

9.3.2 Inference and Sampling

The works in this thesis follow traditional memory retrieval inference, using direct minimization of the energy via deterministic gradient descent. But this is only one of many ways to sample from or do inference with an energy function. Future AM inference should borrow

more aggressively from numerical optimization and modern generative modeling, including adaptive stopping rules, learned inference rates, momentum-like methods, ADAM, and L-BFGS [168, 169, 208]. Conditional generation should also be studied as energy composition or energy guidance, in the same spirit that classifier-free guidance combines conditional and unconditional diffusion scores to steer sampling [209]. In addition, the energy signal gives us a natural way to adaptively allocate compute, where difficult problems can take more descent steps, and easy examples can stop once the energy approaches stabilization.

There are also specific opportunities to improve the inference of the ET-like architectures proposed in this work. ET and NRGPT show that transformer-style computation can be written as recurrent optimization using efficient parameter sharing, but the best results exchange parameter compression for additional FLOPs in either depth recurrence or model width. The goal is to make these computational update steps cheaper and better matched to the geometry of the energy landscape.

9.3.3 Training and Learning

We train the architectures in this thesis using standard backpropagation through an unfolded inference pass, a process called *backpropagation through time* in recurrent neural networks [210, 211]. This is a reasonable first step because it is the closest approximation to how deep transformers are trained, but the energy function of AMs affords us many additional ways to encode memories. For AMs to outcompete traditional generative modeling approaches, they need learning rules that store information directly and more efficiently than current backpropagation training. Score-based training, denoising objectives, contrastive energy-based objectives, and equilibrium-matching methods are natural candidates [42, 46, 212, 208]. Recent work has begun to connect these ideas back to AM architectures: ET can be trained directly with an equilibrium-matching objective [213], the fixed points of convergent ET models can be trained with equilibrium propagation [214], and DenseAMs can be trained using an energy-minimization rule inspired by Hopfield's Hebbian learning [215]. These early results are encouraging because they respect the energy structure of AMs rather than treating that structure as a constraint imposed after ordinary backpropagation.

9.3.4 Evaluation and Interpretation

Explicit energies should make AMs easier to inspect, but inspection still requires metrics and tools. Recent work on energy-based geometry suggests that energy landscapes can be

used to define low-energy embeddings, approximate geodesics, and score-based Riemannian metrics for navigating data manifolds [216, 217, 218]. The analogous goal for AMs is somewhat like a debugger for a memory network, where the tool enables a way to see not only what an AM retrieves, but to visualize the structure of the energy landscape that made the retrieval likely to begin with.

9.3.5 Applications

This dissertation evaluates AMs in one of two ways: for the transformer-like AMs, we evaluated on image completion, graph classification, and language modeling downstream tasks; for more pure memory architectures, we studied memory capacity and attractor interference as is typically done in the study of traditional Hopfield Networks. However, there is a broader opportunity to use AMs wherever solutions can be represented as equilibria. Reasoning, planning, constraint satisfaction, scientific inverse problems, anomaly detection, and multimodal completion all contain settings where a system must reconcile partial or inconsistent information until it reaches a solution that satisfies the constraints.

Recent work on energy-based and equilibrium reasoning makes this direction especially timely, especially works on iterative reasoning through energy diffusion [219], equilibrium reasoners [220], and attractor models for language and reasoning [221]. In the AM language of this thesis, those attractors are memories: stable solutions retrieved by dynamics. The open question is whether pure AMs with Lyapunov energies can improve on this kind of looped reasoning using inspectable and controllable architectures.

9.3.6 Specialized Hardware

AMs are special EBMs whose analog computations are *directly designed* to run on physical hardware using energy relaxation. This is shown most clearly by the local dynamics emphasized by HAMUX § 8.5, where each neuron only needs the currents generated by the hypersynapses that touch it to minimize the whole system's global energy. This represents the most exciting future direction for AMs, where the algorithms meet the hardware they were designed for: analog, neuromorphic, optical, and other non-von-Neumann hardware designed around statistical physics.

Recent developments suggest that this direction is gaining momentum. Hardware implementations of classic Hopfield Networks and Ising machines can solve energy-minimization problems using physical dynamics [222], and recent work proposes direct implementa-

tions of DenseAM inference using RC circuits, crossbar arrays, and amplifiers [223]. The moonshot challenge is to move beyond demonstrations of physical AM circuits toward applications where large AMs on neuromorphic hardware produce measurable gains in latency and energy use over alternative algorithms on traditional hardware.

9.4 Final Conclusion

This thesis began as a curious observation that modern AI looked like the iterative error correction formalized by the Hopfield Network all those decades ago. That seed has grown into a belief that *memory can serve as a design language for the next generation of AI architectures*. In this thesis, I have shown how energy-based associative memory (AM) can be used to derive explicit memory dynamics for **modern** AI architectures, how kernels can be used as **general** tools for solving memory compression and creativity, and how to **unify** all AMs into a compositional design language — providing the tooling for a paradigm where interpretable, efficient, and robust AI emerges from physical computation built around memory, energy, and dynamics.

In AMs, memory is more than a storage bank of past observations. It is a way to make AI tangible, inspectable, and dynamically integrated into the same physical world that gave rise to our own intelligence. I envision AI systems whose computational trajectories can be inspected, whose weights can be visualized for the patterns they store, whose energy landscapes can be compressed and composed with other memories, and whose inference can live directly in physical circuits that relax toward meaningful predictions and generations at the speed of light. This thesis is an initial step toward that vision, carrying forward Hopfield's hope that ideas of useful computation performed by physical systems can inspire a new generation of AI [224].

Appendices

APPENDIX A
ENERGY TRANSFORMER SUPPLEMENTARY MATERIAL

This appendix collects the supplementary material from the original Energy Transformer work so that the dissertation is self-contained. It includes experimental details, ablations, and parameter comparisons for the Energy Transformer chapter.

A.1 Details of Training on ImageNet

We trained the ET network on a masked-image completion task on the ImageNet-1k (IN1K) dataset. We treat all images in IN1K as images of shape 224×224 that are normalized according to standard IN1K practices (mean 0, variance 1 on the channel dimension) and use data augmentations provided by the popular `timm` library [225] (See Table A.1). Following the conventional ViT pipeline [74], we split these images into non-overlapping patches of 16x16 RGB pixels which are then projected with a single affine encoder into the token dimension D for a total of 196 encoded tokens per image. We proceed to randomly and uniformly assign 100 of these tokens as "occluded" which are the only tokens considered by the loss function. "Occluded" tokens are designated as follows: of the 100 tokens, 90 tokens are replaced with a learnable MASK token of dimension D and 10 we leave untouched (which we find important for the HN to learn meaningful patch representations). To all tokens we then add a distinct learnable position bias.

These tokens are then passed to our Energy Transformer block which we recur for T steps (the "depth" of the model in conventional transformers). At each step, the feedback signal (the sum of the energy gradients from our attention block and HN block) is subtracted from our original token representation with a scalar step size $\alpha = \frac{dt}{\tau}$ which we treat as a non-learnable hyperparameter in our experiments. The token representations after T steps are passed to a simple linear decoder (consisting of a layer norm and an affine transformation) to project our representations back into the image plane. We then use the standard MSE Loss between the original pixels and reconstructed pixels for only the 100 occluded patches. We allow self attention as in the following formula for the energy of multiheaded attention

$$E^{\text{ATT}} = \sum_h -\frac{1}{\beta} \sum_C \log \left(\sum_B \exp \left(\beta \sum_\alpha K_{\alpha h B} \, Q_{\alpha h C} \right) \right) \tag{A.1}$$

We give details of our architectural choices in Table A.1. In the Energy Transformer chapter we present our Energy Transformer with a configuration similar to the standard base transformer configuration (e.g., token dimension 768, 12 heads each with $Y = 64$, softmax's $\beta = \frac{1}{\sqrt{Y}}, \dots$), with several considerations learned from the qualitative image evaluations:

- The $\frac{dt}{\tau}$ (step size) of 1 implicitly used in the traditional transformer noticeably degrades our ability to smoothly descend the energy function. We find that a step size of 0.1 provides a smoother descent down the energy function and benefits the image reconstruction quality.

- We observe that our MSE loss must include some subset of un-occluded patches in order for the HN to learn meaningful filters.

- Values of β in the energy attention that are too high prevent our model from training. This is possibly due to vanishing gradients in our attention operation from a `softmax` operation that is too spiky.

- Without gradient clipping, our model fails to train at the learning rates we tried higher than 1e-4. We observe that gradient clipping not only helps our model train faster at the trainable learning rates, it also allows us to train at higher learning rates.

Our architecture and experiments for the image reconstruction task were written in JAX [226] using Flax [227]. This engineering choice means that our architecture definitions are quite lightweight, as we can define the desired energy function of the ET and use JAX's autograd to automatically calculate the desired update.

A.1.1 Exploring the Hopfield Memories

A distinctive aspect of our network is that any variable that has a vector index i of tokens can be mapped into the data domain by applying the decoder network to this variable. This makes it possible to inspect all the weights in the model. For instance, the concept of "memories" is crucial to understanding how Hopfield networks function. The memories within the HN module represent the building blocks of *all possible image patches* in our data domain, where an encoded image patch is a superposition of a subset of memories. The complete set of memory vectors from the HN module is shown in Figure A.1. The same

Table A.1: Hyperparameter, architecture, and data augmentation choices for ET model during ImageNet-1k masked training experiments. Data augmentations are listed as parameters passed to the equivalent `timm` dataloader functionality.

Training	
batch_size	768
epochs	100
lr	5e-4
warmup_epochs	2
start & end lr	5e-7
b1, b2 (ADAM)	0.9, 0.99
weight_decay	0.05
grad_clipping	1.

Architecture	
token_dim	768
num_heads	12
head_dim	64
β	1/8
train_betas	No
step size α	0.1
depth	12
hidden_dim HN	3072
bias in HN	None
bias in ATT	None
bias in LNORM	Yes

Data Augmentation	
random_erase	None
horizontal_flip	0.5
vertical_flip	0
color_jitter	0.4
scale	(0.08, 1)
ratio	(3/4, 4/3)
auto_augment	None

analysis can be applied to the weights of the ET-attention module. In Figure A.2, we show all the weights from this module mapped into the image plane.

A.1.2 Positional Bias Correlations

The relationships between our position embeddings exhibit similar behavior to the position correlations of the original ViT in that they are highly susceptible to choices of the hyperparameters (Figure 10 of [74]). In particular, we consider the effect of weight decay and the β parameter that serves as the inverse temperature of the attention operation (see Equation (3.3)). The lower the temperature (i.e., the higher the value of β), the spikier the softmax distribution. By using a lower β, we encourage the attention energy to distribute its positional embeddings across a wider range of patches in the model.

A.2 Details of ET training on Anomaly Detection Task

Graph anomaly detection refers to the process of detecting outliers that deviate significantly from the majority of the samples. Neural network based methods are very popular due to their capability of learning sophisticated data representations. DOMINANT [82] utilizes an auto-encoder framework, using a GCN as an encoder and two decoders for structural reconstruction and attribute reconstruction. ALARM [83] aggregates the encoder information from multiple view of the node attributes. Another study [228], propose a novel loss function to train graph neural networks for anomaly-detectable node representations. In

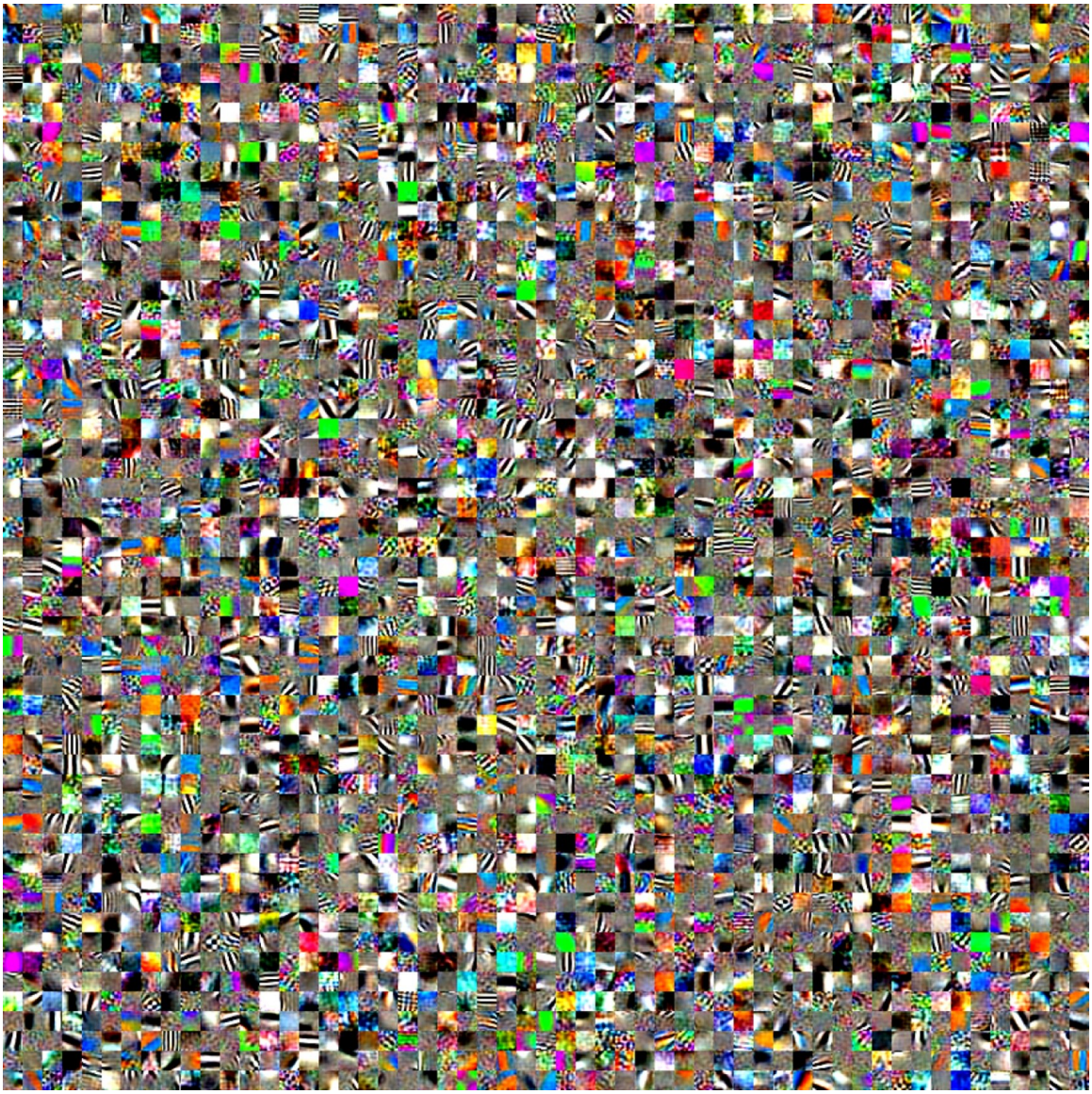

Figure A.1: Visualizing a randomly selected 3025 patch memories of the 3072 learned by weight matrix in the Hopfield Network module (HN) of our model. These memories are vectors of the same dimensions D as the patch tokens, stored as rows in the weight matrix ξ. Each image patch is visualized using the model's trained decoder.

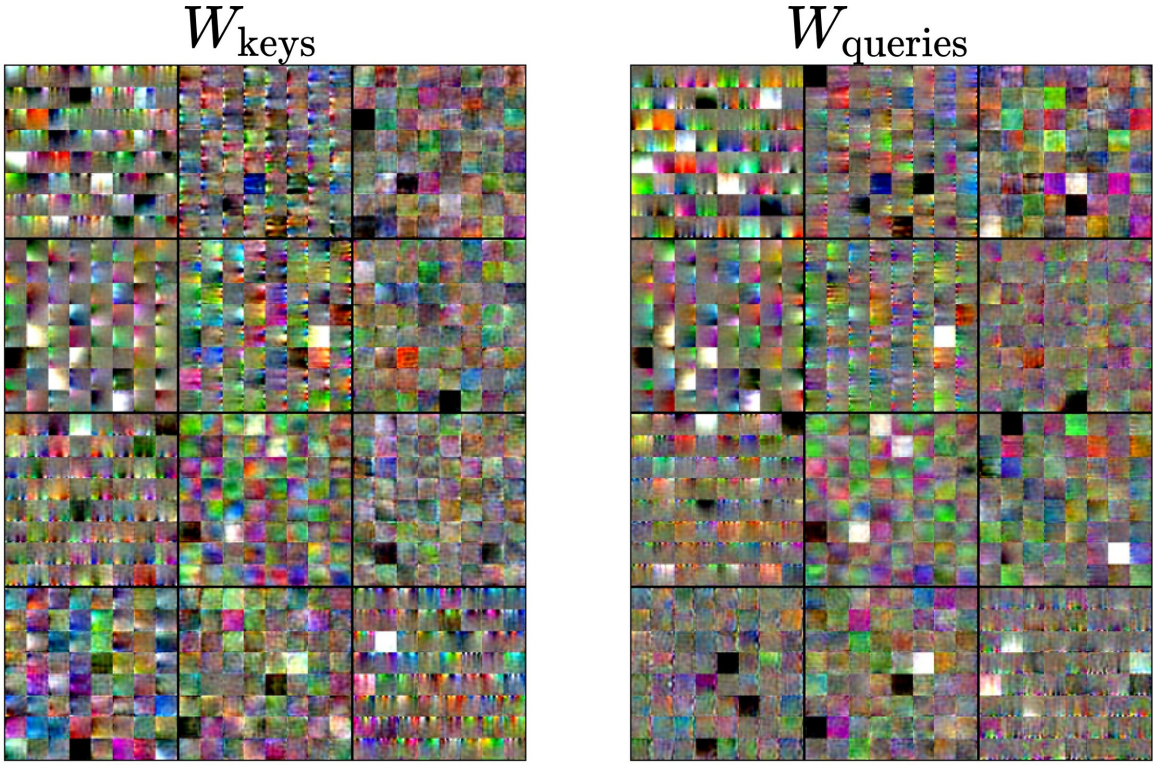

Figure A.2: Visualizing the token dimension of the "key" and "query" matrices of the attention as image patches. Each head is represented as a cell on the 4×3 grid above. We use the trained decoder of our model to visualize each weight.

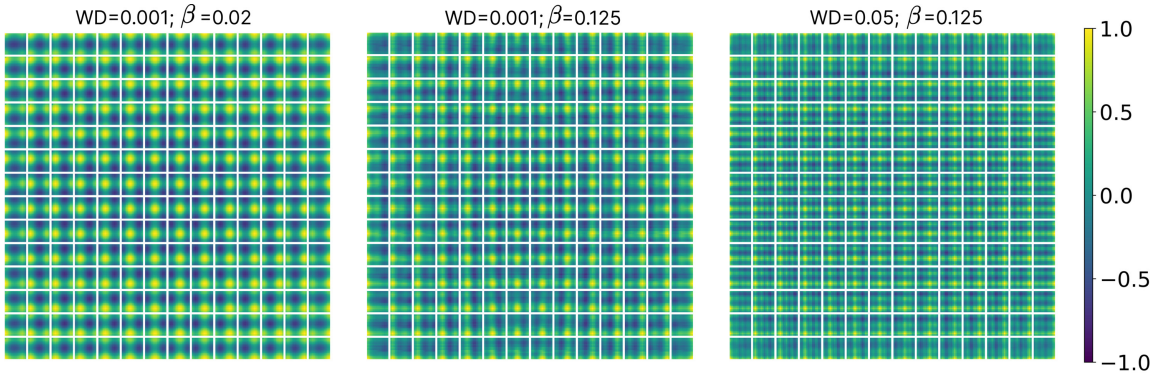

Figure A.3: The cosine similarity between position biases of patches when the ET model is trained under different hyperparameter choices for β (inverse temperature of the attention energy) and weight decay. Our ET sees a trend where smoother correlations are observed with smaller β and weight decay.

[229] generative adversarial learning is used to detect anomaly nodes where a novel layer is designed to learn the anomaly-aware node representation. Recently, [80] pointed out that anomalies can lead to the "rightshift" of the spectral energy distribution – the spectral energy concentrates more on the high frequencies. They designed a filter that can better handle this phenomenon. We propose a new anomaly detection model from the perspective of energy-based Associative Memory (pattern matching), which does not have the over-smoothing problem often faced by GCNs, and has better model interpretability (outliers should be far from the common pattern). We also notice that Modern Hopfield Networks have been used before for node classification, link prediction, and graph coarsening tasks [230].

A.2.1 Detailed Model Structure for the Graph Anomaly Detection

First, we compute the features that are passed to our energy-based transformer. Each node's features $\mathbf{y}_A \in R^F$ are mapped into the token space $\mathbf{x}_A \in R^D$, using a linear projection $\mathbf{E}$. Learnable positional embeddings λ_A are added to this token at $t = 1$,

$$\mathbf{x}_A^{t=1} = \mathbf{E}\mathbf{y}_A + \lambda_A \tag{A.2}$$

At each time step the input to the ET-block is layer normalized:

$$\mathbf{g}_A^t = \text{LayerNorm}(\mathbf{x}_A^t) \tag{A.3}$$

Let $\mathbf{W}^Q \in R^{Y\times H\times D}$ and $\mathbf{W}^K \in R^{Y\times H\times D}$ be the query and key weight matrices, respectively. Here Y is the projection dimension in the attention operation, H is the number of heads. We define

$$\begin{aligned} K_{\alpha hB} &= \sum_j W^K_{\alpha hj}\, g_{jB} \\ Q_{\alpha hC} &= \sum_j W^Q_{\alpha hj}\, g_{jC} \end{aligned} \tag{A.4}$$

If we let h indicate the index of the head, we have

$$\Delta x^t_{iA} = \sum_{C\in\mathcal{N}_A}\sum_{h,\alpha}\Big[W^Q_{\alpha hi}\, K_{\alpha hC}\, \omega_{CA} + W^K_{\alpha hi}\, Q_{\alpha hC}\, \omega_{AC}\Big] + \sum_\mu \xi_{\mu i}\, r\Big(\sum_j \xi_{\mu j} g_{jA}\Big) \tag{A.5}$$

where

$$\omega_{CA} = \underset{C}{\text{softmax}}\Big(\beta \sum_{\gamma} K_{\gamma hC}\, Q_{\gamma hA}\Big) \tag{A.6}$$

Here β controls the temperature of the softmax, $\mathcal{N}_A$ stands for the neighbors of node A —a set of all the nodes connected to node A, r is the ReLU function. Restriction of the attention operation to the neighborhood of a given node is similar to that used in the Graph Attention Networks (GAT), see [231]. Finally, we have residual connection (which is a natural consequence of the discretized differential equation dynamics)

$$\mathbf{x}_A^{t+1} = \mathbf{x}_A^t + \mathbf{\Delta x}_A^t \tag{A.7}$$

Intuitively, the first term in Equation (A.5) describes the influence (attention score) of the neighbor nodes with respect to the target node, the second term describes the influence of the target node with respect to each of its neighbor, and the third term is the contribution of the Hopfield Network module. It can be shown that the forward pass of our energy-based transformer layer minimizes the following energy function:

$$E = -\frac{1}{\beta}\sum_{C}\sum_{h}\log\left(\sum_{B\in\mathcal{N}_C}\exp\left(\beta\sum_{\alpha}K_{\alpha hB}\,Q_{\alpha hC}\right)\right) - \frac{1}{2}\sum_{C,\mu}G\Big(\sum_{j}\xi_{\mu j}\,g_{jC}\Big) \tag{A.8}$$

This energy function will decrease as the forward pass progresses until it reaches a local minimum.

After T iterations when the retrieval is stable, we have the final representation for each node $\mathbf{g}_A^{\text{final}}$ as

$$\mathbf{g}_A^{\text{final}} = \mathbf{g}_A^{t=1} \,||\, \mathbf{g}_A^{t=T} \tag{A.9}$$

where $||$ is the concatenation sign. Following [80], we treat anomaly detection as semi-supervised learning task in this work. The node representation $\mathbf{g}_A^{\text{final}}$ is fed to another MLP with the sigmoid function to compute the abnormal probability p_A, weighted log-likelihood is then used to train the network. The loss function is as follow:

$$\text{Loss} = \sum_{A}\Big[\omega\; l_A \log(p_A) + (1 - l_A)\log(1 - p_A)\Big] \tag{A.10}$$

where ω is the ratio of normal labels (l_A = 0) to anomaly labels (l_A = 1).

A.2.2 Experimental Details

We train all models for 100 epochs using the Adam optimizer with a learning rate of 0.001, and use the model with the best Macro-F1 on the validation set for reporting the final results on the test set. Following [80], we use training ratios 1% and 40% respectively (randomly select 1% and 40% nodes of the dataset to train the model, and use the remaining nodes for the validation and testing). These remaining nodes are split 1:2 for validation:testing. The statistics of the datasets are listed in Table A.2. For the four datasets used in the experiments, Amazon and Yelp datasets can be obtained from the DGL library, T-Finance and T-Social can be obtained from [80]. We report the average performance of 5 runs on the test datasets.

Table A.2: Summary of all the datasets.

Dataset	$\|V\|$	$\|E\|$	Anomaly(%)	Features
Amazon	11944	4398392	6.87%	25
Yelp	45954	3846979	14.53%	32
T-Finance	39357	21222543	4.58%	10
T-Social	5781065	73105508	3.01%	10

The hyperparameters of our model are tuned based on the validation set, selecting the best parameters within 100 epochs. To speedup the training process, for the large graph datasets T-Finance and T-Social, we sample a different subgraph to train for each epoch (subgraphs have 5% of the nodes with respect to the whole training data). The hyperparameters include the number of hidden dimensions in ET-attention Y, the number of neurons K in the hidden layer within the Hopfield Network Module, the number of time iterations T, and the number of heads H. The weights are learned via backpropagation, which includes embedding projection $\mathbf{E}$, positional embedding λ_A, softmax inverse temperature parameter β, ET-attention weight tensors $\mathbf{W}^Q$ and $\mathbf{W}^K$. The optimal hyperparameters used in Table 3.1 are reported in Table A.3. The last row in that table summarizes the range of the hyperparameter search that was performed in our experiments. In general, we have observed that for small datasets (Yelp, Amazon, T-Finance) a 1 or 2 applications of our network is sufficient for achieving strong results, for larger datasets (T-Social) more iterations (3) are necessary. For even bigger dataset (ImageNet) our network needs about 12 iterations.

Table A.3: Hyperparameters choice of our method on all the datasets.

Dataset	Y	K	T	H
Amazon (40%)	128	640	1	2
Amazon (1%)	64	128	1	1
Yelp (40%)	128	256	1	1
Yelp (1%)	128	256	1	1
T-Finance (40%)	128	256	1	3
T-Finance (1%)	128	256	1	1
T-Social (40%)	128	256	3	3
T-Social (1%)	128	256	3	3
Range of hyperparameters	{64, 128, 256}	{2Y, 3Y, 4Y, 5Y}	{1,2,3}	{1,2,3}

A.3 Graph Classification with ET

Recently, there have been attempts of leveraging transformers in the graph domain, but only certain key modules, such as feature aggregation, are replaced in GNN variants by the softmax attention [232]. However, there remains an interesting and open question about suitability of the transformer architecture to model graphs and how to apply it in the graph classification domain. In this section we explain how ET can be used for graph classification.

A.3.1 Details of Graph Classification ET Model

Given a graph $G = (V, \mathbf{A}, \mathcal{E})$, we have the set of nodes $V = \{v_1, v_2, \ldots, v_N\}$, the adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$, and if it exists, the edge feature matrix $\mathcal{E} \in \mathbb{R}^{N \times N \times P'}$ where each edge has P' raw features. Each feature vector $\mathbf{y}_A \in \mathbb{R}^F$ corresponding to a node v_A is first projected to the token space $\mathbf{x}_\mathbf{A} \in \mathbb{R}^D$ via a linear embedding. Then, the CLS token $\mathbf{x}_{\text{CLS}}$ is concatenated to the set of tokens resulting in $\mathbf{X} \in \mathbb{R}^{(N+1) \times D}$ and the positional embedding $\lambda \in \mathbb{R}^{(N+1) \times D}$ is added to it afterwards.

To obtain the positional embedding $\lambda \in \mathbb{R}^{(N+1) \times D}$, the adjacency matrix $\mathbf{A}$ is first padded in the upper left corner with ones resulting in $\tilde{\mathbf{A}} \in \{0, 1\}^{(N+1) \times (N+1)}$ as a means to provide the CLS token full connectivity with all of the nodes in a given graph. Following [93], the top k smallest eigen-vectors $\tilde{\lambda} \in \mathbb{R}^{(N+1) \times k}$ are extracted from the normalized Laplacian matrix $\tilde{\mathbf{L}}$ obtained from $\tilde{\mathbf{A}}$

$$\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}, \tag{A.11}$$

and projected to the token space via a linear embedding to form λ. Note, $\mathbf{I}$ and $\tilde{\mathbf{D}}$ are the

identity matrix and the degree matrix of $\tilde{\mathbf{A}}$.

Meanwhile, the attention in ET is modified to take in $\hat{\mathbf{A}} \in \mathbb{R}^{(N+1)\times(N+1)\times H}$ the parameterized adjacency tensor, which acts as the weighted 'attention mask' that enables the model to consider the graph structural information. To obtain $\hat{\mathbf{A}}$, a 2D-convolutional layer with H filters equals to the number of heads in the attention block, 'SAME' padding, and a stride of one is performed on the outer product of $\mathbf{X}$ to itself. The result is then multiplied with the tensor $\mathbf{A}' \in \mathbb{R}^{(N+1)\times(N+1)\times H}$ element-wise (denoted by $\odot$) via broadcasting

$$\hat{\mathbf{A}} = \text{Conv2D}(\mathbf{X} \otimes \mathbf{X}) \odot \mathbf{A}'. \tag{A.12}$$

The tensor $\mathbf{A}'$ is a linear projection, via a linear embedding layer, of the edge feature matrix $\mathcal{E}$ if it exists, where P' is projected to H dimension; or the padded adjacency matrix $\tilde{\mathbf{A}}$ where $P' = 1$. Altogether, the resulting energy equation is

$$E^{\text{ATT}} = -\frac{1}{\beta} \sum_h \sum_C \log \left(\sum_{B \neq C} \exp \left(\beta \sum_\alpha K_{\alpha hB}\, Q_{\alpha hC} \odot \hat{A}_{hC} \right) \right). \tag{A.13}$$

Moreover, to only consider the energy of edges in G, we set non-connected entries in $\hat{\mathbf{A}}$ to be $-\infty$, such that the gradient of such entries would be zero following the dynamics. Note, if non-connected entries are not ignored, the gradient of such entries would be non-zero and hence, the resultant attention energy might violate some of the graph structural information.

Meanwhile, in this implementation, the overall model consists of S vertically stacked ET blocks, where each block shares the same number of T depth and has its own LayerNorm. Similarly, the token representation $\mathbf{X}^{\ell,\,t}$ at dynamic step t corresponding to a block ℓ is layer-normalized,

$$\mathbf{g}^{\ell,\,t} = \text{LayerNorm}(\mathbf{X}^{\ell,\,t}). \tag{A.14}$$

Keep in mind, $\mathbf{X} = \mathbf{X}^{\ell=1,\,t=1}$ is the initial token representation.

Following the dynamic equations in Equation (3.6), we inject a small amount of noise $\epsilon^{\ell,\,t} \in (0, 1)$, generated from a normal distribution using a standard deviation σ_ϵ and zero mean, into the gradient of energy function to produce $\mathbf{X}^{\ell,\,t+1}$, the new token representation of block ℓ. The premise of this noise injection is to 'robustify' the model and help it escape saddle points of the energy function.

$$\mathbf{X}^{\ell,\,t+1} = \mathbf{X}^{\ell,\,t} - \alpha \nabla_{\mathbf{g}} E^{\ell,\,t} + \sqrt{\alpha}\, \epsilon^{\ell,\,t}, \qquad \epsilon^{\ell,\,t} \sim \mathcal{N}(0, \sigma_\epsilon^2). \tag{A.15}$$

Once stability is reached in the retrieval dynamics of a block ℓ, the final representation $\mathbf{X}^{\ell,\, t=T}$ is then passed on to the next block $\ell + 1$ and the whole process is repeated again. When the final token representation $\mathbf{X}^{\ell=S,\, t=T}$ is computed by the last block S, the resultant CLS token $\mathbf{x}_{\text{CLS}}^{\ell=S,\, t=T} \in \mathbb{R}^{D'}$, extracted from $\mathbf{X}^{\ell=S,\, t=T}$, is utilized as the predictor of the current graph G.

A.3.2 Experimental Evaluation

As mentioned prior, eight datasets from TUDataset [92] are used for experimentation. Based on the collected results in Table 3.2, we observed that the modified ET demonstrates the best performance across all datasets with the exception of MUTAG. The remainder of this section records the dataset statistics and hyperparameters needed to reproduce the TUDataset graph-classification results reported in the main chapter.

A.3.3 Experimental Details

In the graph domain, it is common to concatenate all of the feature vectors of all graphs in a batch together. However, in order for ET to work, we form the batch dimension by separating the feature vectors of all graphs in a given batch and utilize the largest node count to pad all graphs such that they all share the same number of nodes. Additionally, we set a limit on the number of nodes equal to 500, to prevent out-of-memory error. Specifically, if a graph has a node count exceeding the limit, the number of utilized nodes is equal to the limit. Hence, a portion of the graph structural information is ignored as a result. However, it is worth mentioning such a graph is rare in the experimental datasets. Additionally, instead of ignoring the padded entries, they are altered to be sink nodes in a graph, such that they are connected to the actual nodes of the graph but not vice versa. We found this approach to be helpful when it comes to computing the positional embedding and training the model. Additionally, following [93], the sign of the top k eigen-vectors is flipped randomly during training and fixed during evaluation. The whole experiment is implemented using JAX[226], Flax [227], Optax [233], and PyTorch Geometric [234] frameworks.

For the eight datasets of TUDataset, we train all models for 300 epochs using AdamW [235] and Gradient Centralization [236]. The best model is selected based on its performance obtained from the 10-fold cross validation process delineated in [92]. Since the task is classification, all models are trained with the cross-entropy loss function and label smoothing [237], where the smoothing parameter is set to 0.05. Additionally, the cosine-annealing

Table A.4: The statistics and properties of the eight datasets of TUDataset (additional node attributes are indicated by '+' if exist).

Dataset	Graphs	Avg. Nodes	Avg. Edges	Node Attr	Classes
MUTAG	188	17.93	19.79	7	2
ENZYMES	600	32.63	62.14	18 + 3	6
PROTEINS	1113	39.06	72.82	0 + 4	2
DD	1178	284.32	715.66	89	2
NCI1	4110	29.87	32.30	37	2
NCI109	4127	29.68	32.13	38	2
MUTAGENICITY	4337	30.32	30.77	14	2
FRANKENSTEIN	4337	16.90	17.88	780	2

with warm-up learning rate scheduler [238] is utilized, where the initial and end learning rates are both set as $5e-6$ while the peak learning rate is 0.001. The number of warm-up steps is set to 50 epochs while the batch size is 32 for all datasets. We report the average performance of 100 runs on the 10-fold cross validation process with random seeding. The hyperparameters of our model are tuned based on the performance of the cross validation, selecting within 100 epochs. The optimal hyper-parameters are reported in Table A.5 and the statistics of the used datasets are reported in Table A.4. We also include the number of gpu-devices used for training ET.

A.4 Ablation Study for Attention and Hopfield Network Modules

As described in the Energy Transformer chapter, the ET network consists of two modules processing the tokens in parallel: the attention module (ATT) and the Hopfield Network module (HN). The ATT module is responsible for routing the information between the tokens, while the HN module is responsible for reinforcing the token representation to be consistent with the general expectation about the particular data domain. It is interesting to explore the contribution that these two subnetworks produce on the task performed by the network. In this section we ablate the ET architecture by dropping each of the two subnetworks and measuring the impact of the ablation on the performance.

A.4.1 On Graphs

The results on graphs are reported in Table A.6. From this table it is clear that most of the computation is performed by the ATT block on this task, which pools the information about other tokens to the token of interest. When the ATT block is kept, but the HN block is

Table A.5: Hyperparameter and architecture choices for ET during TUDataset experiments.

Training	
batch_size	32
epochs	300
peak lr	1e-3
warmup_epochs	50
initial and ending lr	5e-6
b1, b2 (ADAM)	0.9, 0.99
weight_decay	0.05
grad_clipping	None
num. of gpu devices	2

Architecture	
token_dim	128
num_heads	12
head_dim	64
β	$\frac{1}{\sqrt{64}}$
train_betas	Yes
step size α	0.01
k eigenvalues	15
noise σ_ϵ	0.02
depth	1
block_size	4
kernel_size	[3, 3]
dilation_size	[1, 1]
hidden_dim HN	512
bias in HN	None
bias in ATT	None
bias in LNORM	Yes
num. of params per ET block	262,929
avg. total num. of params	1071294

removed, the network loses 1% or less relative to the full ET (occasional improvements of the ablated model compared to the full ET are within the statistical error bars). In contrast, removing the ATT module and keeping only the HN effectively turns the ET network into an MLP with shared weights that is recurrently applied. In this regime, the network can only use the features of a given node for that node's anomalous status prediction. This results in a more significant drop in performance, which is about 5% on average.

A.4.2 On Images

The ablation results for image reconstruction are shown in Table A.7. Each experiment was trained using the same hyperparameter settings as shown in Table A.1. After training the model on IN1K, we calculate the average MSE on the reconstructed masked tokens for the validation set (using the same 50% masking ratio used for training) across 10 different random seeds for the masking.

We make several conclusions from these ablation studies.

- We gain several insights regarding the use of "self-attention" in our ET (when a token patch query is allowed to consider itself as a key in the attention weights). When both

Table A.6: Ablation study with respect to ATT block and HN Block. Best results are in **bold**.

	Datasets	Split	ATT✓|HN✗	ATT✗|HN✓	full model (Ours)
Macro-F1	Yelp	1%	$62.5_{\pm 0.3}$ ▼	$57.4_{\pm 0.5}$ ▼(-5.6)	$\mathbf{63.0}_{\pm \mathbf{0.6}}$
		40%	$70.6_{\pm 0.5}$ ▼	$71.2_{\pm 0.7}$ ▼	$\mathbf{71.5}_{\pm \mathbf{0.1}}$
	Amazon	1%	$\mathbf{89.5}_{\pm \mathbf{0.9}}$ ▲	$87.4_{\pm 1.0}$ ▼	$89.3_{\pm 0.7}$
		40%	$91.7_{\pm 0.5}$ ▼(-1.1)	$88.7_{\pm 0.3}$ ▼(-4.1)	$\mathbf{92.8}_{\pm \mathbf{0.3}}$
	T-Finance	1%	$84.7_{\pm 1.0}$ ▼	$80.3_{\pm 0.6}$ ▼(-4.8)	$\mathbf{85.1}_{\pm \mathbf{1.0}}$
		40%	$87.4_{\pm 0.7}$ ▼	$82.3_{\pm 0.8}$ ▼(-5.9)	$\mathbf{88.2}_{\pm \mathbf{1.0}}$
	T-Social	1%	$\mathbf{79.8}_{\pm \mathbf{0.6}}$ ▲	$72.7_{\pm 1.0}$ ▼(-6.4)	$79.1_{\pm 0.7}$
		40%	$82.9_{\pm 1.0}$ ▼	$78.6_{\pm 1.2}$ ▼(-4.9)	$\mathbf{83.5}_{\pm \mathbf{0.4}}$
AUC	Yelp	1%	$72.9_{\pm 0.3}$ ▼	$67.4_{\pm 0.7}$ ▼(-5.8)	$\mathbf{73.2}_{\pm \mathbf{0.8}}$
		40%	$83.5_{\pm 0.4}$ ▼(-1.4)	$83.1_{\pm 0.6}$ ▼(-1.8)	$\mathbf{84.9}_{\pm \mathbf{0.3}}$
	Amazon	1%	$90.7_{\pm 0.8}$ ▼	$89.8_{\pm 1.2}$ ▼	$\mathbf{91.9}_{\pm \mathbf{1.0}}$
		40%	$96.8_{\pm 0.6}$ ▼	$95.7_{\pm 0.5}$ ▼(-1.6)	$\mathbf{97.3}_{\pm \mathbf{0.4}}$
	T-Finance	1%	$91.7_{\pm 1.2}$ ▼	$90.2_{\pm 0.8}$ ▼(-2.6)	$\mathbf{92.8}_{\pm \mathbf{1.1}}$
		40%	$94.3_{\pm 2.6}$ ▼	$90.2_{\pm 2.1}$ ▼	$\mathbf{95.0}_{\pm \mathbf{3.0}}$
	T-Social	1%	$\mathbf{92.2}_{\pm \mathbf{0.8}}$ ▲	$86.4_{\pm 0.7}$ ▼(-5.5)	$91.9_{\pm 0.6}$
		40%	$93.1_{\pm 0.8}$ ▼	$88.3_{\pm 1.3}$ ▼(-5.6)	$\mathbf{93.9}_{\pm \mathbf{0.2}}$

self-attention and HN are present (ET-Full+Self), there is no noticeable benefit over ET-Full for a token to attend to itself. In fact, preventing the ATTN energy module from attending to itself slightly improves the performance. However, when the HN is removed (ET-NoHN*), we notice that allowing self-attention (ET-NoHN+Self) outperforms the version that prevents self-attention (ET-NoHN).

- On its own, allowing self-attention (ET-NoHN+Self) in the ATTN module performs nearly as well as the full ET at a fraction of the total parameters. However, MSE is a forgiving metric for blurry reconstructions. While ATTN can capture the global structure of the image quite well, it does so at the expense of image sharpness (Figure 3.4).

- As expected, removal of the ATTN energy module performs the worst, because the HN operates tokenwise and has no way to aggregate token information across the global image without ATTN.

Figure A.4 shows our best performing model (ET-Full) on the qualitative image re-

constructions corresponding to the *largest* errors across IN1K validation images, averaged across all masking seeds. Likewise, Figure A.5 shows the *lowest* errors across IN1K validation images and masking seeds. In general, image reconstructions that require ET to produce sharp, high frequency, and high contrast lines negatively impact MSE performance.

Table A.7: Module ablation tests for image reconstruction task, reporting average IN1K validation MSE on masked tokens after 100 epochs. Reported number of parameters excludes the constant number of parameters in the affine encoder and decoder.

Model	Has ATTN?	Allow self-attn?	Has HN?	**NParams** (in ET block)	**MSE**
ET-Full **(Ours)**	✓	✗	✓	3.7M	$\mathbf{0.306_{\pm 0.10}}$
ET-Full+Self	✓	✓	✓	3.7M	$0.312_{\pm 0.10}$
ET-NoHN+Self	✓	✓	✗	1.3M	$0.343_{\pm 0.10}$
ET-NoHN	✓	✗	✗	1.3M	$0.403_{\pm 0.11}$
ET-NoATT	✗	✗	✓	2.5M	$0.825_{\pm 0.20}$

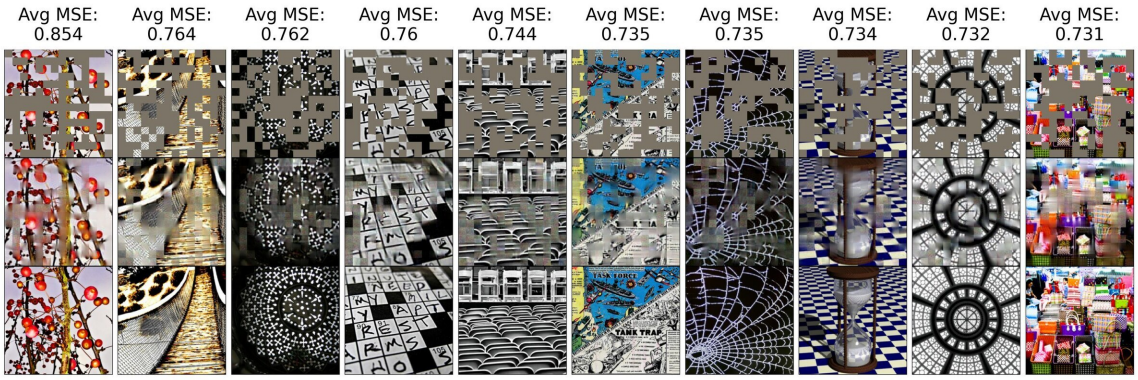

Figure A.4: Reconstruction examples of images with the worst MSE from the IN1k validation set. *Top row:* input images where 50% of the patches are masked with the learned MASK token. *Middle row*: all tokens reconstructed after 12 time steps. *Bottom row*: original images.

A.5 Parameter Comparison

The energy function enforces symmetries in our model, which means ET has fewer parameters than its ViT counterparts. In particular, ET has no "Value Matrix" $\mathbf{W}^V$ in the attention mechanism, and the HN module has only one of the two matrices in the standard MLP of the traditional transformer block. We report these differences in Table A.8. We take the ET configuration used in this work, which has an architecture fully comparable to

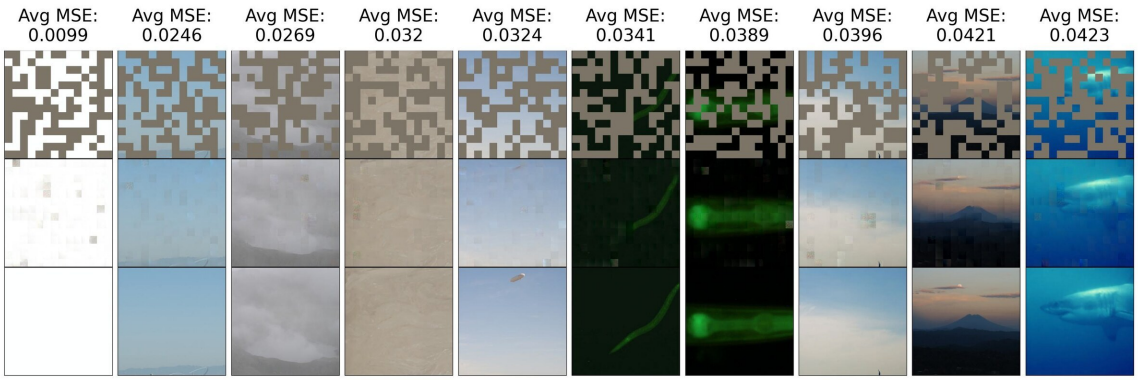

Figure A.5: Reconstruction examples of images with the best (lowest) MSE from the IN1k validation set. *Top row:* input images where 50% of the patches are masked with the learned MASK token. *Middle row*: all tokens reconstructed after 12 time steps. *Bottom row*: original images.

the original ViT-base [74] with `patch_size=16`, and report the number of parameters against ViT-base and an "ALBERT" version of ViT [71] where a single ViT block is shared across layers. We saw no benefit when including biases in ET, so we also exclude the biases from the total parameter count in the configuration of ViT and ALBERT-ViT. We report both the total number of parameters and the number of parameters per transformer block. Furthermore, as a means for a fairer comparison, we reduce the parameters of ViT-Base such that the parameter count is similar to that of ET. We contrast ET and the reduced ViT on the image reconstruction task using PSNR (Peak-Signal-to-Noise-Ratio) and SSIM (Structural-SIMilarity) [239] as the evaluation metrics of the reconstructions. The results are demonstrated in Table A.9, where the performance of ET is very close to that of the reduced ViT, which has a slight advantage in parameter count.

Table A.8: Comparison between the number of parameters in a standard ViT, an ALBERT version of ViT where standard transformer blocks are shared across layers, and our ET. Comparison is done assuming no biases in any operation.

Model	**NParams**		**NParams** (per block)	
ViT-Base	86.28M	▼0.00%	7.08M	▼0.00%
ALBERT ViT-Base	8.41M	▼90.25%	7.08M	▼0.00%
ET	**4.87M**	**▼94.36%**	**3.54M**	**▼50.02%**

Table A.9: Comparison between ET and 'comparable-size' ViT on image reconstruction task. Given ViT-Base, we reduce its parameter count down to a number similar to that of ET for the image domain. The metrics, PSNR (Peak-Signal-to-Noise-Ratio) and SSIM (Structural-SIMilarity), are recorded for the image reconstruction evaluations.

Model	**NParams**		**PSNR**		**SSIM**	
ViT	5.52M	▼0.00%	**22.11**	▼0.00	**0.715**	▼0.00
ET	**4.28M**	▼**22.46%**	21.51	▼0.59	0.681	▼0.03

APPENDIX B
NRGPT SUPPLEMENTARY MATERIAL

This appendix collects the supplementary material from the original NRGPT work so that the dissertation is self-contained. It includes comparisons with related energy-based language models, details of the convergence argument, experimental settings, and generation examples.

B.1 NRGPT Compared to EBT and ET

B.1.1 Distinction from EBT

Both the Energy-Based Transformers (EBT) of [106] and NRGPT are methods that can be used to minimize an explicit energy during autoregressive generation. However, the methods differ in *how* that energy is modeled. Specifically, the EBT paper uses standard, feed-forward transformer architectures with modified attention masks and a standard loss function on predicted tokens to turn the transformer predictions into an energy, whereas we use a novel, causal energy formulation of the transformer block inspired by the Energy Transformer [11].

NRGPT is most comparable to the autoregressive EBT. The EBT paper mentions that "the autoregressive EBT presents greater implementation challenges, primarily due to the potential for information leakage in naïve implementations", and they discuss the design challenges in Sec C.3 of their Appendix, which requires reframing how tokens are processed by the standard transformer architecture. To prevent information leakage, they duplicate a sequence of tokens into *observed* tokens and *predicted* tokens while carefully tuning both the attention masks and the contraction over observed values.

We believe that NRGPT's solution for a causal EBM is more succinct and elegant than EBT's, preventing duplicate tokens and the need for careful tuning of attention masks. Our solution is expressed in Equation (4.3), where E_A describes the energy to predict token A. By restricting the summation of the `logsumexp` to include summation only over previous tokens $B < A$, and then by minimizing the energy w.r.t. token $\mathbf{g}_A$, we are guaranteed to propagate information causally without any of the engineering tricks of EBT.

B.1.2 Distinction from ET

The architecture of ENERGY TRANSFORMER (ET) [11] is more similar to that of NRGPT, but it is fundamentally different. This is because ET is designed for MASKed-token prediction tasks whereas NRGPT is a special energy function designed for *autoregressive token prediction*, a paradigm that is not compatible with the original ET's bidirectional attention design (where attention signal propagates both forward and backwards in time such that past tokens can attend to future tokens). A simple causal mask on ET attention breaks ET's guarantee of monotonic energy minimization.

The key innovation of NRGPT is to model the sequence as a collection of token-wise energies E_A for token index A, rather than a global sequence energy $E = \sum_A E_A$. We discover that tokens still converge even when all tokens minimize their individual energy simultaneously (see Figure 4.2), and we argue that this generalizes the ideas of ET to allow tokens to explore a meaningful energy landscape during causal token generation.

We also theoretically and empirically study the **projection matrix** that is present in all standard transformer attention but that is *noticeably absent* in ET's attention formulation. We interpret this matrix as an "inference rate" matrix $\boldsymbol{\eta}$ in the gradient descent step, where under certain conditions we are guaranteed to maintain token convergence. Including this projection matrix and playing with various choices for E^{FF} of Equation (4.3) makes the NRGPT block *as expressive* as a recurrent, standard GPT block that can arguably go toe-to-toe with similarly configured GPT models on causal language modeling tasks up to the size of OWT (see Table B.3 and Table B.2). These conclusions and experiments were not evident in the original ET paper.

In summary, NRGPT is distinct from ET because:

1. **NRGPT performs causal language modeling by minimizing a per-token energy**. ET was restricted to strict energy minimization of an entire sequence, a paradigm that is not compatible with the parallel, autoregressive language modeling of GPT-style transformers.

2. **NRGPT uses learnable inference rate matrices $\boldsymbol{\eta}$** during token prediction. Meanwhile, ET was restricted to a fixed, scalar gradient descent step which did not allow additional exploration of the energy landscape.

3. **NRGPT explores alternative energy-replacements for the feed-forward (FF)**

MLP module. ET used a single-layer Hopfield Network with energy $G(\boldsymbol{\xi}\mathbf{g}_A)$, which results in the weights of the two layers to be $\boldsymbol{\xi}$ and $\boldsymbol{\xi}^T$. In NRGPT, we explore a more general form E^{FF} (e.g., Equation (4.24)) for the feed-forward module and find improved results on the causal language modeling task.

B.2 Advantages of EBMs

EBMs offer a paradigm for generative modeling that is inherently about optimization, where the model samples a new token from a **transition probability** described by the prior context. Conventional transformers are trained to *implicitly* learn this transition probability, but it is hidden in the architectural design. The key appeal of an EBM (like NRGPT) is that it models the transition probability *explicitly*. This offers several advantages which we hope to explore in future work:

1. **Systematic exploration of the solution space**. An explicit likelihood function enables us to systematically explore the space of solutions in LLMs using well-established methods in optimization and statistical physics, such as alternative gradient descent methods, minimum-energy paths, saddle-point analysis, and metastability. These tools are not available when the energy is implicitly encoded in a deep architecture, and we believe that finding alternative or creative solutions may correspond to exploring different local minima of the energy during inference. It also means that we can now view the forward process of causal LMs like GPT as a formal optimization process, which despite the prevalence of research around *in-context learning* (ICL) [240], is not a widespread belief.

2. **Variable computation using early stopping criteria**. Both the energy and the norm of the energy's gradient can be used as signals to stop model computation "early" for easier problems, or to continue thinking longer for harder ones. This is a primary motivation of other explicit EBM language models like EBT [106], and we discuss this advantage further below.

3. **Model alignment using energy regularizers**. Note that Equation (4.7) of this work shows NRGPT's architecture as the sum of two energies: a token-mixing *attention energy* E^{AT} and a token-wise *feed-forward energy* E^{FF}. Per the precedent of ET [11], these are chosen to be faithful to the original transformer's design. However, any

scalar objective can theoretically be added to NRGPT's block, and we imagine that adding regularizer terms to bias the energy landscape toward favorable solutions (or away from undesirable ones) is a unique benefit of EBMs that is more robust to prompt injection attacks than current LLMs.

Early stopping Adaptive stopping criteria for "early stopping" is an incredible advantage for EBMs over traditional methods in the context of language modeling. If all of NRGPT's tokens $B < A$ are fixed, then the energy of token A is guaranteed to decrease, and early stopping based on energy values and gradient norms is well-defined — the model iteratively improves a token's embedding until the convergence guarantees are met and there is no purpose to "thinking longer". This iterative improvement needs no fine-tuning as it is baked into the model design. However, the constraint of a fixed preceding context has a strong *disadvantage*: doing this would discard the transformer's unique advantage of full parallelism across tokens. For the causal energy formulation with parallel token evolution studied in this work, we are actually not guaranteed to monotonically decrease the energy and energy deltas and gradient norms would be unreliable (see Figure 4.2 for token-wise energy trajectories).

B.3 On FLOP Complexity

NRGPT is parameter efficient compared to standard GPT and recurrent GPT, but how does it compare on the FLOP cost of the model? When we consider the FLOPs/block (per iteration), we discover that NRGPT is anywhere from $1-2\times$ the FLOP cost of an equivalent RecGPT at constant parameters, where the factor of 2 difference appears depending on which FF variant we use (`NRGPT_H_FF1` or `NRGPT_H_FF2W`). We expand on the contributions of our architectural choices below:

- **NRGPT Attention**: For a single head, the attention block of NRGPT costs approximately the same FLOPS as recurrent and normal GPT. In fact, NRGPT uses slightly fewer FLOPs since the key weights $\boldsymbol{J}\mathbf{g}_B$ are used for both the `keys` and the `values` of the attention update (in contrast to the standard attention where distinct values $\boldsymbol{W}^V\mathbf{g}_B$ are computed for each key B). For multiple heads however, NRGPT costs more because our current solution uses $D \times D$ weights J_h *per head*, while GPT attention uses increasingly narrow KQV matrices as you increase the number of heads

(i.e., the dimension of $\boldsymbol{W}^V$ is of shape $D \times (D/H)$, which decreases as H becomes larger). This is a convention that can be easily adapted to NRGPT but we did not explore in this work.

- **NRGPT FF**: This FLOP count for this FF module can vary, but in this work we tested configurations that are always $\sim 2\times$ the FLOP count of a standard transformer's FF of the same width. We discuss the two FF modules studied in this work below:

  1. **FF1**, where $E^{FF} = \|\sigma(\boldsymbol{W}\boldsymbol{g})\|^2$. In ListOps, we found that FF1 competes with Rec-GPT only when the hidden dimension of $\boldsymbol{W}$ was $8D$ instead of the standard $4D$ used in GPT. This makes FF1 a constant-parameter operation, but because the wider weight matrix is used twice it actually doubles the FLOPS cost of NRGPT.

  2. **FF2W**, where $E^{FF} = \boldsymbol{g}^T \boldsymbol{W^2} \sigma(\boldsymbol{W^1}\boldsymbol{g})$. In this config, the update rule $-\boldsymbol{\eta}\nabla E$ leads to two terms, each being a two layer neural net. We keep the $4D$ expansion used by standard transformers and thus the same parameter count, but this leads to a $\sim 2\times$ FLOPS in this module compared to standard transformers.

Thus, though NRGPT is more parameter efficient in general, it seems that parameter savings need to be exchanged for FLOPs. Note that we use pytorch's functional autograd interface to compute gradients through this network which slows down the wall-clock time evaluation compared to manual implementation of the update rule.

B.4 Learning Rate and Preconditioner of the Forward Pass

$$\textbf{RMSNorm:} \quad g_{Ai} = \gamma_i \frac{x_{Ai}}{\sqrt{\frac{1}{D}\sum_i x_{Ai}^2}} = \sqrt{D}\gamma_i \hat{x}_{Ai} \tag{B.1}$$

$$\textbf{LayerNorm:} \quad g_{Ai} = \gamma_i \frac{x_{Ai} - \mathbb{E}[x_A]}{\sqrt{\mathrm{Var}[x_A] + \epsilon}} + \beta_i, \tag{B.2}$$

where x_{Ai} are the components with $A \in 1 \ldots N$ and $i \in 1 \ldots D$.

Proof of Proposition 4.1: Energy Descent. Using chain rule

$$\begin{aligned}\dot{E}_A &= \sum_{B,C} \frac{\mathrm{o}E_A}{\mathrm{o}\boldsymbol{g}_B} \frac{\mathrm{o}\boldsymbol{g}_B}{\mathrm{o}\boldsymbol{x}_C} \dot{\boldsymbol{x}}_C \\ &= -\sum_{B,C} \frac{\mathrm{o}E_A}{\mathrm{o}\boldsymbol{g}_B} \frac{\mathrm{o}\boldsymbol{g}_B}{\mathrm{o}\boldsymbol{x}_C} \boldsymbol{\eta} \frac{\mathrm{o}E_C}{\mathrm{o}\boldsymbol{g}_C}\end{aligned} \tag{B.3}$$

Defining $\Gamma = \mathrm{diag}(\gamma)$, the Jacobian of g becomes

$$\frac{\partial \boldsymbol{g}_{Ai}}{\partial \boldsymbol{x}_{B\rho}} = \Gamma_{ij} \frac{\delta_{AB}}{r_A} \left(\delta_{j\rho} - y_{Aj} y_{A\rho}\right), \tag{B.4}$$

where

$$\textbf{RMSNorm:} \qquad r_A = \|\boldsymbol{x}_A\| / \sqrt{D}, \qquad y_A = \boldsymbol{x}_A / r_A \tag{B.5}$$

$$\textbf{LayerNorm:} \qquad r_A = \sqrt{\mathrm{Var}[\boldsymbol{x}_A] + \epsilon}, \qquad y_A = (\boldsymbol{x}_A - \boldsymbol{\mu}_A)/r_A. \tag{B.6}$$

Since typically $\epsilon = 10^{-5}$ is a small constant, $\|y_A\| \approx 1$ for LayerNorm, and exactly 1 for RMSNorm. Therefore

$$\frac{\partial \boldsymbol{g}_A}{\partial \boldsymbol{x}_A} = \frac{1}{r_A} \Gamma P_A, \qquad P_A = P_A^T, \qquad P_A^2 = P_A + O(\epsilon), \tag{B.7}$$

where the approximate projection matrix P_A is positive semi-definite, with $\hat{y}_A / \|y_A\|$ being its sole null eigenvector. Plugging into Equation (B.3)

$$\dot{E}_A = -\sum_B \frac{1}{r_B} \mathrm{Tr} \frac{\partial E_A}{\partial g_B} \Gamma P_B \boldsymbol{\eta}^T \frac{\partial E_B}{\partial g_B}^T \tag{B.8}$$

Note that when $B > A$, $\mathrm{o}E_A / \mathrm{o}\boldsymbol{g}_B = 0$. Hence, Equation (B.8) can be separated into $A = B$ and $B < A$

$$\dot{E}_A = \sum_{B<A} \mathrm{Tr} \frac{\partial E_A}{\partial \boldsymbol{g}_B} \frac{\mathrm{o}\boldsymbol{g}_B}{\mathrm{o}\boldsymbol{x}_B} \dot{\boldsymbol{x}}_B - \frac{1}{r_A} \mathrm{Tr} \frac{\partial E_A}{\partial \boldsymbol{g}_A} \Gamma P_A \boldsymbol{\eta}^T \frac{\partial E_A}{\partial \boldsymbol{g}_A}^T \tag{B.9}$$

In order for token A to converge as well, we only need to find conditions under which the second term in Equation (B.9) converges. This is because the same conditions would then lead the second term in all $\dot{E}_B$ for $B < A$ to converge. Then, by induction all E_B will

eventually converge. Thus we want just the second term

$$\delta E_A = -\frac{1}{r_A} \operatorname{Tr} \frac{\partial E_A}{\partial \boldsymbol{g}_A} \Gamma P_A \boldsymbol{\eta}^T \frac{\partial E_A}{\partial \boldsymbol{g}_A}^T , \tag{B.10}$$

to satisfy $\delta E_A < 0$,

which can be achieved if the symmetric part of $\Gamma P_A \boldsymbol{\eta}^T$ is p.s.d.. Two simple solutions to this are

$$\boldsymbol{\eta} = c\Gamma, \qquad \text{or} \qquad \boldsymbol{\eta} = M(x) P_A \Gamma, \tag{B.11}$$

where $M(x)$ is an arbitrary p.s.d. matrix and $c > 0$. If we want $\boldsymbol{\eta}$ to be simple weights instead of an x dependent neural network, the solution is $\boldsymbol{\eta} = c\Gamma$. □

Note that Equation (B.10) does not restrict the anti-symmetric part of $\Gamma P_A \boldsymbol{\eta}$. Using $\boldsymbol{\eta} = c\Gamma + B$, the antisymmetric part satisfies $B^T P_A \Gamma = -\Gamma P_A B$. Since P_A is rank $D-1$ for each A, the anti-symmetric part doesn't seem to have an x-independent solution.

The fact that P_A is different for each token severely restricts the form of $\boldsymbol{\eta}$ to get convergence. However, the boundedness of the energies E_A due to boundedness of $\boldsymbol{g}_A$ can tame the dynamics and lead to more choices for $\boldsymbol{\eta}$ yielding convergence. For example, a damped harmonic oscillator has a dynamics in which the state oscillates in damped fashion, but the energy is constantly decreasing.

B.4.1 Evolution of loss

We can always absorb Γ into the weights of AT and FF, so we will assume $\Gamma = I$ from here on. To find $\dot{E}_A$ we need to use chain rule with derivatives w.r.t. all tokens $B < A$.

$$\begin{aligned}
\dot{E}_A &= \sum_B \mathrm{o}_{g_B} E_A \mathrm{o}_B g_B \dot{x}_B \\
&= -\sum_B \mathrm{o}_{g_B} E_A P_B \boldsymbol{\eta} \mathrm{o}_{g_B} E_B \\
&= \sum_{B<A} \mathrm{o}_{g_B} E_A^T P_B \dot{x}_B - \mathrm{o}_{g_A} E_A^T P_A \boldsymbol{\eta} \mathrm{o}_{g_A} E_A,
\end{aligned} \tag{B.12}$$

because of the coupling between different tokens, it is not clear whether $\dot{E}_A$ is negative or not. Yet, due to the causality of the $B < A$ interaction, token B is not affected by any future token. It follows that if token B dynamics converges, meaning $\dot{x}_B \to 0$ for all $B < A$, we

have

$$\text{if } \dot{x}_B = 0, \quad \forall B < A: \quad \dot{E}_A = \left(\frac{\mathrm{o}E_A}{\mathrm{o}g_A}\right)^T P_A \boldsymbol{\eta} \frac{\mathrm{o}E_A}{\mathrm{o}g_A} = -\dot{x}_A^T \boldsymbol{\eta}^{-1} P_A \dot{x}_A, \tag{B.13}$$

where the last part assumes $\boldsymbol{\eta}$ is invertible. Hence, token A will converge if $P_A\boldsymbol{\eta}$ is p.s.d. Since P_A is a projection away from x_A and $P_A\boldsymbol{\eta}$ should be p.s.d. for all A, it follows that on the data manifold $\boldsymbol{\eta}$ can only be a constant times identity. If the data is on a low-rank subspace, $\boldsymbol{\eta}$ can be arbitrary on the orthogonal subspace. More formally, let $P_\parallel$ represent projection onto $S_\parallel = \mathrm{Span}\{x_A | \forall A\}$, meaning $x_A^T P_\parallel x_A = \|x_A\|^2$. Let $P_\perp$ be the projection to the subspace orthogonal to all data x_A, meaning $x_A^T P_\perp x_A = 0$, and so $\|P_\perp P_\parallel\| = 0$. The $\boldsymbol{\eta}_\parallel = P_\parallel \boldsymbol{\eta} P_\parallel = cP_\parallel$ for some $c > 0$, while $\boldsymbol{\eta}_\perp = P_\perp \boldsymbol{\eta} P_\perp$ is an arbitrary p.s.d. matrix with null space including $S_\parallel$.

By induction, we can show that the $\boldsymbol{\eta}$ above leads to convergence of all tokens. For the first token $A = 1$, there are no past tokens, so convergence requires

$$\dot{E}_1 = -\left(\frac{\mathrm{o}E_1}{\mathrm{o}g_1}\right)^T P_1 \boldsymbol{\eta} \frac{\mathrm{o}E_1}{\mathrm{o}g_1} < 0. \tag{B.14}$$

For the second token, when the first token converges, $\dot{x}_1 = 0$, resulting in the same condition for $\dot{E}_2$. Assume $\boldsymbol{\eta} = I$ and introduce the shorthand $E_{A;B} = \mathrm{o}E_A/\mathrm{o}g_B$, and the Mahalanobis norm $\|z\|_M^2 = z^T M z$. For any A

$$\begin{aligned}
E_{A;A} P_A \dot{x}_A &= -\|E_{A;A}\|_{P_A}^2 = -\|\dot{x}_A\|_{P_A}^2 \\
\dot{E}_A &= E_{A;A} P_A \dot{x}_A + \sum_{B<A} E_{A;B} P_B \dot{x}_B \\
&= -\|\dot{x}_A\|_{P_A}^2 + \sum_{B<A} E_{A;B} P_B \dot{x}_B \qquad \text{(B.15)} \\
&\leq \sum_{B<A} E_{A;B} P_B \dot{x}_B. \qquad \text{(B.16)}
\end{aligned}$$

Since E_A are bounded from below (shown next), $\dot{E}_A < 0$ should eventually lead to convergence. For E_1, there are no past tokens and $\dot{E}_1 < 0$ always, resulting in convergence.

B.4.2 Boundedness of the energy.

First, observe that $E_A = E_A^{AT} + E_A^{FF}$ is bounded from below, given some assumptions about FF. For AT, since $g_B = \delta x_A / \|\delta x_A\| + \beta = \hat{y} + \beta$, we have

$$\begin{aligned} \|g_A\|^2 &= 1 + 2\beta \cdot \hat{y}_A & \Rightarrow 1 - 2\|\beta\| \leq \|g_A\|^2 \leq 1 + 2\|\beta\| \\ g_A^T \boldsymbol{J} g_B &= \hat{y}_A^T \boldsymbol{J} \hat{y}_B + \beta \cdot (\boldsymbol{J}\hat{y}_B + \boldsymbol{J}^T \hat{y}_A) \\ -(1 + 2\|\beta\|)\|\boldsymbol{J}\|_2 &\leq g_A^T \boldsymbol{J} g_B & \leq (1 + 2\|\beta\|)\|\boldsymbol{J}\|_2, \end{aligned} \tag{B.17}$$

where $\|\boldsymbol{J}\|_2$ is the spectral norm of $\boldsymbol{J}$ equal to the largest singular value of $\boldsymbol{J}$. Therefore, using monotonicity of log and exp

$$\begin{aligned} E_A^{AT} &= \log \sum_{B<A} \exp[g_A^T \boldsymbol{J} g_B] \leq \log(A \max_{B<A}\{\exp[g_A^T \boldsymbol{J} g_B]\}) \\ &\leq \log A + \max_{B<A}\{g_A^T \boldsymbol{J} g_B\} \leq \log A + (1 + 2\|\beta\|)\|\boldsymbol{J}\|_2, \end{aligned} \tag{B.18}$$

and similarly for the lower bound resulting in

$$-(1 + 2\|\beta\|)\|\boldsymbol{J}\|_2 \leq |E_A^{AT} - \log A| \leq (1 + 2\|\beta\|)\|\boldsymbol{J}\|_2. \tag{B.19}$$

For FF, we need to assume a form for the energy. First, considering $E^{FF}(g_A) = -\|\sigma(W g_A)\|^2$ with activation σ being Lipschitz, as in ReLU or GeLU. The Lipschitz condition means

$$\|\sigma(x) - \sigma(y)\| \leq L\|x - y\|, \tag{B.20}$$

for some Lipschitz constant $L > 0$. Setting $y = 0$ and $x = W g_A$ we get

$$\|\sigma(W g_A) - \sigma(0)\| \leq L\|W g_A\| \leq L\|W\|_2 \|g_A\| \leq L\|W\|_2(1 + \|\beta\|), \tag{B.21}$$

using triangle inequality $\|a + b\| \leq \|a\| + \|b\|$, with $a = \sigma(W g_A) - \sigma(0)$ and $b = \sigma(0)$ we get

$$\begin{aligned} \|\sigma(W g_A)\| &\leq \|\sigma(W g_A) - \sigma(0)\| + \|\sigma(0)\| \leq L\|W g_A\| + \|\sigma(0)\| \\ &\leq L\|W\|_2(1 + \|\beta\|) + \|\sigma(0)\|, \end{aligned} \tag{B.22}$$

and similarly, using the other side of triangle inequality $\|a\| \geq \|a+b\| - \|b\|$ with $a = \sigma(Wg_A)$ and $b = -\sigma(Wg_A) + \sigma(0)$ we have

$$\|\sigma(Wg_A)\| \geq \|\sigma(0)\| - \|\sigma(Wg_A) - \sigma(0)\| \geq \|\sigma(0)\| - L\|W\|_2(1 + \|\beta\|). \tag{B.23}$$

Therefore

$$-(\|\sigma(0)\| + L\|W\|_2(1 + \|\beta\|))^2 \leq E^{FF}(g_A) \leq -(\|\sigma(0)\| - L\|W\|_2(1 + \|\beta\|))^2. \tag{B.24}$$

For more general FF, as long as they are MLP with Lipschitz activations, the normalized nature of g_A should guarantee boundedness of E^{FF} and, consequently, E.

B.5 Experiments

B.5.1 Architecture Details

NRGPT serves as a drop-in replacement for a standard GPT block, where tokens and positions are embedded using learnable embedding matrices following the precedent of GPT-2.[1] Text is tokenized using the default OpenAI BPE tokenizer from GPT-2 (with just over 50k vocab tokens). For prediction, tokens are "unembedded" using a linear projection layer whose weights are shared by the input's embedding matrix.

We describe the predictions of NRGPT as a *sampling process* obtained by minimizing the energy of each token, which serves as a recurrent replacement of the *complete forward pass* through the transformer. This is easiest to explain for when $\eta = 1$ where the forward pass is explicit gradient descent on the energy E_A of token index A (these energy trajectories are shown in Figure 4.2). Gradient descent (GD) provides a fast, differentiable way to move initial points to locations which have lower energy (higher likelihood). Each new token starts from an initial position $\mathbf{x}_A(t=0)$, and the forward pass does GD, evolving it to a point $x_A(t=t_f)$ which has lower energy. In this case, E_A can be interpreted as the log-transition probability. In all GPT-style models and therefore also in NRGPT, we choose the initial point $\mathbf{x}_A(t=0)$ to be $\mathbf{x}_{A-1}$, the embedding of the previous token, but tokens could be initialized to anything in theory.

In the case where $\boldsymbol{\eta} \neq \mathbf{1}$, the forward pass follows more complex dynamics than pure

[1] Specifically, GPT-2's implementation by the nanoGPT repository

GD, but in general the model learns dynamics that convert $\mathbf{x}_{A-1}$ to $\mathbf{x}_A$. We refer to this process as "sampling" throughout this work; however, we note that this is not a strict sampling process for any value of $\boldsymbol{\eta}$.

B.5.2 Evaluation Metrics

To assess the generation quality of our language models, we utilize several complementary metrics. We use Perplexity as a measure of model uncertainty, computed using a pretrained GPT-2 model to evaluate how well the generated text aligns with expected language patterns. Lower perplexity indicates more fluent and predictable text, with scores typically ranging from 10 (excellent) to 1000+ (poor quality). For lexical diversity, we utilize Distinct-1 and Distinct-2, which measure the ratio of unique unigrams and bigrams to total n-grams (or total words) in the generated text, respectively. These metrics range from 0 to 1, where higher values indicate greater vocabulary diversity and less repetitive generation. A Distinct-1 score near 0 suggests highly repetitive text, while scores above 0.8 indicate rich vocabulary usage. We utilize Average Pairwise Cosine Similarity using Sentence-BERT embeddings [241] to measure semantic diversity within generated samples. This metric calculates the mean cosine similarity between all pairs of generated sentences, ranging from -1 to 1. Optimal values fall between 0.3 and 0.6, balancing semantic diversity with topical coherence. Values below 0.3 indicate excessive divergence with potentially incoherent topic-jumping between sentences, while values above 0.7 suggest repetitive or redundant content with insufficient variation. The target range of 0.3-0.6 represents healthy diversity where generated sentences explore different aspects of a topic while maintaining semantic relevance and coherent narrative flow.

Finally, we compute the Grammar Quality Score (GQS), a composite metric that combines rule-based grammar error detection, spelling accuracy via spellchecker [242], and readability assessment using Flesch-Kincaid grade level [243]. GQS ranges from 0 (poor grammar) to 1 (perfect grammar), weighting grammatical correctness (50%), spelling accuracy (30%), and readability (20%). The metric identifies errors across ten categories including subject-verb agreement, tense consistency, and punctuation, with severity-weighted scoring. For complete context and to understand what the ideal ranges are for all of these metrics, see Table B.1.

Table B.1: Ideal ranges for generation quality metrics

Metric	Good Range	Interpretation
Perplexity	15-50	Lower is better (fluency)
Distinct-1	0.6–0.9	Higher is better (vocabulary diversity)
Distinct-2	0.8–0.95	Higher is better (bigram diversity)
GQS	0.8–1.0	Higher is better (grammatical quality)
Avg. Cosine Similarity	0.3–0.6	Moderate values best (semantic diversity)

B.5.3 Shakespeare

As training data we used the full Shakespeare training set, tokenized such that each character constitutes a single token. Models were evaluated across the same held out validation subset. Across all models, we used a context window of 256 tokens, dropout of 10%, the AdamW(β_1=0.9, β_2=0.99), and 40k maximum update iterations using a minibatch size of 64. We varied model sizes by sweeping over embedding dimensions $(32, 64, 128, 256, 380, 512, 768)$ and the number of attention heads $(1, 2, 8)$. For the recurrent models, we additionally varied the number of layers across $(3, 6, 8)$, though this does not affect the parameter count.

We found the recurrent models to be quite sensitive to choices of learning rate and learning rate schedules. Hence, we explored several different maximum learning rates $(1e-3, 7.5e-4, 3e-4, 1e-4)$, schedules (cosine, exponential), and minimum learning rates ($10\times$, $20\times$, and $100\times$ smaller than the max learning rate). LR warm-up was 100 updates for all experiments.

In Figure 4.4 we emphasize the best losses we were able to achieve for each model size. In addition, we capture the model's sensitivity to hyperparams by showing the top 50% performing models across all hyperparameters.

Best Model Configurations and Metrics

Table B.2 shows the best model configuration for baseline GPTs and our model NRGPT with the respective generation quality metrics. We see that NRGPT outperforms the baseline GPT, RGPT and RGPT-parallel in terms of generation quality while it has only half of the nparams of GPT.

Table B.2: Best model configurations and quality metrics for Shakespeare. Note abbreviations: RGPT-parallel → RGPT-P, number of parameters → n_param, grammar quality score → gqs, average pairwise cosine similarity → apcs, distinct-1 → d-1 and distinct-2 → d-2.

Model	**Configuration**						**Metrics**				
	lr	min_lr	n_layer	n_head	n_embed	n_params	perplexity	gqs	apcs	d-1	d-2
GPT	5e-3	5e-4	8	4	64	0.4M	294	0.913	0.198	0.751	0.978
RGPT	1e-3	1e-4	8	1	256	0.8M	476	0.896	0.164	0.794	0.995
RGPT-P	2e-3	2e-4	8	1	128	**0.2M**	410	0.888	0.190	0.838	**1**
NRGPT_H_FF1	3e-4	3e-5	6	2	512	2M	318	0.901	**0.218**	**0.797**	0.995
NRGPT_H_FF2W	1e-3	1e-4	8	1	128	**0.2M**	**283**	**0.975**	0.176	0.765	0.995

B.5.4 OpenWebText

We perform experiments on natural language modeling using the OpenWebText corpus, which is an open-source recreation of GPT-2's WebText dataset [4]. The dataset contains approximately 17GB of text with 9B tokens that came from 8 million documents. We tokenize using byte-pair encoding (BPE) with a vocabulary size of 50,257 tokens. Our training sequences are fixed-length contexts of 1024 tokens. Table B.5 lists the hyperparameters and respective values/ranges used in our experiments. Figure B.1 shows an example of generated text by GPT, RGPT-parallel and NRGPT.

We report the best OWT training configurations for the metrics reported in the main text in Table 4.2. We additionally test models at a more comparable 125M parameter scale in Table B.3.

MMLU

Perplexity and simple generation metrics are insufficient to understand the performance of language models on downstream tasks. Thus, we additionally characterize the equal-parameter models of Table B.3 on the MMLU dataset [118] using five-shot generation in Table B.4. Remarkably, we find that NRGPT *outperforms both GPT and RecGPT at constant parameter count* on the average MMLU score.

As a caveat, we note that transformers at the $\sim$125M scale generally perform poorly on these downstream tasks. For example, the original MMLU paper [118] shows that GPT-3-small (2.7 billion params) and GPT-3-medium (6.7 billion params) [244] achieve a 25% success rate — the exact same as random guessing. A large GPT-2 model (1.5 billion params) [103] achieves a slightly better 32.4% accuracy, but this is still only 7% higher than random prediction. Our NRGPT model at 125M parameters (an order of magnitude

Table B.3: Best Model Configurations and Quality Metrics (average of four generations) for OWT with similar numbers of parameters. Note abbreviations: number of parameters → n_param, best train loss → t_loss, best validation loss → v_loss, grammar quality score → gqs, average pairwise cosine similarity → apcs, distinct-1 → d-1 and distinct-2 → d-2.

Model	Configuration								Metrics				
	lr	min_lr	n_layer	n_head	n_embed	n_params	t_loss	v_loss	perplexity	gqs	apcs	d-1	d-2
GPT	7e-4	7e-5	12	12	768	124M	2.90	2.93	78	0.947	0.274	0.638	**0.963**
GPT_Rec_Parallel	6e-4	6e-5	6	12	1740	126M	3.07	3.08	74	**0.950**	0.275	0.613	0.962
NRGPT_H_FF2W	3e-5	3e-6	6	12	1536	128M	3.29	3.30	105	0.946	0.306	**0.650**	0.960

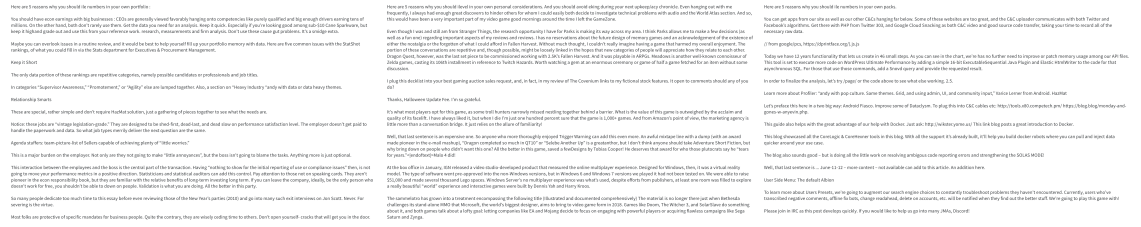

Figure B.1: Best Generation Examples from GPT (left column), RGPT-parallel (middle column) and NRGPT (right column).

smaller than the MMLU results of GPT-2) achieves a 29.3% accuracy, which is the best performance per parameter count on this task that we are aware of.

We believe that our results in Table B.4 should be taken with a grain of salt, since we are testing small models and comparing between low accuracies. However, we also believe that these hint at NRGPT's advantage at greater scales.

Table B.4: MMLU performance comparison using five-shot generation. The models tested in this work are of insufficient size and scale to achieve good performance on all of MMLU, so we report results by subject. Best are bold.

Model	Astronomy	Science	CS	Economics	Medicine	Math	Global Facts	Average
NRGPT (128M)	35.5	**32.1**	**31.5**	30.0	30.5	**29.2**	**19.0**	**29.3**
RGPT (126M)	28.3	27.1	28.0	**32.1**	25.2	24.1	17.0	26.0
GPT (124M)	**36.8**	29.5	26.2	26.7	**31.2**	27.3	18.0	28.0

Table B.5: OWT Hyperparameters and range of values.

Hyperparameter	Used Values
batch_size	12
block_size	1024
n_layer	[3, 6, 9, 12, 24]
n_head	[1, 2, 4, 6, 12, 16]
n_embed	[768, 1020, 1536]
learning_rate, lr	[1e-3 - 1e-5]
min_lr	[lr/10 - lr]
beta1	0.9
beta2	0.99
weight_decay	[1e-1, 1e-2]
gradient_accumulation_steps	40
eval_interval	1,000
eval_iters	200
max_iters	100000
warmup_iters	[100, 2000]
dropout	0.0

APPENDIX C
DRDAM SUPPLEMENTARY MATERIAL

This appendix collects the supplementary material from the original DRDAM work so that the dissertation is self-contained. It includes additional limitations, stored-pattern scaling experiments, basis-function ablations, experimental details, and proofs supporting Chapter 6.

C.1 Limitations

In this chapter, we have explored the use of distributed representations via random feature maps in DenseAMs. However, we are only scratching the surface of opportunities that such distributed representations bring to DenseAMs. There are various aspects we do not cover: (i) We do not cover the ability of these distributed representations to provide (probably lossy) compression. (ii) We do not study the properties of DRDAM relative to MRDAM when DRDAM is allowed to have different step sizes and number of layers than MRDAM. A further limitation of our work is the limited number of datasets on which we have characterized the performance of DRDAM.

C.2 Approximation Error When Increasing the Number of Stored Patterns in DRDAM

Subsection 6.4.1 validated Equation (6.13), confirming that approximation error decreases as the number of random features Y increases under constant number of stored patterns K. We can also consider a related but different question: under constant number of random features Y, how does approximation error behave when increasing the number of stored patterns K? Intuitively, DRDAM's approximation should be good when a small number of patterns are stored in the network, and this approximation should worsen as we increase the number of stored patterns.

Figure C.1 validates this intuition empirically, with the caveat that random queries generally improve in accuracy because the probability of being near a stored pattern (a regime that generally leads to higher accuracy of retrievals, see § 6.4) increases as we store more patterns in the network. For this experiment, $Y = 2e5$ was held constant across all experiments and each plotted approximation error is averaged over a number of queries equal

to the number of stored patterns K. The experimental design otherwise exactly replicates that of Figure 6.3.

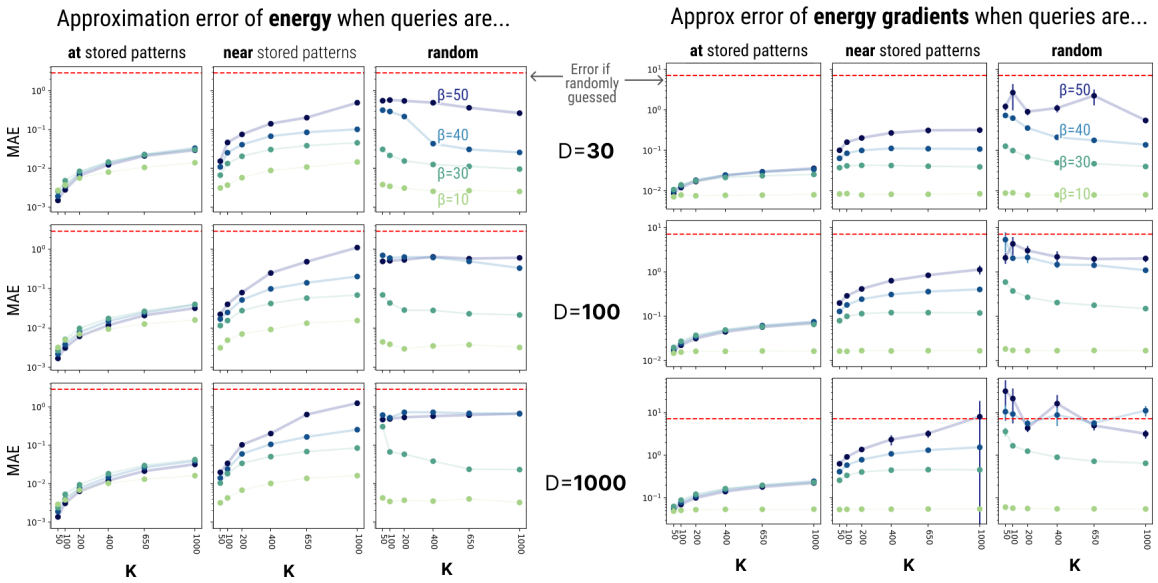

Figure C.1: Mean Approximation Error (MAE, Equation (6.15)) increases as the number of stored patterns K increases (except at random starting positions, where more stored patterns increases the probability that a random query is closer to a memory, a regime that leads to higher accuracy of the retrievals, see Figure 6.3), keeping $Y = 2e5$ constant across all experiments.

C.3 Ablation Study: Comparing Choices for Basis Function

Different basis functions can be used to approximate the RBF kernel used in the energy of the memory representation of MrDAM in Equation (6.7). We considered the following kernels ("Cos", "SinCos", "Exp", "ExpExp"), rewritten here as

$$
\begin{aligned}
\boldsymbol{\varphi}_{\text{Cos}}(\mathbf{x}) &= \sqrt{\frac{2}{Y}} \begin{bmatrix} \cos(\langle \boldsymbol{\omega}^1, \mathbf{x} \rangle + b_1) \\ \cos(\langle \boldsymbol{\omega}^2, \mathbf{x} \rangle + b_2) \\ \dots, \\ \cos(\langle \boldsymbol{\omega}^Y, \mathbf{x} \rangle + b_Y) \end{bmatrix}, \\
\boldsymbol{\varphi}_{\text{SinCos}}(\mathbf{x}) &= \frac{1}{\sqrt{Y}} \begin{bmatrix} \cos(\langle \boldsymbol{\omega}^1, \mathbf{x} \rangle) \\ \sin(\langle \boldsymbol{\omega}^1, \mathbf{x} \rangle) \\ \cos(\langle \boldsymbol{\omega}^2, \mathbf{x} \rangle) \\ \sin(\langle \boldsymbol{\omega}^2, \mathbf{x} \rangle) \\ \dots, \\ \cos(\langle \boldsymbol{\omega}^Y, \mathbf{x} \rangle) \\ \sin(\langle \boldsymbol{\omega}^Y, \mathbf{x} \rangle) \end{bmatrix}, \\
\boldsymbol{\varphi}_{\text{Exp}}(\mathbf{x}) &= \frac{\exp(-\|\mathbf{x}\|_2^2)}{\sqrt{Y}} \begin{bmatrix} \exp(\langle \boldsymbol{\omega}^1, \mathbf{x} \rangle + b_1) \\ \exp(\langle \boldsymbol{\omega}^2, \mathbf{x} \rangle + b_2) \\ \dots, \\ \exp(\langle \boldsymbol{\omega}^Y, \mathbf{x} \rangle + b_Y) \end{bmatrix}, \\
\boldsymbol{\varphi}_{\text{ExpExp}}(\mathbf{x}) &= \frac{\exp(-\|\mathbf{x}\|_2^2)}{\sqrt{2Y}} \begin{bmatrix} \exp(+\langle \boldsymbol{\omega}^1, \mathbf{x} \rangle) \\ \exp(-\langle \boldsymbol{\omega}^1, \mathbf{x} \rangle) \\ \exp(+\langle \boldsymbol{\omega}^2, \mathbf{x} \rangle) \\ \exp(-\langle \boldsymbol{\omega}^2, \mathbf{x} \rangle) \\ \dots, \\ \exp(+\langle \boldsymbol{\omega}^Y, \mathbf{x} \rangle) \\ \exp(-\langle \boldsymbol{\omega}^Y, \mathbf{x} \rangle) \end{bmatrix},
\end{aligned}
$$

where $\boldsymbol{\omega}^\alpha \sim \mathcal{N}(0, I_D)$, $\alpha \in [\![Y]\!]$ are the random projection vectors and $b^\alpha \sim \mathcal{U}(0, 2\pi)$ are random "biases" or shifts in the basis function.

Figure C.2 shows how well the above basis functions approximated the true energy and energy gradient at different values for β and size of feature dimension Y. Specifically, given the Letter dataset [245] which consists of 16-dimensional continuous vectors whose values were normalized to be between $[0, \frac{1}{\sqrt{D}}]$, we randomly selected 900 unique data points, storing 500 patterns in the memory and choosing the remaining 400 to serve as new patterns. We then compared how well the energy and energy gradients of the chosen basis function

approximate the predictions of MRDAM.

We observe that the trigonometric basis functions (i.e., either Cos or SinCos) provide the most accurate approximations for the energy and gradients of the standard MRDAM, especially in the regime of high β which is required for the high memory storage capacity of DenseAMs. Surprisingly, the Positive Random Features (PRFs) of [55] do not perform well in the dense (high β) regime; in general, trigonometric features always provide better approximations than the PRFs.

We conclude that the SinCos basis function is the best approximation for use in the experiments reported in this chapter, as this choice consistently produces the best approximations for the energy gradients across all values of β.

C.4 Tiny ImageNet Experimental Details

In performing the qualitative reconstructions shown in Figure 6.1, we used a standard MRDAM energy (Equation (6.7)) configured with inverse temperature $\beta = 60$. We approximated this energy in a DRDAM using the trigonometric "SinCos" basis function shown in Equation (6.8) configured with feature dimension $Y = 1.8e5$. The four images shown were selected from the Tiny Imagenet [1] dataset, rasterized into a vector, and stored in the memory matrix of MRDAM, resulting in a memory of shape $(4, 12288)$. Energy descent for both MRDAM and DRDAM used standard gradient descent at a step size of 0.1 until the dynamics of all images converged (for Figure 6.1 after 300 steps, see energy traces). Visible pixels are "clamped" throughout the duration of the dynamics by zeroing out the energy gradients on the visible top one-third of the image.

In MRDAM, the memory matrix necessarily grows linearly when storing new patterns $\boldsymbol{\xi}^\mu$. However, the distributed memory tensor $\mathbf{T}$ of DRDAM does not grow when new patterns are stored. This means it is possible to compress the memories into a smaller tensor $\mathbf{T}$ where $Y <$ NumPixels, provided that we operate in a regime that allows larger approximation errors in the retrieval and smaller initial occlusions. Figure 6.2 shows a variation of the setting of Figure 6.1 where stored patterns are actually *compressed* into DRDAM's memory tensor, successfully storing 20×12288 pixels from a distributed tensor of size $Y = 2e5$ and retrieving the memories with 40% initial occlusion of the queries, a $\sim$20% reduction in the number of parameters compared to MRDAM. All other hyperparameters are the same as was used to generate Figure 6.1, and convergence on all images occurs after 1000 steps.

C.5 Details on Computational Environment for the Experiments

All experiments are performed on a single L40s GPU equipped with 46GB VRAM. Experiments were written and performed using the JAX [226] library for tensor manipulations. Unless otherwise noted, gradient computation was performed using JAX's powerful autograd mechanism. Experimental code with instructions to replicate the results in this work is made available at this GitHub repository (https://github.com/bhoov/distributed_DAM), complete with instructions to setup the coding environment and run all experiments.

C.6 Detailed Proofs and Discussions

C.6.1 Details for the DrDAM Energy Gradient Descent Complexity Theorem

Proof of the DrDAM *Energy Gradient Descent Complexity Theorem*

Proof of Theorem 6.1. The proof above involves noting that, first we need to encode all the memories with ProcMems, which takes $O(DYK)$ time and $O(Y + D)$ peak memory using Proposition 6.3.

Then we compute L gradients with GradComp for L iterations of energy gradient descent, taking $O(LD(Y + D))$ time and $O(D + Y)$ peak memory using Proposition 6.4.

Putting the runtimes together, and using the maximum of the peak memories gives us the statement of the theorem. □

Comparing computational complexities of MrDAM *and* DrDAM

Note that, comparing the computational complexities of MrDAM in Proposition 6.1 to that of DrDAM in Theorem 6.1 does not directly provide any computational improvements as it would depend on the choices of D, K, L, Y. The main point of these results is to highlight, that once the memories are processed via ProcMems, the energy descent with DrDAM requires computation and memory that only depends on D and Y. And together with Theorem 6.2 and Corollary 6.1, we characterize situations where the energy descent divergence between MrDAM and DrDAM can be bounded with a choice of Y that only depends on D (and other parameters in the energy function) but not K.

While we do not claim or highlight computational gains over MrDAM, note that the peak memory complexity of MrDAM is $O(KD)$ compared to $O(Y + D)$ for DrDAM. Given that in the interesting regime of $Y \sim O(D/\epsilon^2)$ which upperbounds the energy descent

divergence between DrDAM and MrDAM in Corollary 6.1 to at most some $\epsilon > 0$, DrDAM is more memory efficient than MrDAM if the number of memories $K > C/\epsilon^2$ for some sufficiently large positive constant C. Ignoring the time required to encode the memories into the distributed representation in DrDAM using ProcMems, the runtime complexities are $O(LKD)$ for MrDAM compared to $O(LD(Y+D))$ for DrDAM. Again, considering the interesting regime of $Y \sim O(D/\epsilon^2)$, DrDAM will be computationally more efficient than MrDAM if the number of memories $K > \widetilde{C}D/\epsilon^2$ for some sufficiently large positive constant $\widetilde{C}$.

C.6.2 Details for the Energy Descent Divergence Theorem

Proof of the Energy Descent Divergence Theorem

Here we will make use of the following result from Li et al. [246]:

Lemma C.1 (adapted from Li et al. [246] Lemma B.1)**.** *For* $\mathbf{x}, \mathbf{z} \in \mathbb{R}^K$ *with* $\max_{i,j \in [\![K]\!]}(\mathbf{x}_i - \mathbf{x}_j) \leq \delta$ *and* $\max_{i,j \in [\![K]\!]}(\mathbf{z}_i - \mathbf{z}_j) \leq \delta$*, we have the following:*

$$\|\text{softmax}(\mathbf{x})\|_\infty \leq \frac{e^\delta}{K}, \quad \|\text{softmax}(\mathbf{x}) - \text{softmax}(\mathbf{z})\|_1 \leq \frac{e^\delta}{K}\|\mathbf{x} - \mathbf{z}\|_1. \tag{C.1}$$

We now develop the following results:

Lemma C.2. *Under the conditions and notation of Theorem 6.2, for* $\mathbf{x}, \mathbf{z} \in \mathcal{X}$*, we have*

$$\|\nabla_{\mathbf{x}} E(\mathbf{x}) - \nabla_{\mathbf{x}} E(\mathbf{z})\| \leq (1 + 2K\beta e^{\beta/2})\|\mathbf{x} - \mathbf{z}\|. \tag{C.2}$$

Proof. Given the energy function in Equation (6.11), we can write the energy gradient $\nabla_{\mathbf{x}} E(\mathbf{x})$ as:

$$\nabla_{\mathbf{x}} E(\mathbf{x}) = \text{softmax}(-\beta/2\|\mathbf{x} - \Xi\|_2^2)(\mathbf{x} - \Xi) = \mathbf{x} - \text{softmax}(-\beta/2\|\mathbf{x} - \Xi\|_2^2)\Xi, \tag{C.3}$$

where $\Xi = [\xi^1, \ldots, \xi^K]$, $\|\mathbf{x} - \Xi\|_2^2$ denotes $[\|\mathbf{x} - \xi^1\|_2^2, \ldots \|\mathbf{x} - \xi^K\|_2^2]$ and $(\mathbf{x} - \Xi)$ denotes $[(\mathbf{x} - \xi^1), \ldots, (\mathbf{x} - \xi^K)]$. Then we have

$$\|\nabla_{\mathbf{x}} E(\mathbf{x}) - \nabla_{\mathbf{x}} E(\mathbf{z})\|_2 \tag{C.4}$$

$$= \|\mathbf{x} - \text{softmax}(-\beta/2\|\mathbf{x} - \Xi\|_2^2)\Xi - \mathbf{z} + \text{softmax}(-\beta/2\|\mathbf{z} - \Xi\|_2^2)\Xi\|_2 \tag{C.5}$$

$$\leq \|\mathbf{x} - \mathbf{z}\| + \|(\text{softmax}(-\beta/2\|\mathbf{x} - \Xi\|_2^2) - \text{softmax}(-\beta/2\|\mathbf{z} - \Xi\|_2^2))\Xi\|_2 \tag{C.6}$$

$$\leq \|\mathbf{x} - \mathbf{z}\| + \|(\text{softmax}(-\beta/2\|\mathbf{x} - \Xi\|_2^2) - \text{softmax}(-\beta/2\|\mathbf{z} - \Xi\|_2^2))\|_1 \|\Xi\|_2 \tag{C.7}$$

$$\leq \|\mathbf{x} - \mathbf{z}\| + \frac{e^{\beta/2}}{K}\|\Xi\|_2 \left\|\beta/2(\|\mathbf{z} - \Xi\|_2^2 - \|\mathbf{x} - \Xi\|_2^2)\right\|_1, \tag{C.8}$$

where we applied Lemma C.1 to the softmax term in the right hand side of Equation (C.7) with $\delta = \beta/2$ since all pairwise distances in $\mathcal{X}$ are in $[0, 1]$.

Now we have

$$\left\|\beta/2(\|\mathbf{z} - \Xi\|_2^2 - \|\mathbf{x} - \Xi\|_2^2)\right\|_1 = \frac{\beta}{2}\sum_{\mu=1}^{K} \left|\|\mathbf{z} - \boldsymbol{\xi}^\mu\|_2^2 - \|\mathbf{x} - \boldsymbol{\xi}^\mu\|_2^2\right| \tag{C.9}$$

$$= \frac{\beta}{2}\sum_{\mu=1}^{K} \left|\langle \mathbf{z} + \mathbf{x}, \mathbf{z} - \mathbf{x}\rangle + 2\langle \boldsymbol{\xi}^\mu, \mathbf{x} - \mathbf{z}\rangle\right| \tag{C.10}$$

$$\leq \frac{\beta}{2}\sum_{\mu=1}^{K} \|\mathbf{z} - \mathbf{x}\|(\|\mathbf{z} + \mathbf{x}\| + 2\|\boldsymbol{\xi}^\mu\|) \leq \frac{\beta}{2}\sum_{\mu=1}^{K} 4\|\mathbf{z} - \mathbf{x}\|, \tag{C.11}$$

since $\|\boldsymbol{\xi}^\mu\| \leq 1$ and $\|\mathbf{x}+\mathbf{z}\| \leq \|\mathbf{x}\|+\|\mathbf{z}\| \leq 2$. Putting Equation (C.11) in Equation (C.8), and using the fact that $\|\Xi\|_2 \leq K$, we have

$$\|\nabla_{\mathbf{x}} E(\mathbf{x}) - \nabla_{\mathbf{x}} E(\mathbf{z})\|_2 \leq \|\mathbf{x}-\mathbf{z}\| + \frac{e^{\beta/2}}{K} K 2\beta \sum_{\mu=1}^{K} \|\mathbf{z}-\mathbf{x}\| = (1+2K\beta e^{\beta/2})\|\mathbf{x}-\mathbf{z}\|, \tag{C.12}$$

giving us Equation (C.1) in the statement of the lemma. □

Given the structure of the energy gradient in Equation (C.3) of the energy function in Equation (6.11), we consider a specific energy gradient for this specific energy function instead of the generic energy gradient in Equation (6.10). We can rewrite the exact energy

gradient as

$$\nabla_{\mathbf{x}} E(\mathbf{x}) = \mathbf{x} - \sum_{\mu=1}^{K} \frac{\exp(-\beta/2 \|\mathbf{x} - \boldsymbol{\xi}^{\mu}\|_2^2) \boldsymbol{\xi}^{\mu}}{\sum_{\mu'=1}^{K} \exp(-\beta/2 \|\mathbf{x} - \boldsymbol{\xi}^{\mu'}\|_2^2)}. \tag{C.13}$$

Using random feature maps, we can write the approximate gradient as

$$\nabla_{\mathbf{x}} \hat{E}(\mathbf{x}) = \mathbf{x} - \sum_{\mu=1}^{K} \frac{\left\langle \boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}), \boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu}) \right\rangle \boldsymbol{\xi}^{\mu}}{\sum_{\mu'=1}^{K} \left\langle \boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}), \boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu'}) \right\rangle} \tag{C.14}$$

$$= \frac{\sum_{\mu=1}^{K} \boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}) \cdot \boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu}) \cdot \boldsymbol{\xi}^{\mu\top}}{\left\langle \boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}), \sum_{\mu'=1}^{K} \boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu'}) \right\rangle} \tag{C.15}$$

$$= \frac{\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}) \cdot \sum_{\mu=1}^{K} \boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu}) \cdot \boldsymbol{\xi}^{\mu\top}}{\left\langle \boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}), \mathbf{T} \right\rangle}, \quad \text{where } \mathbf{T} = \sum_{\mu'=1}^{K} \boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu'}) \tag{C.16}$$

$$= \frac{\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}) \cdot \mathbf{R}}{\left\langle \boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}), \mathbf{T} \right\rangle}, \quad \text{where } \mathbf{R} = \sum_{\mu=1}^{K} \boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu}) \cdot \boldsymbol{\xi}^{\mu\top}, \tag{C.17}$$

where we again just need to store $\mathbf{T}$ and $\mathbf{R}$ as defined above and do not need to maintain the original memory matrix Ξ.

Lemma C.3. *Under the conditions and notation of Theorem 6.2, and assuming that* $\langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\varphi}(\mathbf{x}') \rangle \geq 0 \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$*, for the approximate gradient* $\nabla_{\mathbf{x}} \hat{E}(\mathbf{x})$ *in Equation (C.17), we have*

$$\|\nabla_{\mathbf{x}} E(\mathbf{x}) - \nabla_{\mathbf{x}} \hat{E}(\mathbf{x})\| \leq 2 C_1 K e^{\beta E(\mathbf{x})} \sqrt{\frac{D}{Y}}. \tag{C.18}$$

Proof. We can expand out the left-hand side of Equation (C.18) as follows:

$$\|\nabla_{\mathbf{x}} E(\mathbf{x}) - \nabla_{\mathbf{x}} \hat{E}(\mathbf{x})\| = \left\| \frac{\sum_{\mu=1}^{K} \exp(-\beta/2 \|\mathbf{x} - \boldsymbol{\xi}^{\mu}\|^2) \boldsymbol{\xi}^{\mu}}{\sum_{\mu=1}^{K} \exp(-\beta/2 \|\mathbf{x} - \boldsymbol{\xi}^{\mu}\|^2)} - \frac{\sum_{\mu=1}^{K} \left\langle \boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}), \boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu}) \right\rangle \boldsymbol{\xi}^{\mu}}{\sum_{\mu=1}^{K} \left\langle \boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}), \boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu}) \right\rangle} \right\| \tag{C.19}$$

by reversing the simplifying steps made above to arrive at Equation (C.17).

Let us denote $\epsilon = C_1 \sqrt{D/Y}$ the approximation in the kernel value induced by the

random feature map $\boldsymbol{\varphi}$. Then considering the terms in the denominator above, we have

$$(1/K)\left|\sum_{\mu}\exp(-\beta/2\|\mathbf{x}-\boldsymbol{\xi}^{\mu}\|^2)-\left\langle\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}),\sum_{\mu}\boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu})\right\rangle\right| \tag{C.20}$$

$$=(1/K)\left|\sum_{\mu}\left(\exp(-\beta/2\|\mathbf{x}-\boldsymbol{\xi}^{\mu}\|^2)-\left\langle\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}),\boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu})\right\rangle\right)\right| \tag{C.21}$$

$$\leq(1/K)\sum_{\mu}\left|\left(\exp(-\beta/2\|\mathbf{x}-\boldsymbol{\xi}^{\mu}\|^2)-\left\langle\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}),\boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu})\right\rangle\right)\right|\leq(1/K)\sum_{\mu}\epsilon=\epsilon. \tag{C.22}$$

Considering the terms in the numerators, we have

$$(1/K)\left\|\sum_{\mu}\exp(-\beta/2\|\mathbf{x}-\boldsymbol{\xi}^{\mu}\|^2)\boldsymbol{\xi}^{\mu}-\left\langle\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}),\sum_{\mu}\boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu})\right\rangle\boldsymbol{\xi}^{\mu}\right\| \tag{C.23}$$

$$=(1/K)\left\|\sum_{\mu}\left(\exp(-\beta/2\|\mathbf{x}-\boldsymbol{\xi}^{\mu}\|^2)-\left\langle\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}),\boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu})\right\rangle\right)\boldsymbol{\xi}^{\mu}\right\| \tag{C.24}$$

$$\leq(1/K)\sum_{\mu}\left\|\left(\exp(-\beta/2\|\mathbf{x}-\boldsymbol{\xi}^{\mu}\|^2)-\left\langle\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}),\boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu})\right\rangle\right)\boldsymbol{\xi}^{\mu}\right\| \tag{C.25}$$

$$\leq(1/K)\sum_{\mu}\left|\left(\exp(-\beta/2\|\mathbf{x}-\boldsymbol{\xi}^{\mu}\|^2)-\left\langle\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}),\boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu})\right\rangle\right)\right|\|\boldsymbol{\xi}^{\mu}\| \tag{C.26}$$

$$\leq(1/K)\sum_{\mu}\epsilon\|\boldsymbol{\xi}^{\mu}\|=\epsilon\qquad\because\|\boldsymbol{\xi}^{\mu}\|\leq 1. \tag{C.27}$$

Let us define the following terms for convenience:

- $a=1/K\sum_{\mu}\exp(-\beta/2\|\mathbf{x}-\boldsymbol{\xi}^{\mu}\|^2)\boldsymbol{\xi}^{\mu}$
- $b=1/K\sum_{\mu}\exp(-\beta/2\|\mathbf{x}-\boldsymbol{\xi}^{\mu}\|^2)$
- $\hat{a}=1/K\sum_{\mu}\left\langle\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}),\boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu})\right\rangle\boldsymbol{\xi}^{\mu}$
- $\hat{b}=1/K\sum_{\mu}\left\langle\boldsymbol{\varphi}(\sqrt{\beta}\mathbf{x}),\boldsymbol{\varphi}(\sqrt{\beta}\boldsymbol{\xi}^{\mu})\right\rangle$

Then, based on our previous bounds, we know that

$$\|a-\hat{a}\|\leq\epsilon,\quad|b-\hat{b}|\leq\epsilon,\quad\|\nabla_{\mathbf{x}}E(\mathbf{x})-\nabla_{\mathbf{x}}\hat{E}(\mathbf{x})\|=\left\|\frac{a}{b}-\frac{\hat{a}}{\hat{b}}\right\| \tag{C.28}$$

$$\|\nabla_{\mathbf{x}} E(\mathbf{x}) - \nabla_{\mathbf{x}} \hat{E}(\mathbf{x})\| = \left\| \frac{a}{b} - \frac{\hat{a}}{\hat{b}} \right\| = \left\| \frac{a-\hat{a}}{b} + \frac{\hat{a}}{\hat{b}}\frac{\hat{b}}{b} - \frac{\hat{a}}{\hat{b}} \right\| \leq \left\| \frac{a-\hat{a}}{b} \right\| + \left\| \frac{\hat{a}}{\hat{b}} \right\| \left| \frac{\hat{b}}{b} - 1 \right| \tag{C.29}$$

$$\leq \frac{1}{b}\|a-\hat{a}\| + \left\| \frac{\hat{a}}{\hat{b}} \right\| \frac{1}{b}|\hat{b}-b| \leq \frac{1}{b}\left(\epsilon + \left\| \frac{\hat{a}}{\hat{b}} \right\| \epsilon\right) \tag{C.30}$$

$$\leq \epsilon \frac{1}{b}\left(1 + \left\| \frac{\hat{a}}{\hat{b}} \right\|\right) \tag{C.31}$$

.

Note that $(\hat{a}/\hat{b})$ is in the convex hull of the memories since this is a weighted sum of the memories where the weights are positive and add up to 1. Thus $(\hat{a}/\hat{b}) \in [0, 1/\sqrt{d}]^d$, so $\|\hat{a}/\hat{b}\| \leq 1$. Now $b = (1/K)\exp(-\beta E(\mathbf{x}))$. Thus

$$\|\nabla_{\mathbf{x}} E(\mathbf{x}) - \nabla_{\mathbf{x}} \hat{E}(\mathbf{x})\| \leq 2\epsilon K \exp(\beta E(\mathbf{x})) = 2C_1 K \exp(\beta E(\mathbf{x}))\sqrt{\frac{D}{Y}}, \tag{C.32}$$

giving us the right-hand side of Equation (C.18). □

Proof of Theorem 6.2. Expanding out the divergence $D^{(L)}$ after L energy descent steps, and using the fact that $\mathbf{x}^{(0)} = \hat{\mathbf{x}}^{(0)} = \mathbf{x}$, we have

$$D^{(L)} \triangleq \|\mathbf{x}^{(L)} - \hat{\mathbf{x}}^{(L)}\| \tag{C.33}$$

$$= \left\| \left(\mathbf{x}^{(0)} - \sum_{t \in [\![L]\!]} \eta \nabla_{\mathbf{x}} E(\mathbf{x}^{(t-1)}) \right) - \left(\hat{\mathbf{x}}^{(0)} - \sum_{t \in [\![L]\!]} \eta \nabla_{\mathbf{x}} \hat{E}(\hat{\mathbf{x}}^{(t-1)}) \right) \right\| \tag{C.34}$$

$$= \left\| \sum_{t \in [\![L]\!]} -\eta \left(\nabla_{\mathbf{x}} E(\mathbf{x}^{(t-1)}) - \nabla_{\mathbf{x}} \hat{E}(\hat{\mathbf{x}}^{(t-1)}) \right) \right\| \tag{C.35}$$

$$= \left\| \sum_{t \in [\![L]\!]} \eta \left(\nabla_{\mathbf{x}} E(\mathbf{x}^{(t-1)}) - \nabla_{\mathbf{x}} \hat{E}(\hat{\mathbf{x}}^{(t-1)}) \right) \right\| \tag{C.36}$$

$$\leq \sum_{t \in [\![L]\!]} \left\| \eta \left(\nabla_{\mathbf{x}} E(\mathbf{x}^{(t-1)}) - \nabla_{\mathbf{x}} \hat{E}(\hat{\mathbf{x}}^{(t-1)}) \right) \right\|. \tag{C.37}$$

Let us denote the individual terms above as $d^{(t)}$ with $D^{(L)} = \sum_{t \in [\![L]\!]} d^{(t)}$. Also, let us denote by $A = 2C_1 K e^{\beta E(\mathbf{x})}\sqrt{D/Y}$ and by $B = (1 + 2K\beta e^{\beta/2})$. Then writing out the t-th term

$$d^{(t)} = \left\| \eta \left(\nabla_{\mathbf{x}} E(\mathbf{x}^{(t-1)}) - \nabla_{\mathbf{x}} \hat{E}(\hat{\mathbf{x}}^{(t-1)}) \right) \right\| \tag{C.38}$$

$$\leq \left\| \eta \left(\nabla_{\mathbf{x}} E(\mathbf{x}^{(t-1)}) - \nabla_{\mathbf{x}} E(\hat{\mathbf{x}}^{(t-1)}) \right) \right\| + \left\| \eta \left(\nabla_{\mathbf{x}} E(\hat{\mathbf{x}}^{(t-1)}) - \nabla_{\mathbf{x}} \hat{E}(\hat{\mathbf{x}}^{(t-1)}) \right) \right\| \tag{C.39}$$

$$\leq \eta B \|\mathbf{x}^{(t-1)} - \hat{\mathbf{x}}^{(t-1)}\| + \eta A, \tag{C.40}$$

where the first term is bounded using Lemma C.2 and the definition of B, and the second term is bounded using Lemma C.3 and the definition of A.

Note that this gives us the recursion $d^{(t)} \leq \eta A + \eta B D^{(t-1)}$, and thus, $D^{(L)} \leq \sum_{t \in [\![L]\!]} \eta(A + BD^{(t-1)})$.

Writing out the recursion using induction, we can show that

$$D^{(L)}$$

$$= \eta A \left(\sum_{t \in [\![L]\!]} 1 + \sum_{t \in [\![L]\!]} t(\eta B) + \sum_{t \in [\![L]\!]} \sum_{t_1 \in [\![L]\!]} t_1 (\eta B)^2 + \sum_{t \in [\![L]\!]} \sum_{t_1 \in [\![L]\!]} \sum_{t_2 \in [\![t_1]\!]} t_2 (\eta B)^3 \right.$$
$$\left. + \cdots + \sum_{t \in [\![L]\!]} \sum_{t_1 \in [\![L]\!]} \cdots \sum_{t_{L-1} \in [\![t_{L-2}]\!]} t_{L-1} (\eta B)^{L-1} \right) \tag{C.41}$$

$$\leq \eta A \left(L + L(\eta BL) + L(\eta BL)^2 + \cdots + L(\eta BL)^{L-1} \right) \tag{C.42}$$

$$= \eta AL \frac{1 - (\eta BL)^L}{1 - \eta BL}. \tag{C.43}$$

Replacing the values of A and B above gives us the statement of the theorem. □

Dependence on the initial energy $E(\mathbf{x})$ of the input.

The divergence upper bound in Equation (6.12) (in Theorem 6.2) depends on the term $\exp(\beta E(\mathbf{x}))$. However, note that, for the energy function defined in Equation (6.11), assuming that all memories and the initial queries are in a ball of diameter 1 (which is the assumption A1 in Theorem 6.2), $E(\mathbf{x}) \leq \frac{1}{2} - \frac{\log K}{\beta}$, implying that $\exp(\beta E(\mathbf{x})) \leq \exp(\beta/2)/K$, and we can replace this in the upper bound and remove the dependence on $E(\mathbf{x})$.

However, an important aspect of our analysis is that the bound is input specific, and

depends on the initial energy $E(\mathbf{x})$. As discussed above, this can be upper bounded uniformly, but our bound is more adaptive to the input $\mathbf{x}$.

For example, if the input is initialized near one of the memories, while being sufficiently far from the remaining $(K-1)$ memories, then the $\exp(\beta E(\mathbf{x}))$ term can be relatively small. More precisely, with all memories and queries lying in a ball of diameter 1, let the query be at a distance $r < 1$ to its closest memories, and as far as possible from the remaining $(K-1)$ memories. In this case, the initial energy $E(\mathbf{x}) \approx -(1/\beta)\log(\exp(-\beta r/2) + (K-1)\exp(-\beta/2))$, implying that

$$\exp(\beta E(\mathbf{x})) \approx \frac{\exp(\beta r/2)}{1+\exp(-\beta(1-r)/2)} \leq \exp(\beta r/2). \tag{C.44}$$

For sufficiently small $r < 1$, the above quantity can be relatively small. If, for example, $r \sim O(\log K)$, then $\exp(\beta E(\mathbf{x})) \sim O(K^{\beta})$, while $r \to 0$ gives us $\exp(\beta E(\mathbf{x})) \to O(1)$. This highlights the adaptive input-dependent nature of our analysis.

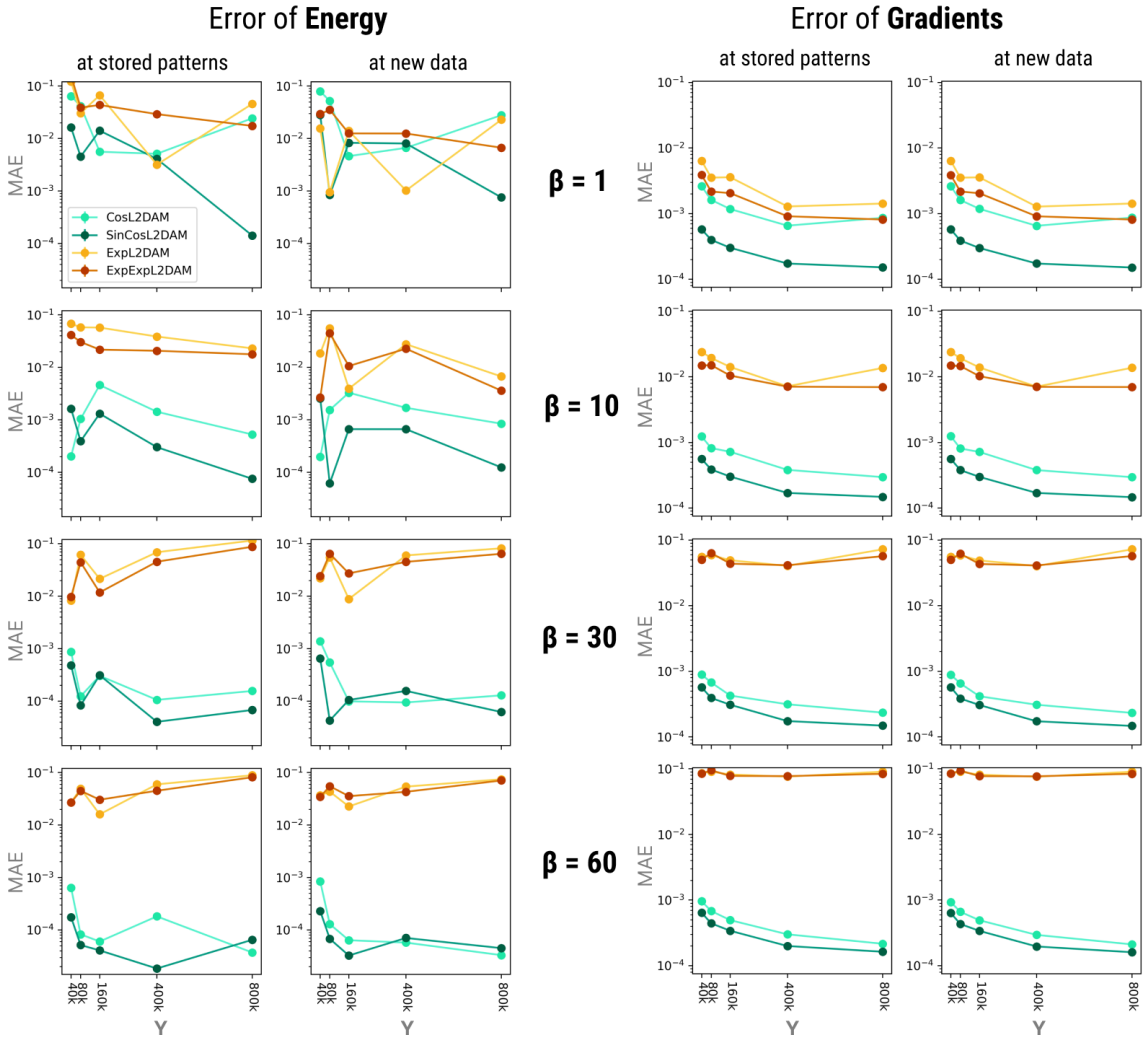

Figure C.2: Trigonometric basis functions significantly outperform Positive Random Features, especially in the regime of large β. We end up choosing the SinCos function to analyze in this chapter, as this choice of basis function always produced the best approximations to the energy gradient. Experiments performed on the 16-dimensional Letters dataset [245].

APPENDIX D
LSRDAM SUPPLEMENTARY MATERIAL

This appendix collects the supplementary material from the original LSRDAM work so that the dissertation is self-contained. It includes experimental details, qualitative ablations, limitations, discussion of emergent memories, and comparisons across compact-support kernels.

D.1 Experimental Details

D.1.1 Details: Quantifying Novel Minima

In this experiment we tested across a geometrically spaced range of $\beta \in [2d^{-1}, 2r_{\text{min}}^{-2}]$, where $r_{\text{min}} := \min_{\mu \neq \nu} \|\boldsymbol{\xi}_\mu - \boldsymbol{\xi}_\nu\|$ is the minimum pairwise distance between any two stored patterns in the current subset $\mathcal{K} \subseteq [\![M]\!]$. At the largest β, the support regions of the stored patterns are disjoint and the only memories are the M stored patterns themselves; this configuration has a very small support region (shown as the shaded green curve in Figure 7.3, which is computed by monte carlo sampling 1e6 points on the unit hypercube and computing the fraction of energies that are finite at $\epsilon = 0$) as a fraction of the unit hypercube. At the smallest tested β, only a single energy minimum is induced at the centroid of all stored patterns with a region of support covering the whole unit hypercube. At the largest tested β, all original memories are recoverable and there are no spurious memories.

D.1.2 Details: Generative Quality of Memories

Mixture Sweep Experiments

We ran the same experiment in Subsection 7.4.2 under varying dimensions $d = [8, 16]$ and number of mixtures $k = [5, 10]$, averaging the results of each run across 5 different random seeds. The results for each of these experiments is shown below in Figures D.1 and D.2, where Figure D.2 (left) is the same as reported in Figure 7.3 (right) in the main text.

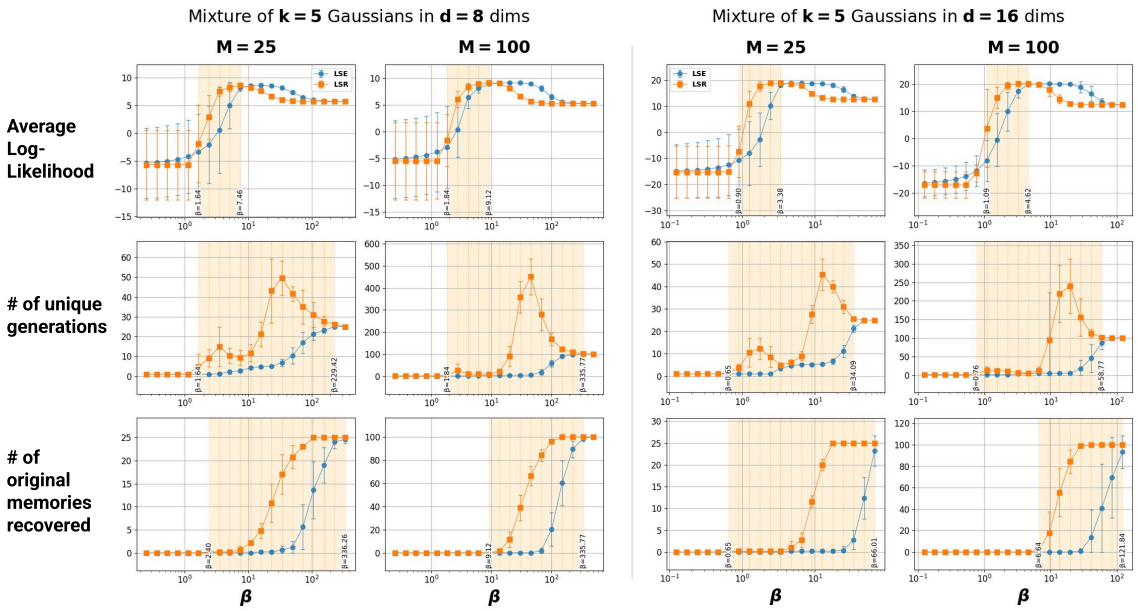

Figure D.1: Comparing $d = 8$ and $d = 16$ for $k = 5$ mixture of gaussians at number of stored patterns $M = 10$ and $M = 100$. Error bars are computed by averaging the results of 5 different random seeds. Regions of β where LSR outperforms LSE on a particular metric (on average) are shaded orange.

Aligning β's across random seeds

We used the same setup for choosing an interesting range of β as we did in Appendix D.1.1. However, over random seeds the value for r_{min} can vary since it is dependent on the random choice of stored patterns. This makes it difficult to plot error bars over an individual β across seeds. To fix this, we use the fact that each experiment has the same number of geometrically spaced β's that start from the same β_{low} and compute statistics averaged across the β's that share an index. The x-axis then represents the β value for each index averaged over seeds.

Determining the uniqueness of minima

To sample the LSE minima, 500 initial points are uniformly sampled from the support boundary around each memory and gradient descent Equation (7.6) is performed for 13000 steps at a cosine-decayed learning rate α from $0.01 \to 0.0001$. However, even after this descent process, there are variations in the retrieved memories due to discrete step size α and floating point precision requiring us to be careful when deciding if two samples are distinct. Generally, memory retrieval is said to converge when $\|\nabla_{\mathbf{x}} E(\mathbf{x})\| < \epsilon$ for some small $\epsilon > 0$, or when the number of iterations T exceeds some threshold at small α. Because we know

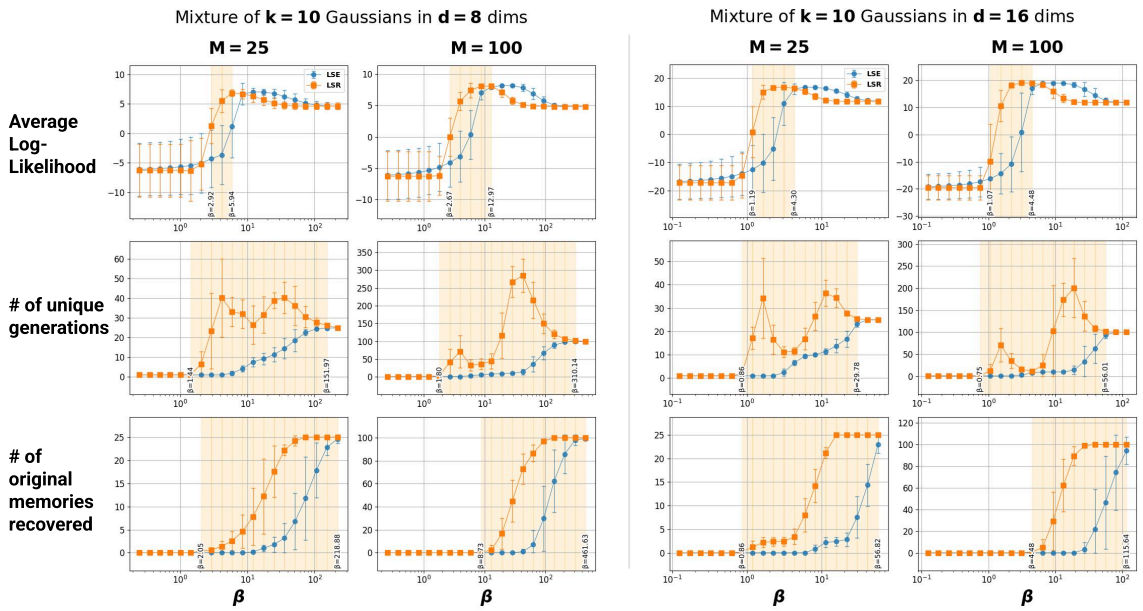

Figure D.2: Comparing $d = 8$ and $d = 16$ for $k = 10$ mixture of gaussians at number of stored patterns $M = 10$ and $M = 100$. Error bars are computed by averaging the results of 5 different random seeds. Regions of β where LSR outperforms LSE on a particular metric (on average) are shaded orange.

the β of the LSE energy, by the properties of the Gaussian kernel we know that two basins merge when their means are within two standard deviations of each other. Thus, we can say that two distinct samples are generated by the same memory if they are within $\frac{2}{\sqrt{\beta}}$ of each other.

When counting the uniqueness of the samples from the LSR energy, we perform the following trick to exactly compute the fixed points of the dynamics. We first compute our "best guess" for the fixed point by performing standard gradient descent according to Equation (7.6) for T steps, at which point $\mathbf{z} := \mathbf{x}^{(T)}$ is close (but not exactly equal) to the fixed point $\mathbf{x}^\star$. We then pass $\mathbf{z}$ to Algorithm 2 to compute the fixed point exactly. With a good initial guess $\mathbf{z}$, Algorithm 2 converges after a single iteration.

Finally, we choose to sample points near the support boundary of each stored pattern because this maximizes the probability that we will end up in a spurious minimum. The size of spurious basins in high dimension can be very small, and the probability of landing in them decreases rapidly with increasing β (see the region of support plot in Figure 7.3).

Algorithm 2: Fixed Point Computation for the LSR Memory Retrieval

Input: Initial guess $\mathbf{z}$, stored patterns $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^M$, inverse temperature β
Output: Fixed point $\mathbf{z}^\star$

Initialize previous point $\mathbf{z}_{\text{prev}} \leftarrow \mathbf{z} + \infty$
while $\mathbf{z}_{prev} \neq \mathbf{z}$ **do**
  $\mathbf{z}_{\text{prev}} \leftarrow \mathbf{z}$
  Compute supports near $\mathbf{z}$ $S(\mathbf{z}) \leftarrow \{\boldsymbol{\xi}_\mu : \|\mathbf{z} - \boldsymbol{\xi}_\mu\| \leq \sqrt{\frac{2}{\beta}}\}$
  Update to mean of support centroids $\mathbf{z} \leftarrow \frac{1}{|S(\mathbf{z})|} \sum_{\boldsymbol{\xi}_\mu \in S(\mathbf{z})} \boldsymbol{\xi}_\mu$
end
return $\mathbf{z}$

D.1.3 Details: Qualitative Reconstructions

The experiments in Figure 7.4 are conducted in two steps. First, we **design an energy** for each dataset. Then, we **discover all memories** for each system.

Designing the energy The DenseAM energies studied in this work are described by a matrix of stored patterns Ξ and an inverse temperature hyperparameter β.

Choosing Ξ. For MNIST, Ξ is obtained as follows: 24 random images from the MNIST training set are normalized to be $[0, 1]$, rasterized into a 784-dim vector, and projected into a 10-dim latent space of a β-VAE trained according to the methods laid out below. These latents become our stored patterns $\Xi_{\text{MNIST}} \in [0, 1]^{24\times 768}$ that we use to parameterize both the LSR energy and the LSE energy. The procedure for Tiny ImageNet [1] is similar. We randomly select 40 images from the dataset, each of shape `(C,H,W)=(3,64,64)`. These samples are passed through a small pretrained VAE called TAESD [2] to produce latents that are of shape `(4,8,8)` which are then rasterized into vectors of shape (256,). Thus, our stored pattern matrix for Tiny ImageNet is $\Xi_{\text{TinyImageNet}} \in \mathbb{R}^{40\times 256}$.

Choosing β Tuning β for the LSR energy is challenging. When β is too small, each stored pattern interacts with all other stored patterns to induce one minimum at the centroid and several minima far from the data distribution; when β is too large, each stored pattern will interact with no other stored patterns and will induce only a single minimum at itself. Neither of these regimes are interesting. For large ranges of β between these limits, the combinatorial search space of possible memories is computationally prohibitive. For this reason, we use a binary search algorithm to choose a β that, on average, causes each stored pattern to interact with approximately 4 other stored patterns (in the 10-dim MNIST case)

and 2 other stored patterns in the 256-dim Tiny ImageNet case. This β encourages the LSR energy to exhibit emergence where it is computationally feasible to enumerate all memories, and we use it for both the LSR and LSE energy experiments. In pseudocode, the search algorithm is given by Algorithm 3.

Training the β-VAE for MNIST. MNIST images are encoded by a β-VAE [126, 247] with a latent dimension of 10. The VAE takes as input the 784-dim rasterized MNIST images. The VAE's encoder and decoder are two layer MLPs configured with LeakyReLU and BatchNorm activations, with a hidden dimension size of 512. Training proceeded for 50 epochs using a learning rate of 1e-3, the Adam optimizer, and a minibatch size of 128. The β of the β-VAE (distinct from the inverse temperature β used by the LSR energy) is set to 4.

Discovering all memories Once we have chosen a β that results in a feasible number of basin interactions K (where the combinatorial search space is tractable), all memories for LSR are discovered by filtering the set of "memory candidates" — the set of centroids formed by at overlapping basins around each stored patterns — to those whose energy gradient is zero. This method is explicitly described in Algorithm 4, which iterates through each stored patterns $\boldsymbol{\xi}_\mu$ and only searches for emergent memories formed by near-enough stored patterns.

All LSE memories are discovered via gradient descent. We initialize queries $\mathbf{x}_\mu = \boldsymbol{\xi}_\mu$ and perform gradient descent according to Equation (7.6) until convergence (for this experiment, we iterated for 20k steps at small step-size $\alpha = 0.002$). The retrieved fixed points are the complete set of LSE memories.

D.1.4 Scaling Number of Stored Patterns

The LSR energy function is simple and its properties hold across any scale of stored patterns. To show this, we store all 60,000 MNIST training images into the LSR energy and select a β for which $\sim$50% of the stored patterns are still retrievable. We select a "seed" image at random and randomly select 15 images from all images whose basins interact with the seed image at the chosen β. This example is shown in Figure D.3.

D.2 Limitations

To enable both emergence and exact memorization, the LSR energy studied in this work comes with certain limitations. Unlike the LSE energy whose gradient remains nonzero

Algorithm 3: Binary Search over β to achieve desired memory interactions

Input: Desired avg. interactions per pattern K, stored patterns $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^M$, max iterations $n_{\max}$

Output: Optimal β^* achieving number of basin interactions K

Compute
  Distance matrix $D_{\mu\nu} \leftarrow \{\|\boldsymbol{\xi}_\mu - \boldsymbol{\xi}_\nu\|\}$
  Binary bounds $(r_{\text{min}}, r_{\text{max}}) \leftarrow (0.5\min(D), 4\max(D))$
  Initial basin radius $r \leftarrow \text{mean}(r_{\text{min}}, r_{\text{max}})$
end
$n_{\text{iter}} \leftarrow 0$
repeat
  Compute avg. number of interacting basins per memory
    $K' \leftarrow \text{mean}_\mu \left(\sum_{\nu=1}^M \mathbb{1}[D_{\mu\nu} \leq 2r]\right)$
  Update binary search conditions
    $r_{\text{min}} \leftarrow r$ if $K' < K$
    $r_{\text{max}} \leftarrow r$ if $K' > K$
    $r \leftarrow \text{mean}(r_{\text{min}}, r_{\text{max}})$
  $n_{\text{iter}} \leftarrow n_{\text{iter}} + 1$
until $K' = K$ or $n_{iter} \geq n_{max}$
$\beta^* \leftarrow 2/r^2$
return β^*

regardless of the query's distance from the stored patterns, the gradient of LSR energy vanishes exactly outside the support of the stored pattern. This makes gradient-based retrieval ineffective when the query lies far from any memorized pattern (though one can easily introduce a query-dependent temperature parameter $\beta(\mathbf{x})$ that dynamically adjusts to ensure the query lies within at least one basin of attraction). Alternatively, one can create a "hybrid" energy function that combines the LSE and LSR energies, taking advantage of LSE's non-zero gradient everywhere outside LSR's support around the stored patterns. Such temperature-tuned DenseAMs and hybrid energies are of independent interest and we leave their systematic study to future work.

Additionally, we want to emphasize that LSR can only create emergent memories precisely at the centroid of overlapping basins between stored patterns. This makes it possible to quickly and exactly retrieve memories (see Algorithm 4), but it also means that the "creativity" of this model is limited to a predictable subset of the convex hull of the stored patterns. The apparent novelty and creativity of Figure 7.4 is strongly aided by the semantic structure contained the VAEs' latent space.

Algorithm 4: Discover local minima of the LSR energy at a specific β.

Input: Stored patterns $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^M$, inverse temperature β, gradient norm threshold δ near 0

Output: Set of LSR memories $\mathcal{X}^*$.

Compute
  Distance matrix $D_{\mu\nu} \leftarrow \{\|\boldsymbol{\xi}_\mu - \boldsymbol{\xi}_\nu\|\}$
  Basin radius $r \leftarrow \sqrt{2/\beta}$
end
Initialize set of local minima $\mathcal{X}^* \leftarrow \emptyset$
for $\mu \in [\![M]\!]$ **do**
  Compute set of interacting neighbors $\mathbf{X}_\mu \leftarrow \{\boldsymbol{\xi}_\nu : D_{\mu\nu} \leq 2r\}$
  Compute the set of all non-empty subsets $\mathcal{C} \leftarrow \{S \subseteq \mathbf{X}_\mu : \text{size}(S) > 0\}$
  for $S \in \mathcal{C}$ **do**
    Compute centroid of neighbors $\bar{\mathbf{x}}_S \leftarrow \text{mean}(S)$
    Compute $\bar{\mathbf{x}}_S$ neighbors $T_S \leftarrow \{\boldsymbol{\xi}_\nu : \|\boldsymbol{\xi}_\nu - \bar{\mathbf{x}}_S\| \leq r\}$
    if $\|\nabla E_{\text{LSR}}(\bar{\mathbf{x}}_S)\| < \delta$ **&** $E_{\text{LSR}}(\bar{\mathbf{x}}_S) < \infty$ **then**
      Update set of local minima $\mathcal{X}^* \leftarrow \mathcal{X}^* \cup \{\bar{\mathbf{x}}_S\}$
    end
  end
end
return $\mathcal{X}^*$

D.3 Emergent Memories, Hallucination, and Biological Implementation

D.3.1 On the Relationship Between Emergent Memories and Hallucination

In everyday use of LLMs, we call it a *hallucination* when the model confidently produces an incorrect fact, especially when we know the model is capable of producing the correct fact in other settings. To formalize this idea into the context of emergent memories, we must imagine a setting where the model has an explicit energy function and a known set of stored "facts."

This is the setting of AM as studied in this work, where the outputs of our model are always energy minima (memories). A "hallucination" can occur when two or more stored facts interfere, leading the system to settle into an emergent minimum that looks meaningful (i.e., it lies near the "factual manifold" that generated our stored patterns) but does not correspond to any true, stored fact. Emergent memories are precisely these "interference minima" when they coexist with the original stored patterns. By definition, they are not stored facts, yet they can resemble meaningful combinations of them — making them the memory-system analog of hallucinations.

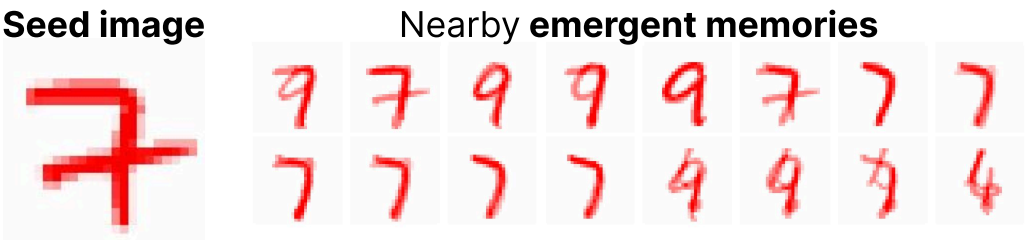

Figure D.3: Sampling emergent memories near a **seed image** when all 60k MNIST training images are stored into the LSR energy. Left: Random seed image, which is a preserved memory. Right: 16 randomly sampled emergent memories formed by the seed image's interactions with other stored patterns (at $\beta = 0.11$). Because the seed image interacts with the basins of $\sim$7.3k other stored patterns, these emergent memories represent a *tiny sample* of the total emergent memories near the seed image.

D.3.2 On a Biological Implementation of Emergent Memories

Dense AMs permit a standard mapping onto "biological" networks by introducing auxiliary hidden neurons following the method of [75]. Thus, the feature neurons in both LSE and LSR models can be augmented with auxiliary hidden neurons, resulting in a model with only pair-wise neuron interactions and bipartite connectivity between feature and hidden layers. This augmented model is more biological compared to the model considered in the main text. The model defined by the energy Equation (7.3) can be obtained by integrating out these auxiliary hidden neurons, which means LSR energy will still have the same emergent behavior. The augmented model will still have some un-biological aspects because of the weight symmetry between forward and backward projections, which is necessary to ensure that the network has a global energy function.

D.4 Other Compact-Support Kernels

We show different kernels that are typically used for KDE and their efficiency relative to the Epanechnikov kernel in Figure D.4. See the explanation on optimal kernel density estimation in § 7.2 for more details.

Though the main analysis focuses on the Epanechnikov kernel, we find that the phenomenon of emergence (Definitions 7.2 and 7.3) is not limited to the Epanechnikov kernel. We state our findings for other popular kernels below and graphically in Figure D.5. In summary, all compact support kernels are capable of producing emergent memories *except* for the TriWeight kernel (meanwhile, the uniform kernel is impractical for AM). Details below:

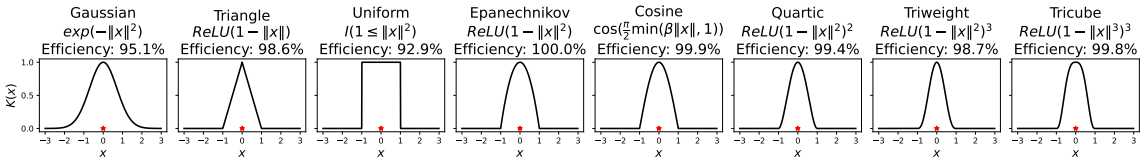

Figure D.4: Different kernels used in KDE with their expression and KDE efficiency relative to the Epanechnikov kernel (*higher is better*, see text for details). The center of each kernel is marked with a red $\star$. To highlight the shape of the kernel, we have removed any scaling in the kernel expression. Note that all above kernels except Gaussian have finite support. The Epanechnikov kernel has the highest efficiency (100%). While the Gaussian kernel is extremely popular, and it is more efficient (95.1%) than the Uniform kernel (92.9%), there are various other kernels with better efficiency.

1. **Triangle kernel** — The triangle kernel exhibits very interesting emergence behavior. A perfectly *flat* energy manifold is formed where two or more basins overlap. If the energy of this manifold is lower than the energy of the stored patterns, the stored patterns "merge" and are no longer retrievable. If the energy of the manifold is higher, both original patterns are preserved and we observe a local *emergent manifold* of memory.

2. **Uniform kernel** — A uniform kernel produces an energy landscape that is impractical for AM because the gradient is zero everywhere except at the discontinuities. Emergent minima exist according to Definition 7.3 and have exclusively lower energy than the stored patterns.

3. **Triweight kernel** — We were not able to find emergence at any temperature with the triweight kernel, despite its compact support. Indeed, looking at the $\beta = 0.19$ row of Figure D.5, we see that the transition of two basins merging results in a single, almost flat minimum. This phenomenon of compact support without emergence (or perhaps, emergence that is limited to an extremely narrow range of β that we were not able to find) requires further investigation.

4. **Quartic kernel** — The quartic kernel produces smoother energy landscapes than the Epanechnikov does, and emergence appears within a very narrow range of β. Unlike the Epanechnikov kernel, overlapping basins do not guarantee emergent minima, and the emergent minimum is unlikely to have lower energy than the stored patterns.

5. **Tricube kernel** — The tricube kernel behaves like the Quartic kernel, but emergent

memories are likely to have lower energy than the stored patterns. The Tricube kernel has an interesting property where local minima flatten right before basins merge. The resulting energy landscape is smoother than that of the Epanechnikov kernel.

6. **Cosine kernel** — The cosine kernel looks remarkably like the Epanechnikov kernel, and its emergence properties are almost identical to that of LSR. However, it appears that Cosine-emergent memories have slightly higher energy than their Epanechnikov counterpart.

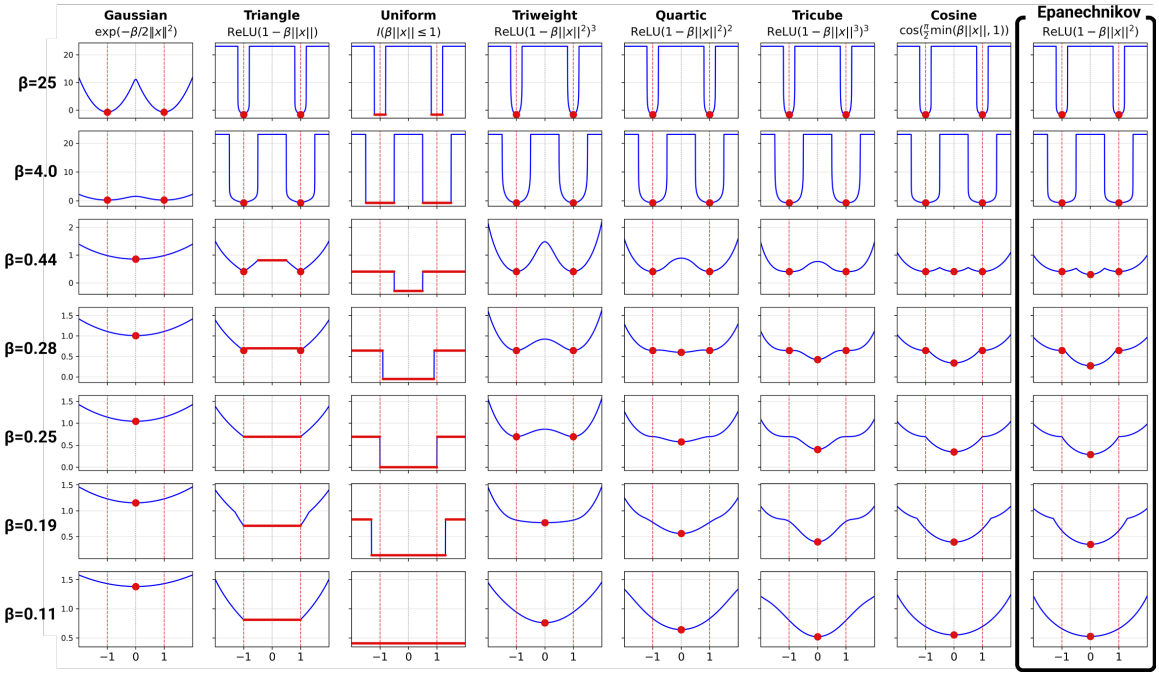

Figure D.5: Comparing emergence across different choices of kernel in the DenseAM energy function. Emergent memories are highlighted in red, where manifolds are shown as a flat line and single points as larger dots. Interestingly, all compact kernels exhibit some form of emergence *except* the TriWeight kernel.

REFERENCES

[1] Y. Le and X. S. Yang, "Tiny ImageNet visual recognition challenge," 2015.

[2] O. B. Bohan, *Tiny autoencoder for stable diffusion*, https://github.com/madebyollin/taesd, 2023.

[3] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon et al., Eds., vol. 30, Curran Associates, Inc., 2017.

[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI, Tech. Rep., 2019.

[5] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 2256–2265.

[6] Y. Song and S. Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[7] S.-I. Amari, "Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements," *IEEE Transactions on Computers*, vol. C-21, no. 11, pp. 1197–1206, Nov. 1972.

[8] S.-I. Amari, "Neural theory of association and concept-formation," *Biological cybernetics*, vol. 26, no. 3, pp. 175–185, 1977.

[9] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities.," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.

[10] J. J. Hopfield, "Neurons With Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 81, pp. 3088–92, Jun. 1984.

[11] B. Hoover et al., "Energy Transformer," *Advances in neural information processing systems*, vol. 36, pp. 27 532–27 559, 2023. arXiv: 2302.07253.

[12] N. Dehmamy, B. Hoover, B. Saha, L. Kozachkov, J.-J. Slotine, and D. Krotov, "Nrgpt: An energy-based alternative for gpt," in *International Conference on Learning Representations (ICLR)*, 2026. arXiv: 2512.16762.

[13] B. Hoover, H. Strobelt, D. Krotov, J. Hoffman, Z. Kira, and D. H. Chau, *Memory in Plain Sight: A Survey of the Uncanny Resemblances between Diffusion Models and Associative Memories*, Sep. 2023. arXiv: 2309.16750 [cs, math].

[14] B. Hoover, D. H. Chau, H. Strobelt, P. Ram, and D. Krotov, "Dense associative memory through the lens of random features," *Advances in Neural Information Processing Systems*, vol. 37, pp. 23 549–23 576, 2024. arXiv: 2410.24153.

[15] B. Hoover, Z. Shi, K. Balasubramanian, D. Krotov, and P. Ram, *Dense Associative Memory with Epanechnikov Energy*, Jun. 2025. arXiv: 2506.10801 [cs].

[16] D. Krotov, B. Hoover, P. Ram, and B. Pham, *Modern Methods in Associative Memory*, Jul. 2025. arXiv: 2507.06211 [cs.LG].

[17] B. Hoover, D. H. Chau, H. Strobelt, and D. Krotov, "A Universal Abstraction for Hierarchical Hopfield Networks," in *The Symbiosis of Deep Learning and Differential Equations II*, Oct. 2022.

[18] T. Kohonen, *Content-Addressable Memories* (Springer Series in Information Sciences), 2nd ed. Springer Berlin Heidelberg, 1987, vol. 1, ISBN: 978-3-540-17625-1.

[19] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[20] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[21] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[22] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-holographic associative memory," *Nature*, vol. 222, no. 5197, pp. 960–962, 1969.

[23] M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 815–826, Sep. 1983.

[24] D. J. Amit, H. Gutfreund, and H. Sompolinsky, "Storing infinite numbers of patterns in a spin-glass model of neural networks," *Physical Review Letters*, vol. 55, no. 14, pp. 1530–1533, 1985.

[25] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the hopfield associative memory," *IEEE transactions on Information Theory*, vol. 33, no. 4, pp. 461–482, 1987.

[26] D. Krotov and J. J. Hopfield, "Dense associative memory for pattern recognition," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, Curran Associates, Inc., 2016.

[27] M. Demircigil, J. Heusel, M. Löwe, S. Upgang, and F. Vermet, "On a Model of Associative Memory with Huge Storage Capacity," *Journal of Statistical Physics*, vol. 168, no. 2, pp. 288–299, Jul. 2017.

[28] H. Ramsauer et al., "Hopfield networks is all you need," in *International Conference on Learning Representations*, 2021.

[29] P. Kanerva, *Sparse Distributed Memory*. Cambridge, MA: MIT Press, 1988, ISBN: 9780262111324.

[30] L. A. Jaeckel, "An alternative design for a sparse distributed memory," Jul. 1989.

[31] J. Weston, S. Chopra, and A. Bordes, *Memory Networks*, Nov. 2015. arXiv: 1410.3916 [cs, stat].

[32] A. H. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1400–1409. arXiv: 1606.03126.

[33] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, "Boltzmann machines: Constraint satisfaction networks that learn," Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-84-119, 1984.

[34] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, no. 1, pp. 147–169, Jan. 1985.

[35] G. E. Hinton, "Training Products of Experts by Minimizing Contrastive Divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002.

[36] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. J. Huang, "A Tutorial on Energy-Based Learning," in *Predicting Structured Data*, G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, and B. Taskar, Eds., MIT Press, 2006, pp. 191–246.

[37] Y. Song and D. P. Kingma, *How to Train Your Energy-Based Models*, Feb. 2021. arXiv: 2101.03288 [cs, stat].

[38] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th International Conference on Machine Learning*, ser. ICML'11, 2011, pp. 681–688.

[39] Y. Du and I. Mordatch, "Implicit generation and modeling with energy based models," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 3603–3613. arXiv: 1903.08689.

[40] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, 1984.

[41] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky, "Your classifier is secretly an energy based model and you should treat it like one," in *International Conference on Learning Representations*, Sep. 2019.

[42] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching.," *Journal of Machine Learning Research*, vol. 6, no. 24, pp. 695–709, 2005.

[43] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, Jul. 2011.

[44] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*, 2021.

[45] Y. Du et al., "Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202, PMLR, Jul. 2023, pp. 8489–8510. arXiv: 2302.11552.

[46] Y. Du, S. Li, J. Tenenbaum, and I. Mordatch, "Improved Contrastive Divergence Training of Energy-Based Models," in *Proceedings of the 38th International Conference on Machine Learning*, PMLR, Jul. 2021, pp. 2837–2848.

[47] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 6840–6851.

[48] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[49] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser, "Universal transformers," in *International Conference on Learning Representations*, 2019.

[50] L. Yang, K. Lee, R. D. Nowak, and D. Papailiopoulos, "Looped transformers are better at learning learning algorithms," in *International Conference on Learning Representations*, 2024.

[51] I. Rodkin, Y. Kuratov, A. Bulatov, and M. Burtsev, "Associative recurrent memory transformer," *arXiv preprint arXiv:2407.04841*, 2024.

[52] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[53] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[54] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proceedings of the 21st International Conference on Neural Information Processing Systems*, ser. NIPS'07, Red Hook, NY, USA: Curran Associates Inc., Dec. 2007, pp. 1177–1184, ISBN: 978-1-60560-352-0.

[55] K. Choromanski et al., "Rethinking attention with performers," *Proceedings of ICLR*, 2020.

[56] G. Iatropoulos, J. Brea, and W. Gerstner, "Kernel memory networks: A unifying framework for memory modeling," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.

[57] J. Y.-C. Hu, D. Yang, D. Wu, C. Xu, B.-Y. Chen, and H. Liu, "On sparse modern hopfield model," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[58] S. J. R. D. Santos, V. Niculae, D. C. Mcnamee, and A. Martins, "Sparse and structured hopfield networks," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235, PMLR, Jul. 2024, pp. 43 368–43 388.

[59] D. Wu, J. Y.-C. Hu, T.-Y. Hsiao, and H. Liu, "Uniform memory retrieval with larger capacity for modern hopfield models," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235, PMLR, Jul. 2024, pp. 53 471–53 514.

[60] M. P. Wand and M. C. Jones, *Kernel Smoothing*. New York: Chapman and Hall/CRC, Nov. 1994, ISBN: 978-0-429-17059-1.

[61] B. W. Silverman, *Density estimation for statistics and data analysis* (Monographs on Statistics and Applied Probability). London: Chapman & Hall, 1986, vol. 26, ISBN: 9780412246203.

[62] V. A. Epanechnikov, "Non-parametric estimation of a multivariate probability density," *Theory of Probability & Its Applications*, vol. 14, no. 1, pp. 153–158, 1969.

[63] E. Mammen, "On qualitative smoothness of kernel density estimates," *Statistics: a journal of theoretical applied statistics*, vol. 26, no. 3, pp. 253–267, 1995.

[64] B. Geshkovski, P. Rigollet, and Y. Sun, "On the number of modes of gaussian kernel density estimators," *arXiv preprint arXiv:2412.09080*, 2024. arXiv: 2412.09080.

[65] O. Press, N. A. Smith, and O. Levy, "Improving transformer models by reordering their sublayers," *arXiv preprint arXiv:1911.03864*, 2019.

[66] Y. Lu et al., "Understanding and improving transformer from a multi-particle dynamic system point of view," *arXiv preprint arXiv:1906.02762*, 2019.

[67] D. So, Q. Le, and C. Liang, "The evolved transformer," in *International Conference on Machine Learning*, PMLR, 2019, pp. 5877–5886.

[68] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *arXiv preprint arXiv:2106.04554*, 2021.

[69] W. Yu et al., "Metaformer is actually what you need for vision," *arXiv preprint arXiv:2111.11418*, 2021.

[70] D. Krotov, "A new frontier for Hopfield networks," *Nature Reviews Physics*, vol. 5, no. 7, pp. 366–367, Jul. 2023.

[71] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2020.

[72] Y. Yang, Z. Huang, and D. Wipf, "Transformers from an optimization perspective," *arXiv preprint arXiv:2205.13891*, 2022. arXiv: 2205.13891.

[73] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 794–816, 2016.

[74] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[75] D. Krotov and J. J. Hopfield, "Large associative memory problem in neurobiology and machine learning," in *International Conference on Learning Representations*, 2021.

[76] F. Tang and M. Kopp, "A remark on a paper of krotov and hopfield," *arXiv preprint arXiv:2105.15034*, 2021.

[77] D. Krotov, *Hierarchical Associative Memory*, Jul. 2021. arXiv: 2107.06446 [cond-mat, q-bio, stat].

[78] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[79] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Springer, 2014, pp. 818–833.

[80] J. Tang, J. Li, Z. Gao, and J. Li, "Rethinking graph neural networks for anomaly detection," *arXiv preprint arXiv:2205.15508*, 2022.

[81] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[82] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proceedings of the 2019 SIAM International Conference on Data Mining*, SIAM, 2019, pp. 594–602.

[83] Z. Peng, M. Luo, J. Li, L. Xue, and Q. Zheng, "A deep multi-view framework for anomaly detection on attributed networks," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[84] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*, PMLR, 2019, pp. 6861–6871.

[85] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, 2015, pp. 985–994.

[86] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 897–908.

[87] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng, "Alleviating the inconsistency problem of applying graph neural network to fraud detection," in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 2020, pp. 1569–1572.

[88] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 315–324.

[89] Y. Liu et al., "Pick and choose: A gnn-based imbalanced learning approach for fraud detection," in *Proceedings of the Web Conference 2021*, 2021, pp. 3168–3177.

[90] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," *arXiv preprint arXiv:2012.09699*, 2020.

[91] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240.

[92] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," *CoRR*, vol. abs/2007.08663, 2020. arXiv: 2007.08663.

[93] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, *Benchmarking graph neural networks*, 2022. arXiv: 2003.00982 [cs.LG].

[94] Q. Zhao and Y. Wang, "Learning metrics for persistence-based summaries and applications for graph classification," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019.

[95] M. Balcilar, G. Renton, P. Héroux, B. Gaüzère, S. Adam, and P. Honeine, "Bridging the gap between spectral and spatial domains in graph neural networks," *CoRR*, vol. abs/2003.11702, 2020. arXiv: 2003.11702.

[96] Z. Zhang et al., "Hierarchical multi-view graph pooling with structure learning," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.

[97] D. Q. Nguyen, T. D. Nguyen, and D. Phung, "Universal graph transformer self-attention networks," in *Companion Proceedings of the Web Conference 2022*, ser. WWW '22, Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 193–196, ISBN: 9781450391306.

[98] M. Domingue, R. Dhamdhere, N. D. Harish Kanamarlapudi, S. Raghupathi, and R. Ptucha, "Evolution of graph classifiers," in *2019 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, 2019, pp. 1–5.

[99] M. Yang, Y. Shen, R. Li, H. Qi, Q. Zhang, and B. Yin, "A new perspective on the effects of spectrum in graph neural networks," in *Proceedings of the 39th International Conference on Machine Learning*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., ser. Proceedings of Machine Learning Research, vol. 162, PMLR, Jul. 2022, pp. 25 261–25 279.

[100] F. Orsini, P. Frasconi, and L. De Raedt, "Graph invariant kernels," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15, Buenos Aires, Argentina: AAAI Press, 2015, pp. 3756–3762, ISBN: 9781577357384.

[101] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Hierarchical representation learning in graph neural networks with node decimation pooling," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[102] E. Ranjan, S. Sanyal, and P. P. Talukdar, "ASAP: Adaptive structure aware pooling for learning hierarchical graph representations," *arXiv preprint arXiv:1911.07979*, 2019.

[103] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., *Improving language understanding by generative pre-training*, 2018.

[104] J. Von Oswald et al., "Transformers learn in-context by gradient descent," in *International Conference on Machine Learning*, PMLR, 2023, pp. 35 151–35 174.

[105] K. Ahn, X. Cheng, H. Daneshmand, and S. Sra, "Transformers learn to implement preconditioned gradient descent for in-context learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[106] A. Gladstone et al., "Energy-based transformers are scalable learners and thinkers," *arXiv preprint arXiv:2507.02092*, 2025.

[107] B. Geshkovski, C. Letrouit, Y. Polyanskiy, and P. Rigollet, "A mathematical perspective on transformers," *arXiv preprint arXiv:2312.10794*, 2023.

[108] B. Geshkovski, C. Letrouit, Y. Polyanskiy, and P. Rigollet, "The emergence of clusters in self-attention dynamics," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[109] B. Wang and A. Komatsuzaki, *Gpt-j-6b: A 6 billion parameter autoregressive language model*, 2021.

[110] A. Chowdhery et al., "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[111] B. He and T. Hofmann, "Simplifying transformer blocks," in *The Twelfth International Conference on Learning Representations*, 2024.

[112] J. Bai et al., "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.

[113] A. Grattafiori et al., "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[114] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Advances in neural information processing systems*, vol. 32, 2019.

[115] X. S. Huang, F. Perez, J. Ba, and M. Volkovs, "Improving transformer optimization through better initialization," in *International Conference on Machine Learning*, PMLR, 2020, pp. 4475–4483.

[116] H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, and F. Wei, "Deepnet: Scaling transformers to 1,000 layers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 10, pp. 6761–6774, 2024.

[117] N. Nangia and S. Bowman, "Listops: A diagnostic dataset for latent tree learning," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, 2018, pp. 92–99.

[118] D. Hendrycks et al., "Measuring massive multitask language understanding," *arXiv preprint arXiv:2009.03300*, 2020.

[119] Y. Song and S. Ermon, "Improved Techniques for Training Score-Based Generative Models," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 12 438–12 448.

[120] P. Dhariwal and A. Q. Nichol, "Diffusion Models Beat GANs on Image Synthesis," in *Advances in Neural Information Processing Systems*, Nov. 2021.

[121] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, *Hierarchical Text-Conditional Image Generation with CLIP Latents*, Apr. 2022. arXiv: 2204.06125 [cs].

[122] C. Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," in *Advances in Neural Information Processing Systems*, May 2022.

[123] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[124] A. Nichol et al., *GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models*, Mar. 2022. arXiv: 2112.10741 [cs].

[125] I. Goodfellow et al., "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[126] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 2013.

[127] D. McAllester, *On the Mathematics of Diffusion Models*, Feb. 2023. arXiv: 2301.11108 [cs, math].

[128] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models," in *International Conference on Learning Representations*, Sep. 2018.

[129] P. Lippe, *Tutorial 7: Deep Energy-Based Generative Models*, Sep. 2021.

[130] Y. Takagi and S. Nishimoto, "High-resolution image reconstruction with latent diffusion models from human brain activity," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 453–14 463.

[131] J. C. R. Whittington, J. Warren, and T. E. Behrens, "Relating transformers to models and neural representations of the hippocampal formation," in *International Conference on Learning Representations*, 2022.

[132] J. Fu et al., *Pattern completion and disruption characterize contextual modulation in mouse visual cortex*, Mar. 2023.

[133] F. Ozcelik and R. VanRullen, *Brain-Diffuser: Natural scene reconstruction from fMRI signals using generative latent diffusion*, Mar. 2023. arXiv: 2303.05334 [cs, q-bio].

[134] A. Y. Wang, K. Kay, T. Naselaris, M. J. Tarr, and L. Wehbe, *Incorporating natural language into vision models improves prediction and understanding of higher visual cortex*, Sep. 2022.

[135] A. Majumdar et al., "Where are we in the search for an Artificial Visual Cortex for Embodied Intelligence?" In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, Mar. 2023.

[136] L. Kozachkov, K. V. Kastanenka, and D. Krotov, "Building transformers from neurons and astrocytes," *Proceedings of the National Academy of Sciences*, vol. 120, no. 34, e2219150120, Aug. 2023.

[137] L. Yang et al., *Diffusion Models: A Comprehensive Survey of Methods and Applications*, Oct. 2022. arXiv: 2209.00796 [cs].

[138] E. J. Ma, *A Pedagogical Introduction to Score Models*, Apr. 2022.

[139] C. Luo, *Understanding Diffusion Models: A Unified Perspective*, Aug. 2022. arXiv: 2208.11970 [cs].

[140] K. Kreis, R. Gao, and A. Vahdat, *Denoising Diffusion-based Generative Modeling: Foundations and Applications*, CVPR 2022, 2023.

[141] J. Thorton and D. Bortoli, *What's the score? – Review of latest Score Based Generative Modeling papers.* 2023.

[142] H. K. Sulehria and Y. Zhang, "Hopfield neural networks: A survey," in *Proceedings of the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases*, Citeseer, vol. 6, 2007, pp. 125–130.

[143] A. G. Hanlon, "Content-Addressable and Associative Memory Systems a Survey," *IEEE Transactions on Electronic Computers*, vol. EC-15, no. 4, pp. 509–521, Aug. 1966.

[144] B. D. C. N. Prasad, P. E. S. N. K. Prasad, S. Yeruva, and P. S. R. Murty, "A Study on Associative Neural Memories," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 1, no. 6, Feb. 2012.

[145] V. I. Gritsenko, D. A. Rachkovskij, A. A. Frolov, R. W. Gayler, D. Kleyko, and E. Osipov, "Neural distributed autoassociative memories: A survey," *CoRR*, vol. abs/1709.00848, 2017. arXiv: 1709.00848.

[146] S. S.A, A. Swedha, and D. Naveen, "Survey of Content Addressable Memory," vol. 06, p. 1516, Feb. 2018.

[147] T. Poggio, "From Associative Memories to Deep Networks," 2021.

[148] L. Ambrogioni, "In Search of Dispersed Memories: Generative Diffusion Models Are Associative Memory Networks," *Entropy*, vol. 26, no. 5, 2024.

[149] R. S. Zimmermann, L. Schott, Y. Song, B. A. Dunn, and D. A. Klindt, "Score-Based Generative Classifiers," in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, Dec. 2021.

[150] T. Salimans and J. Ho, "Should EBMs model the energy or the score?" In *Energy Based Models Workshop - ICLR 2021*, Apr. 2021.

[151] Y. Song, S. Garg, J. Shi, and S. Ermon, "Sliced score matching: A scalable approach to density and score estimation," in *Uncertainty in Artificial Intelligence*, PMLR, 2020, pp. 574–584.

[152] M. Raphan and E. Simoncelli, "Least Squares Estimation Without Priors or Supervision," *Neural computation*, vol. 23, pp. 374–420, Feb. 2011.

[153] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the Design Space of Diffusion-Based Generative Models," in *Advances in Neural Information Processing Systems*, May 2022.

[154] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency Models," in *International Conference for Machine Learning*, Jan. 2023.

[155] G. Raya and L. Ambrogioni, "Spontaneous symmetry breaking in generative diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[156] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.

[157] N. Carlini et al., "Extracting training data from diffusion models," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 5253–5270.

[158] V. Voleti, C. Pal, and A. Oberman, *Score-based Denoising Diffusion with Non-Isotropic Gaussian Noise Models*, Nov. 2022. arXiv: 2210.12254 [cs].

[159] E. Nachmani, R. S. Roman, and L. Wolf, *Non Gaussian Denoising Diffusion Models*, Jun. 2021. arXiv: 2106.07582 [cs, eess].

[160] A. Bansal et al., *Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise*, Aug. 2022. arXiv: 2208.09392 [cs].

[161] F. Bao, C. Li, Y. Cao, and J. Zhu, "All are Worth Words: A ViT Backbone for Score-based Diffusion Models," in *NeurIPS 2022 Workshop on Score-Based Methods*, Nov. 2022.

[162] X. Yang, S.-M. Shih, Y. Fu, X. Zhao, and S. Ji, *Your ViT is Secretly a Hybrid Discriminative-Generative Diffusion Model*, Aug. 2022. arXiv: 2208.07791 [cs].

[163] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[164] B. Millidge, T. Salvatori, Y. Song, T. Lukasiewicz, and R. Bogacz, "Universal Hopfield Networks: A General Framework for Single-Shot Associative Memory Models," *Proceedings of machine learning research*, vol. 162, pp. 15 561–15 583, Jul. 2022.

[165] T. F. Burns and T. Fukai, "Simplicial hopfield networks," in *The Eleventh International Conference on Learning Representations*, 2023.

[166] Y. Liang et al., *Can a Fruit Fly Learn Word Embeddings?* Mar. 2021. arXiv: 2101.06887 [cs, q-bio, stat].

[167] M. Widrich et al., "Modern Hopfield Networks and Attention for Immune Repertoire Classification," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 18 832–18 845.

[168] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, Jan. 2017. arXiv: 1412.6980 [cs].

[169] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1-3, pp. 503–528, Aug. 1989.

[170] J. Kaplan et al., *Scaling Laws for Neural Language Models*, Jan. 2020. arXiv: 2001.08361 [cs, stat].

[171] A. Nichol and P. Dhariwal, *Improved Denoising Diffusion Probabilistic Models*, Feb. 2021. arXiv: 2102.09672 [cs, stat].

[172] V. Cabannes, E. Dohmatob, and A. Bietti, *Scaling Laws for Associative Memories*, Oct. 2023. arXiv: 2310.02984 [cs, stat].

[173] X. Niu, B. Bai, L. Deng, and W. Han, "Beyond scaling laws: Understanding transformer performance with associative memory," *arXiv preprint arXiv:2405.08707*, 2024.

[174] R. R. Curtin, P. Ram, and A. G. Gray, "Fast exact max-kernel search," in *Proceedings of the 2013 SIAM International Conference on Data Mining*, SIAM, 2013, pp. 1–9.

[175] R. R. Curtin and P. Ram, "Dual-tree fast exact max-kernel search," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 7, no. 4, pp. 229–253, 2014.

[176] C. Leslie, E. Eskin, and W. S. Noble, "The spectrum kernel: A string kernel for svm protein classification," in *Biocomputing 2002*, World Scientific, 2001, pp. 564–575.

[177] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.

[178] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, "Predicting time series with support vector machines," in *International conference on artificial neural networks*, Springer, 1997, pp. 999–1004.

[179] D. Jain et al., "Mnemosyne: Learning to train transformers with transformers," in *NeurIPS*, 2023.

[180] J. Y.-C. Hu, B.-Y. Chen, D. Wu, F. Ruan, and H. Liu, "Nonparametric modern hopfield models," *arXiv preprint arXiv:2404.03900*, 2024.

[181] H. Chaudhry, J. Zavatone-Veth, D. Krotov, and C. Pehlevan, "Long sequence hopfield memory," *Advances in Neural Information Processing Systems*, vol. 36, pp. 54 300–54 340, 2023.

[182] D. Wu, J. Y.-C. Hu, W. Li, B.-Y. Chen, and H. Liu, "STanhop: Sparse tandem hopfield model for memory-enhanced time series prediction," in *The Twelfth International Conference on Learning Representations*, 2024.

[183] J. Y.-C. Hu, T. Lin, Z. Song, and H. Liu, "On computational limits of modern hopfield models: A fine-grained complexity analysis," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235, PMLR, Jul. 2024, pp. 19 327–19 343.

[184] M. Negri, C. Lauditi, G. Perugini, C. Lucibello, and E. M. Malatesta, "Random feature hopfield networks generalize retrieval to previously unseen examples," in *Associative Memory & Hopfield Networks in 2023*, 2023.

[185] M. Negri, C. Lauditi, G. Perugini, C. Lucibello, and E. Malatesta, "Storage and learning phase transitions in the random-features hopfield model," *Physical Review Letters*, vol. 131, no. 25, p. 257 301, 2023.

[186] B. Saha, D. Krotov, M. J. Zaki, and P. Ram, "End-to-end differentiable clustering with associative memories," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202, PMLR, Jul. 2023, pp. 29 649–29 670.

[187] R. Schaeffer et al., *Bridging associative memory and probabilistic modeling*, 2024. arXiv: 2402.10202 [cs.LG].

[188] L. Kozachkov, J.-J. Slotine, and D. Krotov, "Neuron–astrocyte associative memory," *Proceedings of the National Academy of Sciences*, vol. 122, no. 21, e2417788122, May 2025.

[189] L. Bottou, *Large-scale kernel machines*. MIT press, 2007.

[190] V. Likhosherstov, K. M. Choromanski, K. A. Dubey, F. Liu, T. Sarlos, and A. Weller, "Dense-exponential random features: Sharp positive estimators of the gaussian kernel," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[191] P. Kar and H. Karnick, "Random feature maps for dot product kernels," in *Artificial intelligence and statistics*, PMLR, 2012, pp. 583–591.

[192] N. Pham and R. Pagh, "Fast and scalable polynomial kernels via explicit feature maps," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 239–247.

[193] R. Hamid, Y. Xiao, A. Gittens, and D. DeCoste, "Compact random feature maps," in *International conference on machine learning*, PMLR, 2014, pp. 19–27.

[194] K. M. Choromanski, M. Rowland, and A. Weller, "The unreasonable effectiveness of structured random orthogonal embeddings," *Advances in neural information processing systems*, vol. 30, 2017.

[195] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009, Technical report.

[196] D. Krotov and J. Hopfield, "Dense associative memory is robust to adversarial inputs," *Neural computation*, vol. 30, no. 12, pp. 3151–3167, 2018.

[197] C. Lucibello and M. Mézard, "Exponential capacity of dense associative memories," *Physical Review Letters*, vol. 132, no. 7, p. 077 301, 2024.

[198] M. Belkin, A. Rakhlin, and A. B. Tsybakov, "Does data interpolation contradict statistical optimality?" In *The 22nd International Conference on Artificial Intelligence and Statistics*, PMLR, 2019, pp. 1611–1619.

[199] M. Belkin, "Fit without fear: Remarkable mathematical phenomena of deep learning through the prism of interpolation," *Acta Numerica*, vol. 30, pp. 203–248, 2021.

[200] P. L. Bartlett, A. Montanari, and A. Rakhlin, "Deep learning: A statistical viewpoint," *Acta numerica*, vol. 30, pp. 87–201, 2021.

[201] L. Devroye and G. Lugosi, *Combinatorial methods in density estimation*. Springer Science & Business Media, 2001.

[202] H.-G. Müller, "Smooth optimum kernel estimators of densities, regression curves and modes," *The Annals of Statistics*, pp. 766–774, 1984.

[203] S.-I. Amari and K. Maginu, "Statistical neurodynamics of associative memory," *Neural Networks*, vol. 1, no. 1, pp. 63–73, 1988.

[204] A. V. Robins and S. J. McCallum, "A robust method for distinguishing between learned and spurious attractors," *Neural Networks*, vol. 17, no. 3, pp. 313–326, 2004.

[205] A. Barra, G. Genovese, P. Sollich, and D. Tantari, "Phase diagram of restricted boltzmann machines and generalized hopfield networks with arbitrary priors," *Physical Review E*, vol. 97, no. 2, p. 022 310, 2018.

[206] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[207] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.

[208] R. Wang and Y. Du, "Equilibrium matching: Generative modeling with implicit energy-based models," *arXiv preprint arXiv:2510.02300*, 2025. arXiv: 2510.02300 [cs.LG].

[209] J. Ho and T. Salimans, *Classifier-free diffusion guidance*, 2022. arXiv: 2207.12598 [cs.LG].

[210] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[211] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, no. 4, pp. 339–356, 1988.

[212] R. Gao, Y. Song, B. Poole, Y. N. Wu, and D. P. Kingma, "Learning energy-based models by diffusion recovery likelihood," *arXiv preprint arXiv:2012.08125*, 2020.

[213] A. Rodriguez, B. Hoover, Y. Guo, and Y. Du, "Generative associative memory via equilibrium matching," in *ICLR 2026 Workshop on New Frontiers in Associative Memory*, NFAM 2026 Oral, 2026.

[214] R. Høier, T. Kerjan, and B. Scellier, "Training a convergent energy transformer with equilibrium propagation," in *ICLR 2026 Workshop on New Frontiers in Associative Memory*, NFAM 2026 Oral, 2026.

[215] B. Pham, B. Hoover, D. Krotov, and P. Ram, "Energy minimization for training dense associative memory," in *ICLR 2026 Workshop on New Frontiers in Associative Memory*, NFAM 2026 Poster, 2026.

[216] A. Helbling, P. Ram, D. H. Chau, and B. Hoover, "Energymap: Unraveling the data manifold with energy-based dimensionality reduction," in *ICLR 2026 Workshop on New Frontiers in Associative Memory*, NFAM 2026 Poster, 2026.

[217] L. Béthune, D. Vigouroux, Y. Du, R. VanRullen, T. Serre, and V. Boutin, "Follow the energy, find the path: Riemannian metrics from energy-based models," *arXiv preprint arXiv:2505.18230*, 2025. arXiv: 2505.18230 [cs.LG].

[218] S. Azeglio and A. Di Bernardo, "What's inside your diffusion model? a score-based riemannian metric to explore the data manifold," *arXiv preprint arXiv:2505.11128*, 2025. arXiv: 2505.11128 [cs.LG].

[219] Y. Du, J. Mao, and J. B. Tenenbaum, "Learning iterative reasoning through energy diffusion," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235, PMLR, Jul. 2024, pp. 11 764–11 776. arXiv: 2406.11179 [cs.LG].

[220] B. Huang, Z. Geng, and Z. Kolter, "Equilibrium reasoners: Learning attractors enables scalable reasoning," *arXiv preprint arXiv:2605.21488*, 2026. arXiv: 2605.21488 [cs.LG].

[221] J. Fein-Ashley and P. Rashidinejad, "Solve the loop: Attractor models for language and reasoning," *arXiv preprint arXiv:2605.12466*, 2026. arXiv: 2605.12466 [cs.LG].

[222] N. Mohseni, P. L. McMahon, and T. Byrnes, "Ising machines as hardware solvers of combinatorial optimization problems," *Nature Reviews Physics*, vol. 4, no. 6, pp. 363–379, 2022.

[223] M. G. Bacvanski, X. You, J. Hopfield, and D. Krotov, "Dense associative memories with analog circuits," *arXiv preprint arXiv:2512.15002*, 2025. arXiv: 2512.15002 [cs.ET].

[224] J. J. Hopfield. "John hopfield: Collective phenomena & physics-inspired computation," Associative Memory & Hopfield Networks NeurIPS'23, Accessed: Jun. 23, 2026.

[225] R. Wightman, *Pytorch image models*, https://github.com/rwightman/pytorch-image-models, 2019.

[226] J. Bradbury et al., *JAX: Composable transformations of Python+NumPy programs*, version 0.3.13, 2018.

[227] J. Heek et al., *Flax: A neural network library and ecosystem for JAX*, version 0.6.0, 2020.

[228] T. Zhao, C. Deng, K. Yu, T. Jiang, D. Wang, and M. Jiang, "Error-bounded graph anomaly loss for gnns," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1873–1882.

[229] K. Ding, J. Li, N. Agarwal, and H. Liu, "Inductive anomaly detection on attributed networks," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 1288–1294.

[230] Y. Liang, D. Krotov, and M. J. Zaki, "Modern hopfield networks for graph embedding," *Frontiers in big Data*, vol. 5, p. 1 044 709, 2022.

[231] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[232] C. Ying et al., "Do transformers really perform bad for graph representation?" *CoRR*, vol. abs/2106.05234, 2021. arXiv: 2106.05234.

[233] I. Babuschkin et al., *The DeepMind JAX Ecosystem*, 2020.

[234] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[235] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," *CoRR*, vol. abs/1711.05101, 2017. arXiv: 1711.05101.

[236] H. Yong, J. Huang, X. Hua, and L. Zhang, "Gradient-centralization: A new optimization technique for deep neural networks," 2020.

[237] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" *Advances in neural information processing systems*, vol. 32, 2019.

[238] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[239] A. Horé and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 2366–2369.

[240] Q. Dong et al., "A survey on in-context learning," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 1107–1128.

[241] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[242] T. Barrus, *Pyspellchecker: Pure python spell checking*, 2020.

[243] R. Flesch, "A new readability yardstick," *Journal of Applied Psychology*, vol. 32, no. 3, p. 221, 1948.

[244] T. Brown et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[245] P. W. Frey and D. J. Slate, "Letter recognition using holland-style adaptive classifiers," *Machine learning*, vol. 6, pp. 161–182, 1991.

[246] Y. Li, M. E. Ildiz, D. Papailiopoulos, and S. Oymak, "Transformers as algorithms: Generalization and stability in in-context learning," in *Proceedings of the 40th International Conference on Machine Learning*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., ser. Proceedings of Machine Learning Research, vol. 202, PMLR, Jul. 2023, pp. 19 565–19 594.

[247] I. Higgins et al., "Beta-VAE: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations*, 2017.