TRANSFORMERS AS IN-CONTEXT SAMPLERS: FROM CLOSED-FORM DIFFUSION TO ESTIMATION-FREE SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

A growing body of work establishes that large language models are not mere statistical memorizers, but are capable of in-context learning: performing inference at test time using only examples provided in the prompt, without any parameter updates. Prior theoretical work has shown that this capability extends to supervised learning tasks such as linear regression. We prove that in-context learning extends further to *data generation*: frozen transformers can simulate iterative generative samplers from in-context samples. We first show that transformers can realize closed-form and smoothed closed-form diffusion samplers. The construction identifies a concrete generative role for softmax attention: it computes responsibility weights and weighted empirical averages, while feedforward layers implement Euler updates.

To empirically relate these constructions to pretrained language models, we study *semantic-topic sampling*: prompts consisting of words drawn from a common semantic category, such as animals, foods, or cities. Across transformer layers, the normalized hidden states exhibit a two-stage geometry: they move toward a uniform spherical reference in intermediate layers and then return to structured, topic-dependent representations near the output. We further measure an interacting-particle energy on these hidden-state clouds and observe the same U-shaped pattern. We then prove that transformers can approximate an energy-based sampler, constructing the same U-shaped energy across the layers.

1 INTRODUCTION

"In-context learning" (Brown et al., 2020) has changed how we think about transformers. A transformer can adapt its behavior at inference time using only information provided in a prompt, without updating its parameters. This phenomenon was first made prominent by large language models, where a prompt containing a few demonstrations can induce the model to perform a new task (Brown et al., 2020; Chowdhery et al., 2023; Achiam et al., 2023; Touvron et al., 2023a;b). It has since become a central paradigm for studying how models use demonstrations, instructions, and contextual information at inference time (Dong et al., 2024).

Existing studies have mostly focused on conditional data generation, specifically for supervised learning: the prompt contains examples from an unknown task, and the model predicts the output for a new query. A standard formalization studies prompts of the form

$$\underbrace{(x_1, f(x_1), \ldots, x_k, f(x_k), x_{\text{query}})}_{\text{prompt}} \quad \longmapsto \quad \underbrace{f(x_{\text{query}})}_{\text{completion}}.$$

For example, Garg et al. (Garg et al., 2022) ask whether transformers can learn simple function classes in context, such as linear functions, sparse linear functions, decision trees, and shallow neural networks. Subsequent work studies whether transformers implement particular learning algorithms in context, including ridge regression, gradient descent, higher-order optimization, and algorithm selection (Akyürek et al., 2023; Li et al., 2023; Bai et al., 2023b; Fu et al., 2023). These results suggest that transformers can do more than match surface patterns in the prompt: they can execute nontrivial computations inside their forward pass.

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

Transformers were originally proposed for conditional text generation (Vaswani et al., 2017), but have since become the backbone of powerful unconditional generative models, including latent diffusion models (Rombach et al., 2022). While recent studies have extensively investigated the mechanistic interpretation of in-context learning in conditional text generation, it remains an open question whether such capabilities extend to unconditional data generation and broader generative modeling applications. In this paper, we ask whether in-context learning extends to data generation:

Can a transformer implement a sampler in context?

Given i.i.d. samples $x_1, \ldots, x_n \in \mathbb{R}^d$ from an unknown distribution $\mathcal{P}$, a sampling algorithm aims to generate a fresh sample $x_{n+1} \neq x_1, \ldots, x_n$ from $\mathcal{P}$. Akin to the standard settings of generative models, we assume that the density function of the data distribution is unknown, and we only have access to i.i.d. samples from the target distribution. Inspired by prior in-context learning studies (Ahn et al., 2023; Garg et al., 2022), we encode the sampling problem as

$$\text{input:} \quad x_1, \ldots, x_n \sim_{\text{i.i.d.}} \mathcal{P} \quad \longmapsto \quad \text{output:} \quad x_{n+1} \sim \mathcal{P},$$

where $x_1, \ldots, x_n$ are encoded in the word embeddings of a large language model and x_{n+1} is an output embedding vector. We refer to this prompting strategy as *in-context sampling*: the transformer parameters remain fixed while the distribution of $\mathcal{P}$ changes at test-time. A model can leverage two sources of information for inference: statistics from training data and information from in-context samples. In-context learning emerges when the model relies on the latter, adapting its response to the in-context input at test time.

Figure 1 illustrates this capability. A transformer trained on samples from a face distribution containing the boundary and eyes, but no smiling mouth, generates samples matching the prompt distribution, including an unseen smile-shaped curve. Thus, inference uses in-context samples rather than merely reproducing the training distribution or selecting memorized components. Our goal is to explain the mechanism underlying this capability.

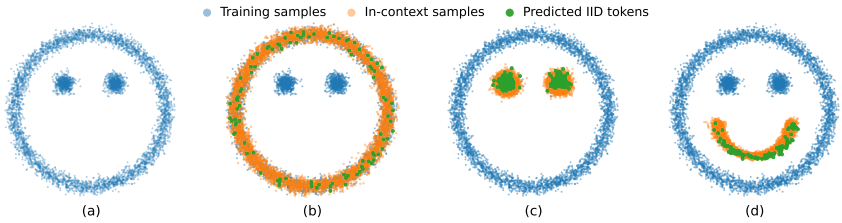


Figure 1: **In-context sampling.** A transformer is trained on the blue face distribution in (a), containing the boundary and eyes but no smiling mouth. Orange context points guide green generated samples along the boundary (b), eyes (c), or an unseen smiling mouth (d). Generalization to the mouth indicates that inference uses in-context samples rather than only training statistics. See Section 6 for details.

CONTRIBUTIONS

We make the following contributions to demonstrate the generative power of transformers.

In-context simulation of diffusion models. We prove that transformers can simulate closed-form diffusion models (Scarvelis et al., 2025) from in-context samples. Closed-form diffusion models leverage the explicit score function to simulate the backward diffusion process. We show that softmax attention heads are particularly well-suited to compute the empirical score function directly from in-context samples.

Mechanistic analysis of pretrained LLMs. We examine the data generation mechanism in pretrained language models, including GPT-2 (Radford et al., 2019), Llama-3.3-70B-Instruct (Grattafiori et al., 2024), OpenLM-Llama-13B/OpenLLaMA (Geng & Liu, 2023), Cerebras-GPT-13B (Dey et al., 2023), Qwen2.5 (Qwen et al., 2024), Falcon-40B (Almazrouei et al., 2023), OPT-66B (Zhang et al., 2022), and BLOOM-7B1 (Workshop et al., 2022). The models are prompted by words drawn from the same semantic topic. This prompt construction mirrors the i.i.d. assumption in the in-context sampling setting. Tracking the empirical embedding distributions across layers, we observe that the distribution of normalized embeddings progressively approaches a uniform distribution over a

sphere in the middle layers, before becoming non-uniform again towards the output layers. The same U-shaped pattern is observed for the interacting energy of word embedding.

Estimation-free sampling. We identify a sampler whose layer-wise energy profile qualitatively matches the observed U-shaped. We prove that transformers can implement estimation-free sampling (EFS), an energy-based generative framework comprising two steps: (1) transporting the empirical data distribution toward a uniform distribution, and (2) inverting this process to push the empirical distribution toward a target distribution. This two-step mechanism resembles the internal distribution shift observed in pretrained transformers.

Connections to prior work. Our work follows the algorithmic view of in-context learning (Garg et al., 2022; Akyürek et al., 2023; Ahn et al., 2023; Li et al., 2023; Bai et al., 2023b) and connects it to diffusion (Ho et al., 2020; Song et al., 2020b;a) and EFS (Daneshmand & Soleymani, 2025). Prompt Diffusion (Wang et al., 2023b) trains a diffusion model for visual tasks from in-context examples; we ask whether a frozen transformer can implement sampling algorithms from the prompt. Rosu et al. (Rosu et al., 2026) establish exact denoising constructions using RBF-based attention, equivalent to standard attention under specific equal-norm conditions. Our construction uses standard softmax attention with norm-dependent quantities supplied in the input encoding. This diffusion result is a warm-up: expressivity does not identify the algorithm learned by a trained transformer. We therefore study EFS, whose forward–backward dynamics qualitatively resemble the observed U-shaped evolution of pretrained representations, without establishing that pretrained models implement EFS. Unlike unsmoothed empirical closed-form diffusion, EFS admits guarantees for transport to the underlying distribution under corresponding mean-field assumptions (Daneshmand & Soleymani, 2025). Appendix A provides further discussion.

2 PRELIMINARIES

2.1 CLOSED-FORM DIFFUSION PROCESS

Score-based diffusion models (Song et al., 2020b) are among the most powerful generative models for transporting a simple reference distribution, such as a Gaussian distribution, to a target data distribution. In these models, the transport dynamics are driven by the score function, i.e., the gradient of the log-density of a smoothed data distribution. In practice, this score is typically approximated by a neural network trained through denoising.

In continuous time, the reverse-time generative dynamics can be described through Anderson's reverse-time diffusion framework (Anderson, 1982). In an abstract form, the reverse time process is written as

$$dz_t = (-z_t - \nabla_z \log p_t(z_t))\, dt + dW_t, \tag{1}$$

where p_t denotes the density of the target distribution after Gaussian smoothing at time t, and W_t is a Wiener process. The term $\nabla_z \log p_t(z_t)$ drives the sample toward regions of high probability under the smoothed data distribution. [1] The Gaussian smoothing and stochastic noise injection play an annealing role: at large noise levels, the density is smoother and easier to explore, while at smaller noise levels, the dynamics refine samples toward the data distribution.

Closed-form diffusion models use the observation that, for a finite empirical distribution, the Gaussian-smoothed density can be written explicitly as a Gaussian mixture (Scarvelis et al., 2025). Let $\{x_i\}_{i=1}^N \subset \mathbb{R}^d$ be the empirical dataset. For $t \in (0, 1)$, define

$$\rho_t^\star(z) = \frac{1}{N} \sum_{i=1}^{N} \phi\big(z; tx_i, (1-t)^2 I_d\big)\,, \qquad z \in \mathbb{R}^d,$$

where $\phi(\cdot; \mu, \Sigma)$ is the Gaussian density with mean μ and covariance Σ. Thus, $\rho_t^\star$ is a mixture of isotropic Gaussians centered at the scaled samples tx_i, with covariance $(1-t)^2 I_d$.

For a current point z, define the responsibility weight of sample x_i by

$$w_i(t, z) = \frac{\phi\big(z; tx_i, (1-t)^2 I_d\big)}{\sum_{j=1}^{N} \phi(z; tx_j, (1-t)^2 I_d)}, \qquad i = 1, \ldots, N. \tag{2}$$

[1] The negative sign of $\nabla \log p_t(z_t)$ arises because the process runs in reverse time, from T back to 0.

These weights satisfy $w_i(t,z) \geq 0$ and $\sum_{i=1}^N w_i(t,z) = 1$. They measure how much the Gaussian component centered at tx_i contributes to the density at z. Define the corresponding weighted mean $k_t(z) = \sum_{i=1}^N w_i(t,z)(tx_i)$. Since the distribution of $p_t^*(z)$ is a Gaussian mixture with explicit density function, its score can be computed in closed-form as (Scarvelis et al., 2025)

$$\nabla_z \log \rho_t^\star(z) = \frac{1}{(1-t)^2}\big(k_t(z) - z\big). \tag{3}$$

Thus, the score tells the sampler how to move z: it points from the current point z toward the weighted average $k_t(z)$ of the data samples.

The sampler converts this score into an update direction, which we write it as

$$v_t(z) = \frac{1}{t}\left(z + (1-t)\nabla_z \log \rho_t^\star(z)\right). \tag{4}$$

Here $v_t(z) \in \mathbb{R}^d$ is the direction in which the sampler moves the current point z at time t. Substituting (3) gives

$$v_t(z) = \frac{1}{t}\left(z + \frac{1}{1-t}(k_t(z) - z)\right) = -\frac{1}{1-t}z + \frac{1}{t(1-t)}k_t(z).$$

Therefore, the main data-dependent computation is the weighted mean $k_t(z)$. Once $k_t(z)$ is known, the update direction $v_t(z)$ is a fixed linear combination of z and $k_t(z)$.

Given a time grid $\tau = \{(t_s, h_s)\}_{s=0}^{S-1}$, where $0 < t_s < 1$ and $h_s > 0$, the closed-form diffusion sampler relies on the following recurrence for sampling:

$$z_{s+1} = z_s + h_s v_{t_s}(z_s), \qquad s = 0, \ldots, S-1. \tag{5}$$

This explicit form is central to our construction. The sampler does not require a trained score network: the score and update direction are computed directly from the in-context samples $x_1, \ldots, x_N$. In later sections, we show that a frozen transformer can implement this computation using attention to compute the weights $w_i(t,z)$ and the weighted mean $k_t(z)$, and using feedforward layers to perform the Euler update.

Smoothing. Since the closed-form diffusion model may memorize data, (Scarvelis et al., 2025) proposes smoothing to avoid data memorization. Smoothing is obtained by averaging the closed-form mean over fixed perturbations of the current state. Let $\sigma \geq 0$ and $\mathcal{E} = \{\epsilon_m\}_{m=1}^M \subset \mathbb{R}^d$ be fixed perturbation vectors. Define $k_{\sigma,t}(z) = M^{-1}\sum_{m=1}^M k_t(z + \sigma\epsilon_m)$, where $k_t(\cdot)$ is the closed-form responsibility-weighted mean defined above. The smoothed velocity field is

$$v_{\sigma,t}(z) = -\frac{1}{1-t}z + \frac{1}{t(1-t)}k_{\sigma,t}(z). \tag{6}$$

Thus, the smoothed sampler replaces $k_t(z)$ by the averaged mean $k_{\sigma,t}(z)$. Setting $\sigma = 0$ recovers the closed-form diffusion method. (Scarvelis et al., 2025) has examined the data generation capability of the above method in comparison to original denoising diffusion models.

This modification preserves the same transformer-realization structure: attention computes $k_t(z + \sigma\epsilon_m)$ for each fixed perturbation, the heads are averaged to obtain $k_{\sigma,t}(z)$, and the feedforward layer applies the affine update (6).

2.2 Embedding encoding

We encode a closed-form diffusion instance in the input word embedding matrix of a transformer, denoted by

$$X = \mathrm{Enc}(\{x_i\}_{i=1}^N, z_0) \in \mathbb{R}^{N \times p}, \qquad p = 3d + 2,$$

where $z_0 \in \mathbb{R}^d$ is a Gaussian random vector which is equivalent to starting state of the closed-form diffusion model defined in (6). The encoded embedding consists of N data tokens; the final token also stores the sampler state. We write $X \in \mathbb{R}^{N \times p}$, with $p = 3d + 2$, and define its rows by

$$X_i = \big[x_i^\top,\ \|x_i\|^2,\ 0_d^\top,\ 0_d^\top,\ 1\big], \qquad i = 1, \ldots, N-1, \quad X_N = \big[x_N^\top,\ \|x_N\|^2,\ z_0^\top,\ 0_d^\top,\ 1\big].$$

Each row stores one in-context sample, and the final row also stores the initial sampler state z_0. Equivalently, each row is partitioned as $X_i = \left[X_i^{(x)} \mid X_i^{(r)} \mid X_i^{(z)} \mid X_i^{(k)} \mid X_i^{(1)}\right]$, where $X_i^{(x)}, X_i^{(z)}, X_i^{(k)} \in \mathbb{R}^d$, $X_i^{(r)} \in \mathbb{R}$, and $X_i^{(1)} \in \mathbb{R}$. These four blocks serve distinct purposes in the design: the x and r blocks encode training samples for in-context learning; the z block acts as memory for newly generated samples; and the k block functions as a scratchpad for intermediate computation, inspired by prior work (Daneshmand, 2026b). The state readout map $\Pi_{\text{state}} : \mathbb{R}^{m\times p} \to \mathbb{R}^d$ extracts the z-block of the final row: $\Pi_{\text{state}}(X) = X_N^{(z)} = X_{N,\, d+2:\, 2d+1}$. We will later use the above notation to reconstruct the output of a transformer.

2.3 Transformer

We use residual softmax self-attention and positionwise feedforward blocks (Vaswani et al., 2017), with no layer normalization or dropout in the theoretical constructions. Each block has the form $Y = X + \mathrm{Attn}(X)$, $X^+ = Y + \mathrm{FF}(Y)$, where FF is a shared, two-layer ReLU network with a freely chosen finite hidden width. Attention is full self-attention; the closed-form diffusion construction also permits standard causal attention. The usual attention scaling is absorbed into the query weights. Let $\mathcal{T}_\theta : \mathbb{R}^{m\times p} \to \mathbb{R}^{m\times p}$ denote a depth-L transformer, where m is the number of tokens, p is the token dimension, $L \in \mathbb{N}$ is the number of layers, and θ denotes the collection of attention and feedforward weight matrices. Thus, the transformer maps an input embedding matrix to an output embedding matrix of the same size.

We generate a sample by applying the state readout map $\Pi_{\text{state}} : \mathbb{R}^{m\times p} \to \mathbb{R}^d$, defined in the previous section, to the final embedding matrix. For a sampling instance $\{x_i\}_{i=1}^N$ with initial state z_0, the transformer output is $\widehat{z}_L = \Pi_{\text{state}}\big(\mathcal{T}_\theta\big(\mathrm{Enc}(\{x_i\}_{i=1}^N, z_0)\big)\big)$. Our goal is to construct parameter choices θ such that $\widehat{z}_L$ matches the output of a specified iterative generative sampler.

3 Simulation of Closed-form Diffusion with Transformers

We now show that the transformer architecture in Section 2.3 can simulate the (smoothed) closed-form diffusion sampler by a suitable choice of its parameters. The prompt contains the empirical samples $\{x_i\}_{i=1}^N \subset \mathbb{R}^d$ and the initial state $z_0 \in \mathbb{R}^d$.

Theorem 1 (Transformer simulation of closed-form diffusion)**.** *There exists a choice of transformer parameters $\theta^\star_{\mathrm{CFD}}$ such that, for every empirical dataset $\{x_i\}_{i=1}^N \subset \mathbb{R}^d$ and every initial state $z_0 \in \mathbb{R}^d$, if z_L is generated by the closed-form diffusion recursion, then*

$$\Pi_{\text{state}}\left(\mathcal{T}_{\theta^\star_{\mathrm{CFD}}}\big(\mathrm{Enc}(\{x_i\}_{i=1}^N, z_0)\big)\right) = z_L,$$

where L is the number of attention blocks. Moreover, fix $\sigma \geq 0$ and $\mathcal{E} = \{\epsilon_m\}_{m=1}^M$. There exists another choice of transformer parameters $\theta^\star_{\sigma\text{-CFD}}$, using the same prompt encoding and state readout, such that, for every empirical dataset and initial state, if z_L is generated by the smoothed recursion, then

$$\Pi_{\text{state}}\left(\mathcal{T}_{\theta^\star_{\sigma\text{-CFD}}}\big(\mathrm{Enc}(\{x_i\}_{i=1}^N, z_0)\big)\right) = z_L.$$

The parameter choices may depend on N, d, τ, and, in the smoothed case, on σ and $\mathcal{E}$, but they do not depend on the realized dataset $\{x_i\}_{i=1}^N$ or on z_0.

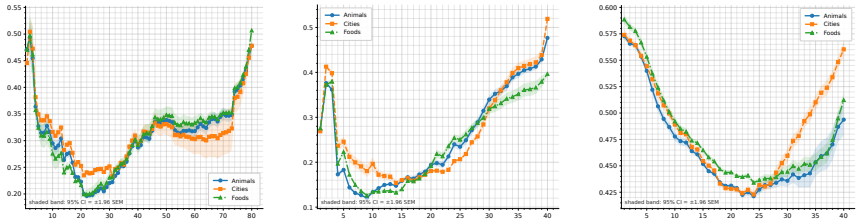
The same frozen parameters implement the prescribed sampler for every dataset and initial state, at fixed context length, dimension, and schedule. This establishes uniform in-context simulation over the data values. Figure 1 illustrates empirical out-of-distribution generation from a prompt distribution absent from training.

The proof uses the computational capabilities of softmax attention (Vaswani et al., 2017). Prior constructions implement gradient descent for regression (von Oswald et al., 2023; Ahn et al., 2023) and temporal-difference learning (Wang et al., 2025) using linear attention. Here softmax normalization is essential: it produces precisely the responsibility weights required by closed-form diffusion, while feedforward layers implement the updates.

4 MECHANISTIC ANALYSIS OF PRETRAINED TRANSFORMERS

We established the expressivity of transformers to implement closed-form diffusion models. In this section, we investigate the internal mechanism of sampling in pretrained transformers. Our goal is not to claim that pretrained language models exactly implement a sampling algorithm. Instead, we study how they shape word embedding distribution for sampling.

In-context sampling can be formulated as generating new words from a semantic topic, such as animals, cities, or foods, given random words from the topic. We construct prompts by sampling distinct words without replacement from a common semantic category, as detailed in Appendix E.3. The words within a prompt are therefore dependent, and contextualization introduces further dependence among their hidden states. Then, we analyze how the distribution of word embeddings changes across the layers. Specifically, we normalize the embeddings and measure their distance to the uniform distribution on the sphere. We use squared maximum mean discrepancy, MMD^2, as the distance metric; MMD is a standard kernel-based discrepancy between probability distributions (Gretton et al., 2012). We report the off-diagonal Gaussian-kernel MMD^2 statistic as a descriptive finite-cloud comparison. Because contextual hidden states are dependent, we do not claim that it is an unbiased estimator of a population discrepancy. Lower values indicate a smaller measured discrepancy to the chosen reference; they do not establish convergence to a uniform distribution.



Llama-3.3-70B-Instruct (Grattafiori et al., 2024) | OpenLM-Llama-13B (Geng & Liu, 2023) | Cerebras-GPT-13B (Dey et al., 2023)

Figure 2: **Mechanism of sampling in pretrained language models.** The y-axis shows the layer-wise MMD^2 distance between normalized token embeddings and the uniform distribution on the sphere, and the x-axis shows the transformer layer index. The models are prompted with distinct words sampled without replacement from the categories animals (blue), cities (orange), and foods (green). Several models exhibit a U-shaped profile: intermediate layers move closer to the uniform reference distribution.

We observe a two-stage pattern for three pretrained foundation models: the measured discrepancy to the uniform spherical reference decreases in intermediate layers and increases near the output. This observation holds for two variants of LLaMA (Grattafiori et al., 2024; Geng & Liu, 2023) and Cerebras-GPT (Dey et al., 2023) in Figure 2. To illustrate this dynamic, we present a scatter plot of word embeddings for a small trained model across the layers in Figure 3, where each point represents one word. The middle-layer clouds visibly spread out; this visualization does not establish convergence to a uniform distribution.

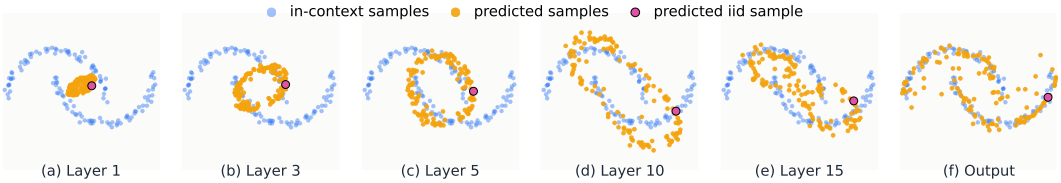
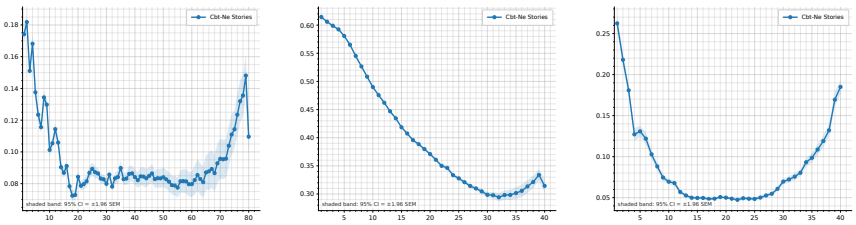


Figure 3: **Uniform bias for in-context sampling.** A small GPT-2-style model is trained to reproduce samples from the two-moons dataset (see Appendix E for details). Orange points represent intermediate embeddings extracted at layers 1, 3, 5, 10, 15, and output layers, illustrating the progressive data evolution across transformer layers at test time.

We observe a similar pattern when pretrained language models are prompted with natural sentences rather than distinct semantic-category words. Specifically, we prompt a pretrained model with natural sentences from the CBT dataset (Hill et al., 2015). Figure 4 plots the MMD distance between word

embeddings and samples drawn uniformly from the unit sphere. We use only one word embedding for repeated words when computing the MMD distance.



Llama-3.3-70B-Instruct (Grattafiori et al., 2024) | Cerebras-GPT-13B (Dey et al., 2023) | OpenLM-Llama-13B (Geng & Liu, 2023)

Figure 4: **Uniform intermediate representations in pretrained language models.** The y-axis shows the layerwise MMD^2 between normalized token embeddings and the uniform distribution on the unit sphere, and the x-axis shows the transformer layer index. The models are prompted with sentences from the CBT dataset, with each repeated word represented by a single embedding vector.

5 Physics of In-context Sampling

5.1 Observation: Embeddings as Interacting Particles

While proving the sampling capability, Theorem 1 cannot explain the U-shaped mechanism of sampling in pretrained models discussed in the last section. This is due to the fact that closed-form diffusion does not change the empirical data distribution. We use tools from interacting particles to provide insights into the shaping of the embedding distribution for sampling. Consider the following interacting energy defined over word embeddings

$$E(x_1, \dots, x_N) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} W_\epsilon^{(s)}(x_i - x_j). \tag{7}$$

where $N \geq 2$, $x_1, \dots, x_N \in \mathbb{R}^d$, $\epsilon > 0$, and $s > -2$. We also write $E_{N,\epsilon} = E$, with s fixed. For $s \neq 0$, the interaction potential is

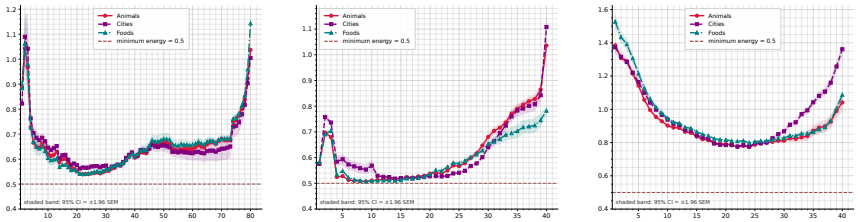
$$W_\epsilon^{(s)}(r) = \frac{1}{2}\|r\|^2 + \frac{1}{s(\|r\|^2 + \epsilon)^{s/2}}, \qquad r \in \mathbb{R}^d.$$

The logarithmic case requires subtracting an additive constant before taking the limit:

$$W_\epsilon^{(0)}(r) = \lim_{s \to 0} \left(W_\epsilon^{(s)}(r) - \frac{1}{s} \right) = \frac{1}{2}\|r\|^2 - \frac{1}{2}\log(\|r\|^2 + \epsilon).$$

This renormalization leaves the interaction force unchanged. Results for continuum minimizers depend on the dimension and potential: for the unregularized logarithmic interaction with quadratic attraction, a uniform measure on a suitable sphere is a minimizer in dimensions $d \geq 4$ (Frank & Matzke, 2025). Such results do not imply convergence of finite-particle gradient descent with $\epsilon > 0$ to a uniform distribution. We use the regularized energy as a diagnostic of the embedding geometry.

We compute the energy of word embeddings where $x_1, \dots, x_N$ are word embeddings across layers of pretrained transformers. When the input prompt consists of distinct words from the same topic, Figure 5 shows that the energy function produces a U-shaped plot across the layers, similar to the MMD distance plot in Figure 2. Energy optimization also motivates EFS (Daneshmand & Soleymani, 2025). We show that a transformer can approximate a finite explicit discretization of the corresponding forward–backward computation, providing a computational analogy for these empirical observations.



Llama-3.3-70B-Instruct (Grattafiori et al., 2024) | OpenLM-Llama-13B (Geng & Liu, 2023) | Cerebras-GPT-13B (Dey et al., 2023)

Figure 5: **Repulsive-interactive energy of word embeddings.** The y-axis shows the energy E of word embeddings across the layers of a pretrained transformer, and the x-axis shows the transformer layer index. We use the renormalized logarithmic potential $W_\epsilon^{(0)}$ and normalize the word embeddings to norm 0.78. The red dashed line at 0.5 is a reference level corresponding to the minimum of the unregularized logarithmic pair potential; it is not, in general, the minimum of the regularized finite-particle energy.

5.2 Theory: Energy-based sampling with particle gradient descent

Estimation-Free Sampling (EFS) (Daneshmand & Soleymani, 2025) combines forward particle optimization with backward transport. We analyze the following finite, explicit forward–backward discretization inspired by EFS. The backward step below is reverse-time Euler, not the exact inverse of a gradient-descent step; the original EFS algorithm instead computes inverse steps through an inner optimization. For fixed $N \geq 2$, d, $K \geq 1$, $\gamma > 0$, $s > -2$, and $\epsilon > 0$, the forward step applies gradient descent:

$$x_i^{(j+1)} = x_i^{(j)} - \gamma \nabla_{x_i} E_{N,\epsilon}\big(x_1^{(j)}, \ldots, x_N^{(j)}\big), \qquad j = 0, \ldots, K-1.$$

Energy decreases under the usual step-size condition $0 < \gamma < 2/L_E$, where L_E bounds the Lipschitz constant of the energy gradient on a domain containing the update segments. Neither this condition nor compactness alone guarantees a uniform terminal cloud; for example, coincident particles are stationary for $\epsilon > 0$. Given a reference point $z^{(K)}$, define the tagged-point potential

$$V_{N,\epsilon}(z; x_1, \ldots, x_N) = c_N \sum_{i=1}^{N} W_\epsilon^{(s)}(z - x_i), \qquad c_N = \frac{2}{N(N-1)}.$$

This normalization gives $\nabla_z V_{N,\epsilon}(x_i; x_1, \ldots, x_N) = \nabla_{x_i} E_{N,\epsilon}(x_1, \ldots, x_N)$. The backward phase evolves the reference point through the explicit recurrence, for $j = K, \ldots, 1$,

$$z^{(j-1)} = z^{(j)} + \gamma \nabla_z V_{N,\epsilon}\big(z^{(j)}; x_1^{(j)}, \ldots, x_N^{(j)}\big).$$

The final generated sample is $z^{(0)}$. We collect the fixed EFS parameters in $\tau = (K, \gamma, s, \epsilon)$, together with any fixed schedule choices.

This two-stage computation motivates an analogy with the observed layerwise geometry. The next theorem establishes uniform approximation of the finite recurrence, without asserting exact inversion, recovery of the population distribution, or a U-shaped energy trajectory.

Theorem 2 (Transformer simulation of an explicit EFS discretization)**.** *Fix the parameters above and a compact set $\mathcal{K} \subset \mathbb{R}^d$ containing all particle and reference iterates, including the initial values, for every admissible input. Then, for every $\varepsilon > 0$, there exist a finite transformer depth, finite feedforward hidden widths, and parameters $\theta_{\mathrm{EFS}}^\star$ such that, for every admissible input $x_1^{(0)}, \ldots, x_N^{(0)}, z^{(K)}$, if $z^{(0)}$ is generated by the explicit forward–backward recursion above, then*

$$\left\| \Pi_{\mathrm{state}} \left(\mathcal{T}_{\theta_{\mathrm{EFS}}^\star} \big(\mathrm{Enc}_{\mathrm{EFS}}(x_1^{(0)}, \ldots, x_N^{(0)}, z^{(K)}) \big) \right) - z^{(0)} \right\| \leq \varepsilon.$$

The depth, feedforward widths, and parameter choice $\theta_{\mathrm{EFS}}^\star$ may depend on $N, d, K, \tau, \varepsilon$, and the compact domain $\mathcal{K}$, but not on the realized particle cloud $x_1^{(0)}, \ldots, x_N^{(0)}$ or reference point $z^{(K)}$.

The theorem shows that a residual softmax transformer can approximate this finite particle-based computation in context using the encoding in Appendix C.1. It controls the terminal state readout. It does not prescribe the interaction energy of the full hidden-state vectors across layers, which also contain trajectory memory and scratch coordinates; the U-shaped curves in Figure 5 remain empirical observations.

6 EXPERIMENTS

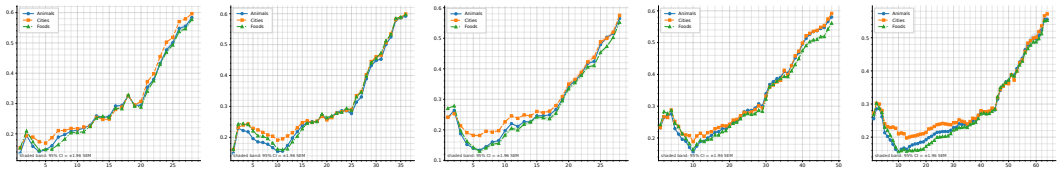
We examine whether transformers generate samples from prompt-specified distributions and whether their hidden states exhibit layerwise dynamics consistent with EFS.

Controlled sampling experiments. We train a small GPT-2-style decoder-only transformer (Radford et al., 2019) on two-dimensional i.i.d. point-cloud sequences using a causal next-token objective. Figure 1 tests generalization to a smile-shaped component absent from training. In a separate two-moons experiment, we project intermediate hidden states back to the data space to examine the evolution of sample geometry; Appendix E.2 provides the full visualization.

Pretrained language models. For prompts of i.i.d. semantic-category words, we normalize token embeddings and compute the unbiased RBF MMD^2 estimator relative to the uniform distribution on the sphere. Figures 2 and 4 report layerwise profiles for these prompts and natural text from CBT (Hill et al., 2015), respectively. We also evaluate the EFS-style interaction energy on the same hidden-state clouds (Figure 5). The corresponding U-shaped profiles are consistent with the proposed interpretation, but do not establish that pretrained models implement EFS. Additional results and experimental details appear in Appendix E.3.

7 DISCUSSION AND LIMITATIONS

Failure mode across model scales. The uniformization effect is not equally pronounced across all models. In the Qwen2.5 family (Qwen et al., 2024), smaller models exhibit only a short or weak movement toward the uniform reference, whereas larger variants show a clearer two-stage profile as shown in Figure 6. This provides a concrete failure mode for the proposed mechanism: when the model has insufficient effective depth or capacity, the intermediate transport phase may be incomplete. This observation is consistent with our theoretical construction, where transformer depth controls the number of iterative computation steps available to the model. Additional failure-mode experiments for Qwen2.5 and also GPT-2 families are provided in Appendix E.4.



(a) 1.5B (b) 3B (c) 7B (d) 14B (e) 32B

Figure 6: Qwen2.5 family (Qwen et al., 2024) on i.i.d. semantic-category prompts. Unlike the clearer U-shaped profiles observed in larger pretrained models, the Qwen2.5 family exhibits a weaker and less pronounced U-shaped pattern. The curves still suggest a middle-layer movement toward a uniform spherical reference distribution followed by a later movement away from it, but the effect is shorter and less robust across model scales.

Gap between expressivity and mechanism. Our theoretical results are expressivity statements: they show that some transformer parameters can implement closed-form diffusion and EFS from in-context samples, but they do not prove that pretrained language models execute these algorithms internally. Our experiments provide compatible mechanistic evidence, including middle-layer movement toward a uniform reference distribution and a decrease in EFS-style interaction energy. Still, these observations do not uniquely identify the internal algorithm. This gap is common in in-context learning: prior work shows that transformers can learn regression functions (Garg et al., 2022), implement gradient-descent-like computations (Ahn et al., 2023; von Oswald et al., 2023), or realize temporal-difference updates (Wang et al., 2025), but such results mainly characterize representational capacity or behavior in controlled settings. Similarly, our work identifies sampling algorithms that transformers can implement and shows compatible layerwise dynamics, while leaving a full characterization of the mechanism to future work.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36:45614–45650, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *International Conference on Learning Representations*, 2023.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023a.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Advances in Neural Information Processing Systems*, volume 36, 2023b.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4005–4019, 2023.

Hadi Daneshmand. In-context learning for discrete optimal transport: Can transformers sort? *AISTATS*, 2026a.

Hadi Daneshmand. In-context learning for discrete optimal transport: Can transformers sort? In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026b. URL https://openreview.net/forum?id=hwIs2DCYFU.

Hadi Daneshmand and Ashkan Soleymani. Data generation without function estimation. *NeurIPS workshop on optimization for machine learning*, 2025.

Nolan Dey, Gurpreet Gosal, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, Joel Hestness, et al. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. *arXiv preprint arXiv:2304.03208*, 2023.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 1107–1128, 2024.

Rupert L Frank and Ryan W Matzke. Minimizers for an aggregation model with attractive–repulsive interaction. *Archive for Rational Mechanics and Analysis*, 249(2):15, 2025.

Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn higher-order optimization methods for in-context learning: A study with linear models. *arXiv preprint arXiv:2310.17086*, 2023.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30583–30598. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c529dba08a146ea8d6cf715ae8930cbe-Paper-Conference.pdf.

Xinyang Geng and Hao Liu. Openllama: An open reproduction of llama, May 2023. URL https://github.com/openlm-research/open_llama.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773, 2012.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pp. 19565–19594, 2023.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pp. 100–114, 2022.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 8086–8098, 2022.

Arvind Mahankali, Tatsunori B. Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. *arXiv preprint arXiv:2307.03576*, 2023.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

A Yang Qwen, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengpeng Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint*, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Paul Rosu, Lawrence Carin, and Xiang Cheng. From softmax to score: Transformers can effectively implement in-context denoising steps. *Advances in Neural Information Processing Systems*, 38: 142994–143017, 2026.

Christopher Scarvelis, Haitz Sáez de Ocáriz Borde, and Justin Solomon. Closed-form diffusion models. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=JkMifr17wc.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

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

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174, 2023.

Jiuqi Wang, Ethan Blaser, Hadi Daneshmand, and Shangtong Zhang. Transformers can learn temporal difference methods for in-context reinforcement learning, 2025. URL https://arxiv.org/abs/2405.13861.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9840–9855, 2023a.

Zhendong Wang, Yifan Jiang, Yadong Lu, Yelong Shen, Pengcheng He, Weizhu Chen, Zhangyang Wang, and Mingyuan Zhou. In-context learning unlocked for diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, 2023b.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 1423–1436, 2023.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit Bayesian inference. In *International Conference on Learning Representations*, 2022.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. What and how does in-context learning learn? Bayesian model averaging, parameterization, and generalization. *arXiv preprint arXiv:2305.19420*, 2023.

TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

A RELATED WORK

In-context learning. In-context learning refers to the ability of a model to adapt its behavior at inference time using only information supplied in the prompt, without updating its parameters. This phenomenon was popularized by large language models (Brown et al., 2020; Chowdhery et al., 2023; Achiam et al., 2023; Touvron et al., 2023a;b; Grattafiori et al., 2024) and has since become a central topic in the study of transformer-based models (Dong et al., 2024). Much of the empirical literature studies how performance depends on prompt construction, including the choice, ordering, and formatting of demonstrations (Liu et al., 2022; Lu et al., 2022; Wu et al., 2023). Complementary work studies mechanisms behind in-context learning, including induction heads, attention-based information flow, Bayesian inference, and gradient-descent-like computation inside transformers (Olsson et al., 2022; Wang et al., 2023a; Xie et al., 2022; Zhang et al., 2023; Dai et al., 2023; von Oswald et al., 2023; Ahn et al., 2023; Mahankali et al., 2023).

Most theoretical studies of in-context learning focus on prediction. In the standard formulation, the prompt contains input–output examples from an unknown task, and the model predicts the output for a new query. Garg et al. (Garg et al., 2022) showed that transformers trained from scratch can learn simple function classes from examples in the prompt, including linear functions, sparse linear functions, decision trees, and shallow neural networks. Related work studies whether transformers implement particular learning algorithms in context, including ridge regression, gradient descent, higher-order optimization, and algorithm selection (Akyürek et al., 2023; Li et al., 2023; Bai et al., 2023b; Fu et al., 2023). Our work follows this algorithmic view, but changes the target from prediction to generation: instead of asking whether a transformer can predict a label or function value, we ask whether it can execute a sampler in context.

Transformers as algorithm executors. A growing line of work studies transformers as models capable of executing structured computations. Prior work has shown that transformer depth and attention can implement iterative optimization procedures, temporal-difference style updates, and discrete optimal transport or sorting-type computations (Ahn et al., 2023; Wang et al., 2025; Daneshmand, 2026a;b). These results suggest that attention and depth can act as computational resources: attention aggregates information from the prompt, while depth implements repeated updates.

Our results extend this perspective to generative algorithms. Closed-form diffusion and Estimation-Free Sampling are not prediction rules; they are iterative procedures that transform an initial state into a sample. Showing that frozen transformers can implement these procedures in context strengthens the interpretation of transformers as general-purpose algorithm executors. The main difference from prior algorithmic ICL work is that the output of the computation is not a predicted label or regression value, but the terminal state of a sampler.

Diffusion and score-based generative models. Diffusion models generate samples by gradually transforming noise into data through an iterative reverse process (Ho et al., 2020). Score-based generative modeling provides a continuous-time formulation in which a forward stochastic process maps data to noise and a reverse-time process generates data from noise using estimated score functions (Song et al., 2020b). Related samplers, including DDIM, modify the reverse process through deterministic or implicit updates (Song et al., 2020a). These methods have made iterative sampling a central computational paradigm in modern generative modeling.

Our work does not propose a new diffusion model or train a new score network. Instead, we isolate the computational structure of diffusion-style sampling. In the closed-form setting, the score and sampling dynamics are explicit functions of the empirical samples and the current state. This allows us to ask a representational question: can a frozen transformer implement the corresponding sampler when the empirical distribution and initial state are supplied in the prompt? Our answer is affirmative, showing that a core diffusion-style computation can be realized as in-context computation by a transformer.

In-context generation with diffusion models. There is prior work on enabling in-context learning inside diffusion models. Prompt Diffusion trains a diffusion-based model to perform vision tasks from

example input–output pairs and a query image, thereby bringing in-context learning to diffusion-based generation (Wang et al., 2023b). The paper is the closest prior work connecting diffusion models and in-context learning, but its direction is different: it equips a diffusion model with in-context learning ability.

Our question reverses this direction. Rather than asking whether a diffusion model can perform in-context learning, we ask whether a transformer can implement diffusion-style and particle-based samplers in context. Thus, the central object in our paper is not a diffusion architecture conditioned on examples, but a frozen transformer that executes a sampling algorithm specified by the prompt.

Estimation-Free Sampling. Estimation-Free Sampling (EFS) proposes a generative mechanism that avoids explicit score estimation and neural generative training (Daneshmand & Soleymani, 2025). Instead of learning a score or density-dependent function, EFS uses deterministic particle optimization. Its forward step transports empirical particles toward a simple reference distribution, and its backward step maps a newly sampled reference point back toward the data distribution. The EFS paper explicitly frames this as a generation without score/function estimation, neural-network training, or noise injection in the mean-field regime.

We include EFS because it tests whether in-context sampling is limited to diffusion-style score dynamics. Our result shows that it is not. A frozen transformer can also implement a particle-optimization sampler in context. This broadens the scope of the theory: transformers can realize not only closed-form diffusion updates, but also deterministic particle-based generative algorithms.

Positioning. Taken together, prior work shows that transformers can learn from context, diffusion models can generate through iterative sampling, and particle methods can generate without explicit function estimation. Our contribution connects these threads. We introduce in-context sampling as a framework for studying generative algorithms executed by frozen transformers. In this framework, the prompt specifies the sampling instance, attention computes interactions among samples or particles, and transformer depth performs the iterative updates. This perspective broadens in-context learning from prediction to generation and places closed-form diffusion and Estimation-Free Sampling under a common computational view.

B Proof of Theorem 1

We use the prompt encoding from Section 2.2. Each token has a block form

$$X_i = \big[X_i^{(x)},\, X_i^{(r)},\, X_i^{(z)},\, X_i^{(k)},\, X_i^{(1)}\big],$$

where $X_i^{(x)}, X_i^{(z)}, X_i^{(k)} \in \mathbb{R}^d$, and $X_i^{(r)}, X_i^{(1)} \in \mathbb{R}$. Initially, for all tokens $i \le N$,

$$X_i^{(x)} = x_i, \qquad X_i^{(r)} = \|x_i\|^2, \qquad X_i^{(k)} = 0_d, \qquad X_i^{(1)} = 1.$$

The final data token $i = N$ is also the state token; initially $X_N^{(z)} = z_0$, while $X_i^{(z)} = 0_d$ for $i < N$.

Let L be the number of transformer layers. We take $L = S$, one layer per Euler step, and write the schedule as

$$\tau = \{(t_\ell, h_\ell)\}_{\ell=0}^{L-1}.$$

We prove the result by induction over layers. The invariant is that after ℓ blocks, the state token stores z_ℓ in its z-block, while every token retains x_i, $\|x_i\|^2$, and 1 in its x-, r-, and constant blocks and has a zero k-block. The other tokens' z-blocks are unrestricted. The base case follows from the prompt encoding. It remains to construct one block $B_{t,h}$ that maps

$$z \mapsto z + hv_t(z)$$

and resets the scratch block.

B.1 Attention layer convention

A single-head attention layer is specified by fixed matrices

$$W_Q, W_K \in \mathbb{R}^{p \times d_q}, \qquad W_V \in \mathbb{R}^{p \times p}.$$

For $X \in \mathbb{R}^{N\times p}$, define

$$q_i = X_i W_Q, \qquad k_i = X_i W_K, \qquad v_i = X_i W_V.$$

Under ordinary full or causal self-attention, the final token attends to all N tokens, including itself. Hence

$$a_{N,i}(X) = \frac{\exp(\langle q_N, k_i\rangle)}{\sum_{j=1}^{N}\exp(\langle q_N, k_j\rangle)}, \qquad i = 1,\dots,N,$$

and

$$\mathrm{Attn}(X)_N = \sum_{i=1}^{N} a_{N,i}(X) v_i.$$

The usual $1/\sqrt{d_q}$ scaling can be absorbed into W_Q or W_K, so we omit it.

B.2 One closed-form diffusion step

Fix $t \in (0,1)$ and $h > 0$. By (5),

$$v_t(z) = -\frac{1}{1-t} z + \frac{1}{t(1-t)} k_t(z).$$

Thus it suffices for attention to compute $k_t(z)$.

Query, key, and value projection weights. Let

$$a_t = \frac{t}{(1-t)^2}, \qquad b_t = -\frac{t^2}{2(1-t)^2}.$$

Choose $d_q = d + 1$. With block ordering $(x, r, z, k, 1)$, define

$$W_Q^{(t)} = \begin{bmatrix} 0_{d\times d} & 0_{d\times 1} \\ 0_{1\times d} & 0 \\ a_t I_d & 0_{d\times 1} \\ 0_{d\times d} & 0_{d\times 1} \\ 0_{1\times d} & 1 \end{bmatrix}, \qquad W_K^{(t)} = \begin{bmatrix} I_d & 0_{d\times 1} \\ 0_{1\times d} & b_t \\ 0_{d\times d} & 0_{d\times 1} \\ 0_{d\times d} & 0_{d\times 1} \\ 0_{1\times d} & 0 \end{bmatrix}.$$

Therefore

$$q_i = X_i W_Q^{(t)} = \big[a_t X_i^{(z)},\, X_i^{(1)}\big], \qquad k_i = X_i W_K^{(t)} = \big[X_i^{(x)},\, b_t X_i^{(r)}\big].$$

For the state token N and any data token $i \le N$, including $i = N$,

$$\langle q_N, k_i\rangle = \frac{t}{(1-t)^2}\langle z, x_i\rangle - \frac{t^2}{2(1-t)^2}\|x_i\|^2.$$

Moreover,

$$-\frac{\|z - t x_i\|^2}{2(1-t)^2} = \frac{t}{(1-t)^2}\langle z, x_i\rangle - \frac{t^2}{2(1-t)^2}\|x_i\|^2 - \frac{\|z\|^2}{2(1-t)^2}.$$

The last term is independent of i, so it cancels in the softmax over data tokens. Hence

$$a_{N,i}(X) = w_i(t,z), \qquad i = 1,\dots,N,$$

where $w_i(t,z)$ are the responsibility weights defined in (2).

The value projection is the block matrix

$$W_V^{(t)} = \begin{bmatrix} 0_{d\times d} & 0_{d\times 1} & 0_{d\times d} & tI_d & 0_{d\times 1} \\ 0_{1\times d} & 0 & 0_{1\times d} & 0_{1\times d} & 0 \\ 0_{d\times d} & 0_{d\times 1} & 0_{d\times d} & 0_{d\times d} & 0_{d\times 1} \\ 0_{d\times d} & 0_{d\times 1} & 0_{d\times d} & 0_{d\times d} & 0_{d\times 1} \\ 0_{1\times d} & 0 & 0_{1\times d} & 0_{1\times d} & 0 \end{bmatrix}.$$

Thus, for a data token i,

$$v_i = X_i W_V^{(t)} = [0_d, 0, 0_d, t x_i, 0],$$

and the attention output at the state token satisfies

$$\mathrm{Attn}(X)_N^{(k)} = \sum_{i=1}^{N} a_{N,i}(X) t x_i = \sum_{i=1}^{N} w_i(t,z)(t x_i) = k_t(z).$$

Feedforward update. Let

$$Y = X + \mathrm{Attn}(X).$$

The feedforward layer is the row-wise linear map below, realizable exactly by a two-layer ReLU network using $u = \mathrm{ReLU}(u) - \mathrm{ReLU}(-u)$:

$$\mathrm{FF}_{t,h}(Y_i) = Y_i W_{\mathrm{FF}}^{(t,h)},$$

where, with the same block ordering $(x, r, z, k, 1)$,

$$W_{\mathrm{FF}}^{(t,h)} = \begin{bmatrix} 0_{d\times d} & 0_{d\times 1} & 0_{d\times d} & 0_{d\times d} & 0_{d\times 1} \\ 0_{1\times d} & 0 & 0_{1\times d} & 0_{1\times d} & 0 \\ 0_{d\times d} & 0_{d\times 1} & -\frac{h}{1-t} I_d & 0_{d\times d} & 0_{d\times 1} \\ 0_{d\times d} & 0_{d\times 1} & \frac{h}{t(1-t)} I_d & -I_d & 0_{d\times 1} \\ 0_{1\times d} & 0 & 0_{1\times d} & 0_{1\times d} & 0 \end{bmatrix}.$$

Equivalently,

$$\mathrm{FF}_{t,h}^{(z)}(Y_i) = -\frac{h}{1-t} Y_i^{(z)} + \frac{h}{t(1-t)} Y_i^{(k)}, \qquad \mathrm{FF}_{t,h}^{(k)}(Y_i) = -Y_i^{(k)},$$

and all other output blocks are zero. For the state token,

$$Y_N^{(z)} = z, \qquad Y_N^{(k)} = k_t(z).$$

Therefore the residual update gives

$$B_{t,h}(X)_N^{(z)} = Y_N^{(z)} + \mathrm{FF}_{t,h}^{(z)}(Y_N) = z - \frac{h}{1-t} z + \frac{h}{t(1-t)} k_t(z) = z + h v_t(z),$$

and

$$B_{t,h}(X)_N^{(k)} = Y_N^{(k)} + \mathrm{FF}_{t,h}^{(k)}(Y_N) = 0_d.$$

Thus $B_{t,h}$ implements one closed-form diffusion Euler step, preserves every token's x-, r-, and constant blocks, and resets every k-block. The other z-blocks may change, but do not enter the key or value projections.

B.3 Completion of the induction

For each layer $\ell = 0, \ldots, L-1$, choose

$$B_\ell = B_{t_\ell, h_\ell}.$$

The parameter choice $\theta_{\mathrm{CFD}}^\star$ is obtained by stacking

$$\mathcal{T}_{\theta_{\mathrm{CFD}}^\star} = B_{L-1} \circ \cdots \circ B_0.$$

The base case $\ell = 0$ follows from the prompt encoding. If the induction invariant holds at layer ℓ, then B_ℓ updates

$$z_\ell \mapsto z_\ell + h_\ell v_{t_\ell}(z_\ell) = z_{\ell+1},$$

resets every k-block to zero, and preserves all x-, r-, and constant blocks. Hence the invariant holds at layer $\ell + 1$. By induction, after L layers the state token stores z_L. Therefore

$$\Pi_{\mathrm{state}} \left(\mathcal{T}_{\theta_{\mathrm{CFD}}^\star} \big(\mathrm{Enc}(\{x_i\}_{i=1}^N, z_0) \big) \right) = z_L,$$

where z_L is generated by the closed-form diffusion recursion.

B.4 Smoothed closed-form diffusion

The smoothed case uses the same induction argument, with one modification: the one-step block computes $k_{\sigma,t}(z)$ instead of $k_t(z)$. Fix $t \in (0, 1)$, $h > 0$, smoothing level $\sigma \geq 0$, and perturbations

$$\epsilon_1, \ldots, \epsilon_M \in \mathbb{R}^d.$$

Recall that

$$k_{\sigma,t}(z) = \frac{1}{M} \sum_{m=1}^{M} k_t(z + \sigma \epsilon_m).$$

Thus we compute $k_t(z + \sigma\epsilon_m)$ for each m and average the results.

Use M attention heads. For head m, keep $W_K^{(t)}$ and $W_V^{(t)}$ as above and replace $W_Q^{(t)}$ by

$$W_{Q,m}^{(\sigma,t)} = \begin{bmatrix} 0_{d\times d} & 0_{d\times 1} \\ 0_{1\times d} & 0 \\ a_t I_d & 0_{d\times 1} \\ 0_{d\times d} & 0_{d\times 1} \\ a_t \sigma \epsilon_m^\top & 1 \end{bmatrix}.$$

Then the state-token query in head m is

$$q_N^{(m)} = \big[a_t(z + \sigma\epsilon_m),\, 1\big].$$

For a data token $i \le N$,

$$\langle q_N^{(m)}, k_i \rangle = \frac{t}{(1-t)^2}\langle z + \sigma\epsilon_m, x_i\rangle - \frac{t^2}{2(1-t)^2}\|x_i\|^2.$$

This equals the Gaussian responsibility logit for the perturbed state $z + \sigma\epsilon_m$, up to an additive term independent of i. Hence

$$a_{N,i}^{(m)}(X) = w_i(t, z + \sigma\epsilon_m).$$

Using the same value projection $W_V^{(t)}$, head m outputs

$$\sum_{i=1}^N w_i(t, z + \sigma\epsilon_m)(tx_i) = k_t(z + \sigma\epsilon_m)$$

in its scratch output. The multi-head output projection averages these heads:

$$\frac{1}{M}\sum_{m=1}^M k_t(z + \sigma\epsilon_m) = k_{\sigma,t}(z).$$

The same feedforward matrix $W_{\mathrm{FF}}^{(t,h)}$, with $k_t(z)$ replaced by $k_{\sigma,t}(z)$, updates the state token to

$$z + hv_{\sigma,t}(z) = z - \frac{h}{1-t}z + \frac{h}{t(1-t)}k_{\sigma,t}(z),$$

and resets the scratch block.

Applying this construction at every layer $\ell = 0, \dots, L-1$ with $t = t_\ell$ and $h = h_\ell$, the same induction gives a parameter choice $\theta^\star_{\sigma\text{-CFD}}$ such that

$$\Pi_{\mathrm{state}}\left(\mathcal{T}_{\theta^\star_{\sigma\text{-CFD}}}\big(\mathrm{Enc}(\{x_i\}_{i=1}^N, z_0)\big)\right) = z_L,$$

where z_L is generated by the smoothed closed-form diffusion recursion. This completes the proof.

C Proof of Theorem 2

C.1 Prompt encoding for EFS

We now describe the prompt used in Theorem 2. The EFS parameters $\tau = (K, \gamma, s, \epsilon)$ are fixed and compiled into the transformer weights. The prompt stores only the instance-specific quantities: the initial particles $x_1^{(0)}, \dots, x_N^{(0)}$ and the reference point $z^{(K)}$.

We use $m = N + 2$ tokens. The first N rows are particle tokens, row $N + 1$ stores the reference point $z^{(K)}$, and row $N + 2$ is the state token. Each token is partitioned into the following blocks:

$$\big[(x^{[0]}, x^{[1]}, \dots, x^{[K]}) \mid (r^{[0]}, r^{[1]}, \dots, r^{[K]}) \mid z \mid r_z \mid g \mid c \mid u \mid b \mid \eta \mid \xi\big].$$

Here $x^{[j]} \in \mathbb{R}^d$ stores the j-th forward particle iterate and $r^{[j]} \in \mathbb{R}$ its squared norm; these quantities are approximated during the computation. The blocks $z \in \mathbb{R}^d$ and $r_z \in \mathbb{R}$ store the backward point and its squared norm. The vector $g \in \mathbb{R}^d$ accumulates repulsive interactions, $c = 1$ is constant, and

$u \in \mathbb{R}^d$, $b \in \mathbb{R}$ are separate attention scratch registers. The bits η and ξ select the state and reference tokens, respectively. Therefore,

$$p_0 = (K+3)d + K + 3, \qquad p = p_0 + d + 3 = (K+4)d + K + 6.$$

The first six blocks form $X_{\text{base}} \in \mathbb{R}^{(N+2)\times p_0}$:

$$X_{\text{base}} = \left[\begin{array}{c|c|c|c|c|c} (x_1^{(0)})^\top,\ 0_d^\top, \dots, 0_d^\top & \|x_1^{(0)}\|^2,\ 0, \dots, 0 & 0_d^\top & 0 & 0_d^\top & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ (x_N^{(0)})^\top,\ 0_d^\top, \dots, 0_d^\top & \|x_N^{(0)}\|^2,\ 0, \dots, 0 & 0_d^\top & 0 & 0_d^\top & 1 \\ 0_d^\top,\ 0_d^\top, \dots, 0_d^\top & 0,\ 0, \dots, 0 & (z^{(K)})^\top & \|z^{(K)}\|^2 & 0_d^\top & 1 \\ 0_d^\top,\ 0_d^\top, \dots, 0_d^\top & 0,\ 0, \dots, 0 & 0_d^\top & 0 & 0_d^\top & 1 \end{array}\right].$$

Writing e_i for the i-th standard basis vector in $\mathbb{R}^{N+2}$, the complete prompt $X_{\text{EFS}} \in \mathbb{R}^{(N+2)\times p}$ is

$$X_{\text{EFS}} = \big[X_{\text{base}} \mid 0_{(N+2)\times d} \mid 0_{N+2} \mid e_{N+2} \mid e_{N+1}\big].$$

Thus $\eta_i = 1$ only for the state token and $\xi_i = 1$ only for the reference token; $\chi_i = 1 - \eta_i - \xi_i$ selects the particle tokens. These constant features are preserved exactly.

Initially, particle row $i \le N$ stores $x_i^{(0)}$ and $\|x_i^{(0)}\|^2$, while the later blocks $x^{[1]}, \dots, x^{[K]}$ and $r^{[1]}, \dots, r^{[K]}$ are zero. During the forward phase, the transformer fills these blocks with approximations of the forward EFS iterates $x_i^{(j)}$ and their squared norms. The reference row stores $z^{(K)}$ and $\|z^{(K)}\|^2$. The final row is the state token used during the backward phase. The readout extracts the z-block of this final row:

$$\Pi_{\text{state}}(X) = X_{N+2}^{(z)}.$$

After the transformer computation, this readout approximates the EFS output $z^{(0)}$. All $x^{[j]}$- and $r^{[j]}$-blocks of the reference and state tokens remain exactly zero. The selectors distinguish a particle at the origin from a reference or state token. The register r_z is reserved for the squared norm and is never used as attention scratch space.

For an attention head h, queries, keys, and values are

$$q_i = X_i W_{Q,h}, \qquad k_i = X_i W_{K,h}, \qquad v_i = X_i W_{V,h}.$$

We use standard softmax attention over the $N+2$ tokens:

$$\alpha_{i\ell}^{(h)} = \frac{\exp(\langle q_i, k_\ell \rangle)}{\sum_{r=1}^{N+2} \exp(\langle q_i, k_r \rangle)}, \qquad \ell = 1, \dots, N+2,$$

and

$$\mathrm{Attn}^{(h)}(X)_i = \sum_{\ell=1}^{N+2} \alpha_{i\ell}^{(h)} v_\ell.$$

C.2 Explicit kernel head

Fix a stage $j \in \{0, \dots, K\}$ and a kernel parameter $\lambda > 0$. We first consider the forward-stage query token $i \le N$. The goal is to recover

$$\sum_{\ell=1}^{N} e^{-\lambda \|x_i^{[j]} - x_\ell^{[j]}\|^2} \big(x_i^{[j]} - x_\ell^{[j]}\big).$$

Let the query/key dimension be $d+1$. Define $W_{Q,\lambda}^{[j]} \in \mathbb{R}^{p\times(d+1)}$ by

$$W_{Q,\lambda}^{[j]}: \qquad x^{[j]} \mapsto 2\lambda x^{[j]}, \qquad c \mapsto 1,$$

with all other input blocks mapped to zero. Equivalently,

$$q_i = [2\lambda x_i^{[j]} \mid 1].$$

Define $W_{K,\lambda}^{[j]} \in \mathbb{R}^{p\times(d+1)}$ by

$$W_{K,\lambda}^{[j]}: \qquad x^{[j]} \mapsto x^{[j]}, \qquad r^{[j]} \mapsto -\lambda r^{[j]},$$

with all other input blocks mapped to zero. Thus

$$k_\ell = [x_\ell^{[j]} \mid -\lambda r_\ell^{[j]}].$$

First consider the ideal arithmetic computation with exact squared-norm registers; errors in these registers are treated below. For a particle token $\ell \le N$,

$$\langle q_i, k_\ell\rangle = 2\lambda\langle x_i^{[j]}, x_\ell^{[j]}\rangle - \lambda\|x_\ell^{[j]}\|^2.$$

For both the reference token $N+1$ and the state token $N+2$, the $x^{[j]}$- and $r^{[j]}$-blocks are zero, so their keys and logits are zero. Consequently, $\alpha_{i,N+1} = \alpha_{i,N+2}$.

Let $\alpha_{i\ell}$ be the attention weight on token ℓ, and write $\alpha_{i0} \equiv \alpha_{i,N+2}$ for the attention weight on the state token. Then

$$\frac{\alpha_{i\ell}}{\alpha_{i0}} = \exp\Big(2\lambda\langle x_i^{[j]}, x_\ell^{[j]}\rangle - \lambda r_\ell^{[j]}\Big), \qquad \ell \le N.$$

Multiplying by $e^{-\lambda r_i^{[j]}}$ gives

$$e^{-\lambda r_i^{[j]}}\frac{\alpha_{i\ell}}{\alpha_{i0}} = e^{-\lambda\|x_i^{[j]} - x_\ell^{[j]}\|^2}.$$

Now define the value projection $W_{V,\lambda}^{[j]} \in \mathbb{R}^{p\times p}$ by

$$W_{V,\lambda}^{[j]}: \qquad x^{[j]} \mapsto u, \qquad \eta \mapsto b,$$

with all other output blocks zero. Initialize $u = b = 0$ before this head. After the attention residual, these registers contain

$$u_i = \sum_{\ell=1}^{N} \alpha_{i\ell} x_\ell^{[j]}, \qquad b_i = \alpha_{i,N+2} = \alpha_{i0}.$$

The total attention mass on the particle tokens is $1 - 2b_i$. Hence the continuous map

$$\mathcal{R}_\lambda(y, r, u, b) = \frac{e^{-\lambda r}}{b}\big((1-2b)y - u\big)$$

satisfies

$$\mathcal{R}_\lambda(x_i^{[j]}, r_i^{[j]}, u_i, b_i) = \sum_{\ell=1}^{N} e^{-\lambda\|x_i^{[j]} - x_\ell^{[j]}\|^2}(x_i^{[j]} - x_\ell^{[j]}).$$

For fixed λ, logits are bounded on a compact register domain, so b_i is bounded away from zero. A positionwise ReLU feedforward network can therefore approximate $\mathcal{R}_\lambda$ uniformly there. It writes the interaction into the accumulator g and resets u, b to zero; the routing and accumulation are specified below.

C.3 Backward-stage kernel head

For the backward stage, the query token is the state token $N+2$, whose z-block stores the current backward iterate $z^{(j)}$. The key and value projections remain $W_{K,\lambda}^{[j]}$ and $W_{V,\lambda}^{[j]}$. The query projection is replaced by $\widetilde{W}_{Q,\lambda}$, defined by

$$\widetilde{W}_{Q,\lambda}: \qquad z \mapsto 2\lambda z, \qquad c \mapsto 1,$$

with all other input blocks mapped to zero. Thus

$$q_{N+2} = [2\lambda z_{N+2} \mid 1], \qquad k_\ell = [x_\ell^{[j]} \mid -\lambda r_\ell^{[j]}].$$

Using $\mathcal{R}_\lambda(z_{N+2}, r_{z,N+2}, u_{N+2}, b_{N+2})$, the same argument yields, up to arbitrary uniform approximation,

$$\sum_{\ell=1}^{N} e^{-\lambda\|z_{N+2} - x_\ell^{[j]}\|^2}\big(z_{N+2} - x_\ell^{[j]}\big).$$

C.4 Approximating the EFS interaction field

Let $a = s/2 + 1 > 0$. Since $\epsilon > 0$, the Laplace identity gives

$$\frac{v}{(\|v\|^2+\epsilon)^a} = \frac{1}{\Gamma(a)} \int_0^\infty t^{a-1} e^{-t\epsilon} e^{-t\|v\|^2} v \, dt.$$

On the compact domain, all relevant differences satisfy $\|v\| \le R$. Hence, for every $\delta > 0$, there exist $H \in \mathbb{N}$, nodes $\lambda_1, \dots, \lambda_H > 0$, and coefficients $\omega_1, \dots, \omega_H > 0$ such that

$$\sup_{\|v\|\le R} \left\| \frac{v}{(\|v\|^2+\epsilon)^a} - \sum_{h=1}^{H} \omega_h e^{-\lambda_h \|v\|^2} v \right\| \le \delta.$$

Indeed, the integrand norm is bounded by $Rt^{a-1}e^{-\epsilon t}/\Gamma(a)$, an integrable function. Truncating the integral near zero and infinity and taking a finite Riemann sum on the remaining interval gives the stated uniform approximation.

We evaluate the H kernel heads *sequentially in separate blocks*. Starting from $g = u = b = 0$, the feedforward map after head h adds $\omega_h \mathcal{R}_{\lambda_h}(y, r, u, b)$, up to uniform approximation, to g on the active tokens and resets u, b exactly. The active selector is χ during the forward phase and η during the backward phase. Thus each head's nonlinear normalization is evaluated before summing its contribution. At the end, g approximates

$$\sum_{\ell=1}^{N} \frac{y - x_\ell^{[j]}}{(\|y - x_\ell^{[j]}\|^2 + \epsilon)^{s/2+1}}$$

up to uniform error.

The attractive term is affine after summation:

$$\sum_{\ell=1}^{N} (y - x_\ell^{[j]}) = Ny - \sum_{\ell=1}^{N} x_\ell^{[j]}.$$

It is computed by a uniform-attention head. Explicitly, choose

$$W_Q^{\mathrm{avg}} = 0, \qquad W_K^{\mathrm{avg}} = 0, \qquad W_V^{\mathrm{avg}} : \; x^{[j]} \mapsto u,$$

with all other output blocks zero. Then all logits are zero, so attention returns

$$\mathrm{Attn}^{\mathrm{avg}}(X)_i^{(u)} = \frac{1}{N+2} \sum_{\ell=1}^{N+2} x_\ell^{[j]} = \frac{1}{N+2} \sum_{\ell=1}^{N} x_\ell^{[j]},$$

because the reference and state tokens have zero $x^{[j]}$-blocks. This head writes into the cleared register u, leaving g unchanged. Combining the two registers gives the force approximation

$$\widehat{F}(y) = c_N \big(Ny - (N+2)u - g \big), \qquad c_N = \frac{2}{N(N-1)},$$

to the normalized field

$$F_{N,\epsilon}(y; x_1, \dots, x_N) = c_N \sum_{\ell=1}^{N} \left(y - x_\ell - \frac{y - x_\ell}{(\|y - x_\ell\|^2 + \epsilon)^{s/2+1}} \right).$$

This is $\nabla_{x_i} E_{N,\epsilon}$ at $y = x_i$ and $\nabla_z V_{N,\epsilon}$ at $y = z$. The term with $x_\ell = y$ is zero.

C.5 Forward and backward simulation

Exact routing and scratch reset. The feedforward network is shared across tokens. It can nevertheless restrict an approximate continuous update to tokens selected by a fixed bit $t \in \{0, 1\}$. To see this, write a two-layer ReLU approximant in column notation as $A\,\mathrm{ReLU}(Bw + a) + d$. On the compact register domain, choose M larger than every component of $Bw + a$. Then

$$A\,\mathrm{ReLU}\big(Bw + a - M(1-t)\mathbf{1}\big) + dt$$

equals the approximant when $t = 1$ and is exactly zero when $t = 0$. The affine term dt is also realizable by ReLU units. We use $t = \chi$ or $t = \eta$, as appropriate. Immutable output coordinates are identically zero in the feedforward residual; scratch reset uses the exact linear residual $(u, b) \mapsto (-u, -b)$, and similarly for g at the end of a stage. Thus inactive trajectory registers and selectors remain unchanged exactly, including the zero particle-history blocks of the reference and state tokens.

Forward stages. At stage $j = 0, \dots, K-1$, the H sequential kernel blocks followed by the uniform-attention block compute $\widehat{F}(x_i^{[j]})$. The final feedforward map, gated by χ_i, writes into the previously zero future blocks

$$x_i^{[j+1]} = x_i^{[j]} - \gamma \widehat{F}(x_i^{[j]}), \qquad r_i^{[j+1]} \approx \|x_i^{[j+1]}\|^2, \qquad i \le N,$$

and resets g, u, b on every token. The coordinate update is affine in the available registers; the squared-norm update is approximated jointly with it by the feedforward network. Earlier trajectory blocks are preserved. The particle and state z, r_z-blocks remain zero throughout the forward phase, while the reference z, r_z-blocks remain unchanged.

Copying the reference point. After the forward phase, use $W_Q = W_K = 0$ and a value map $z \mapsto u$, $r_z \mapsto b$. Only the reference row can have nonzero z, r_z, so the attention residual gives

$$u_i = \frac{z^{(K)}}{N+2}, \qquad b_i = \frac{\|z^{(K)}\|^2}{N+2}.$$

An η_i-gated affine feedforward map writes $(N+2)u_i, (N+2)b_i$ into the state token's initially zero z, r_z-blocks and resets u, b. This copy is exact on the bounded domain and leaves the reference token unchanged.

Backward stages. For $j = K, K-1, \dots, 1$, use the backward query, the stored j-th particle trajectory, and the selector η. The same sequential construction approximates $F_{N,\epsilon}(z; x_1^{[j]}, \dots, x_N^{[j]})$. The final feedforward map updates only the state token:

$$z \leftarrow z + \gamma \widehat{F}(z), \qquad r_z \approx \|z + \gamma \widehat{F}(z)\|^2,$$

where the right-hand sides use the old registers. All trajectory and reference registers are preserved, and g, u, b are reset on every token.

Uniform error control. First compare the exact recurrence with the ideal quadrature computation, using exact squared norms. On a compact neighborhood of all admissible trajectories with positive margin, the $2K$ exact stage maps, including particle histories, have a common Lipschitz bound $L_0 \ge 1$ in the maximum of the vector-register Euclidean norms, since $\epsilon > 0$. A quadrature error δ per pair gives force error at most $c_N N \delta$, so

$$d_{q+1} \le L_0 d_q + \gamma c_N N \delta, \qquad d_0 = 0.$$

Choose δ so that

$$\gamma c_N N \delta \sum_{q=0}^{2K-1} L_0^q \le \varepsilon/2$$

and the left-hand side is smaller than half the neighborhood margin. Choose the quadrature after fixing δ. Induction keeps its ideal trajectories in this neighborhood and gives terminal error at most $\varepsilon/2$.

For the fixed quadrature, the ideal program has J locally Lipschitz blocks and compact intermediate register sets; kernel attention weights b are bounded away from zero. Choose a common neighborhood margin $\rho > 0$ and block Lipschitz bound $C \ge 1$. Use the maximum, over all tokens and registers, of Euclidean vector norms and absolute scalar values. Approximating each nonlinear feedforward map within κ, with exact routing and resets, gives

$$e_{q+1} \le C e_q + \kappa, \qquad e_0 = 0.$$

Choose κ such that

$$\kappa \sum_{q=0}^{J-1} C^q \le \min\{\rho/2, \varepsilon/2\}.$$

This induction accounts for errors in every stored squared norm, attention logit, interaction accumulator, and state update. Inactive history blocks and scratch resets remain exact. The triangle inequality now gives

$$\left\| \Pi_{\text{state}} \left(\mathcal{T}_{\theta^\star_{\text{EFS}}} \big(\text{Enc}_{\text{EFS}}(x_1^{(0)}, \dots, x_N^{(0)}, z^{(K)}) \big) \right) - z^{(0)} \right\| \le \varepsilon.$$

All choices depend only on the fixed parameters, compact domain, and tolerance. The construction uses $J = 2K(H+1)+1$ blocks and token width $p = (K+4)d + K + 6$, with sufficiently large finite feedforward hidden widths. This proves Theorem 2.

D GUARANTEES, RESOURCES, AND INTERPRETATION OF THE LAYERWISE DIAGNOSTICS

D.1 THREE NOTIONS OF GENERALIZATION IN IN-CONTEXT SAMPLING

Let $X_1, \dots, X_N$ be prompt samples from a population distribution μ_0, and let $\widehat{\mu}_N = N^{-1} \sum_{i=1}^{N} \delta_{X_i}$ be their empirical measure. We distinguish three claims about an in-context sampler.

Algorithmic generalization across prompts. For fixed N, dimension, and sampling schedule, Theorem 1 constructs one set of transformer parameters that exactly executes the prescribed finite-step closed-form diffusion recursion for every admissible prompt and initial state. The parameters do not depend on the realized samples. Consequently, the same transformer can execute this algorithm on prompts drawn from distributions not used to choose its parameters. This is uniform algorithmic execution across prompts, not a claim that the output has law μ_0. In its unsmoothed continuous-time terminal limit, closed-form diffusion targets $\widehat{\mu}_N$; at a finite Euler step size, our theorem identifies the transformer output with the corresponding discrete recursion.

Finite-sample novelty. An output is novel relative to a prompt if it lies outside $\{X_1, \dots, X_N\}$. The smoothed construction exactly simulates a prescribed smoothed recursion. Existing results characterize limiting outputs of smoothed closed-form diffusion in terms of barycenters of tuples of prompt points (Scarvelis et al., 2025). When such a barycenter differs from every prompt point, this provides finite-sample novelty under the conditions of that characterization. The simulation theorem alone does not guarantee novelty for every prompt, smoothing choice, or finite discretization: it establishes execution of the smoothed recursion, while smoothing changes the recursion's vector field. Furthermore, an output can differ from every prompt point without having law μ_0.

Population generation. Population generation concerns the output law rather than whether the output equals an observed point. The population result for the original Estimation-Free Sampling (EFS) reverse flow establishes convergence of its output law to μ_0 under its mean-field and continuous-time assumptions, including the potential regime $s = d - 2$ (Daneshmand & Soleymani, 2025). Our Theorem 2 approximates a regularized, finite-particle recurrence with an explicit reverse-time Euler step. That step is not the inverse optimization used by the original EFS algorithm. In particular, the cited population theorem does not directly apply to our recurrence for arbitrary s, fixed $\epsilon > 0$, and fixed step size. The logarithmic potential $s = 0$ used in Figure 5 is an experimental diagnostic, distinct from the regime of that population theorem.

To make the gap precise, let $Z^{\mathrm{rev}}_{N,K}$ be the output of our reverse-Euler recurrence, $\widehat{Z}_{N,K}$ its transformer approximation, and $Z^{\mathrm{orig}}_{N,K}$ the output of a separately specified discretization of the original EFS procedure. If the random inputs satisfy the compact-domain assumption of Theorem 2 almost surely, coupling the transformer and its target recurrence with the same inputs yields

$$W_1\Big(\mathcal{L}(\widehat{Z}_{N,K}), \mathcal{L}(Z^{\mathrm{rev}}_{N,K})\Big) \le \varepsilon_{\mathrm{tr}}. \tag{8}$$

If the relevant first moments exist, the triangle inequality gives

$$\begin{aligned} W_1\Big(\mathcal{L}(\widehat{Z}_{N,K}), \mu_0\Big) \le \varepsilon_{\mathrm{tr}} &+ W_1\Big(\mathcal{L}(Z^{\mathrm{rev}}_{N,K}), \mathcal{L}(Z^{\mathrm{orig}}_{N,K})\Big) \\ &+ W_1\Big(\mathcal{L}(Z^{\mathrm{orig}}_{N,K}), \mu_0\Big). \end{aligned} \tag{9}$$

Theorem 2 bounds the first term only. The second requires a comparison of reverse Euler with inverse optimization, including discretization and regularization errors. The third requires the assumptions and limits of a population result for the original procedure. Hence, the decomposition is not itself a finite-sample population guarantee.

Thus, our transformer theorems establish uniform execution of prompt-dependent samplers. Smoothed closed-form diffusion can yield finite-sample novelty under appropriate conditions, while the original EFS method has a conditional asymptotic population guarantee. The present theorems do not establish all three properties for one finite transformer implementation.

D.2 Architecture, supplied features, and accuracy

The following counts refer to the implementations actually proved in Appendices B and C; L is the number of diffusion steps, K is the number of forward and backward particle stages, and H is the chosen number of Gaussian-kernel quadrature terms.

Construction	Tokens and width	Attention, depth, feedforward
Closed-form diffusion	N, $p = 3d + 2$	One head per step; query/key size $d + 1$; L blocks; at most $2p$ ReLU hidden units
Smoothed diffusion	N, $p = 3d + 2$	M heads per step; L blocks; at most $2p$ ReLU hidden units
EFS recurrence	$N+2$, $p = (K+4)d+K+6$	One head per block; H kernel and one averaging block per stage; $2K(H + 1) + 1$ blocks; finite feedforward width

The EFS construction evaluates the H kernel heads *sequentially*, because a tokenwise nonlinear normalization is applied after each head. It is therefore not a depth-$2K$ parallel-head construction. Its current universal-approximation argument gives no explicit feedforward-width bound in d and $\varepsilon_{\mathrm{tr}}$; a polynomial-width claim would require an additional quantitative approximation proof. The dependence on N also appears in attention computation and fixed coefficients, even when the token width is independent of N.

We can make the quadrature part of the accuracy dependence explicit. Set $a = s/2 + 1 > 0$ and assume $\|y - x_i\| \le D$ throughout the compact domain, with $D > 0$. For a desired *per-pair* force error $\delta > 0$, define

$$T = \frac{2}{\epsilon} \log\left(1 + \frac{2D(2/\epsilon)^a}{\delta}\right), \qquad H = \max\left\{1, \left\lceil \frac{2D^3\epsilon^{-a}T}{\delta} \right\rceil\right\}.$$

Divide $[0, T]$ into H equal intervals I_h, take their midpoints $\lambda_h > 0$, and set $\omega_h = \Gamma(a)^{-1} \int_{I_h} t^{a-1}e^{-\epsilon t}\, dt$. The Laplace identity in Appendix C then implies

$$\sup_{\|v\| \le D} \left\| \frac{v}{(\|v\|^2 + \epsilon)^a} - \sum_{h=1}^{H} \omega_h e^{-\lambda_h \|v\|^2} v \right\| \le \delta. \tag{10}$$

Indeed, the omitted tail is at most $D(2/\epsilon)^a e^{-\epsilon T/2} \le \delta/2$. On $[0, T]$, the difference between the integrand at t and at the midpoint of its interval is bounded by D^3T/H; the total integrated weight is at most ϵ^{-a}, giving another error of at most $\delta/2$. The resulting force error is at most $c_N N\delta$ per stage, where $c_N = 2/[N(N - 1)]$. To obtain terminal error $\varepsilon_{\mathrm{tr}}$, choose δ so that $\gamma c_N N\delta \sum_{q=0}^{2K-1} L_0^q \le \varepsilon_{\mathrm{tr}}/2$, and choose the feedforward accuracy as in the last paragraph of Appendix C. The Lipschitz constant L_0 depends on the chosen compact neighborhood, s, ϵ, and step size; no dimension-free end-to-end complexity is asserted.

In the exact theorem encodings, the instance-dependent inputs are the samples and a starting/reference point. A deterministic transformer needs this latter point as a random seed if a stochastic output law is desired. The encodings additionally contain squared norms, fixed selector bits, and zero-initialized scratch coordinates. Norms carry no information beyond the raw vectors: on a compact set, a preliminary positionwise ReLU network can approximate them to arbitrary accuracy, followed by the same softmax construction and a propagated finite-step error bound. This gives an *approximate* raw-vector variant, while the stated closed-form diffusion theorem uses exact supplied norms. Selectors and scratch slots are fixed workspace for routing and storing iterates, not learned information about μ_0.

D.3 Exact relationship between spherical MMD and interaction energy

The potential in Figure 5 is selected by the specified interaction energy, not by fitting the observed layerwise curves. Its connection with RBF MMD is exact at the population level. Let μ be any probability measure on a sphere of radius r centered at the origin of a fixed linear subspace, let σ_r be uniform on that *same* sphere, and write $m_\mu = \int x\, d\mu(x)$ and $k_t(x, y) = e^{-t\|x-y\|^2}$. Define

$\mathcal{E}_{\epsilon,0}(\mu) = \iint W_\epsilon^{(0)}(x-y)\, d\mu(x) d\mu(y)$. Then, for $\epsilon > 0$,

$$\boxed{\mathcal{E}_{\epsilon,0}(\mu) - \mathcal{E}_{\epsilon,0}(\sigma_r) = -\|m_\mu\|^2 + \frac{1}{2}\int_0^\infty \frac{e^{-\epsilon t}}{t}\, \mathrm{MMD}^2_{k_t}(\mu, \sigma_r)\, dt.} \tag{11}$$

To prove the identity, the quadratic part contributes $\frac{1}{2}\iint \|x-y\|^2 d\mu(x) d\mu(y) = r^2 - \|m_\mu\|^2$, while its uniform counterpart equals r^2. Use $-\log A = \int_0^\infty (e^{-tA} - e^{-t})\, dt/t$ for $A > 0$. Rotational invariance on the common sphere makes $\int k_t(x,y) d\sigma_r(y)$ independent of x on that sphere, so the difference of the two within-measure kernel expectations is precisely $\mathrm{MMD}^2_{k_t}(\mu, \sigma_r)$. Regularization makes the integral finite. A one-bandwidth RBF MMD therefore captures one scale of the repulsive term, while the centroid term can move in the opposite direction; the two plots need not have identical shapes.

The plotted energy uses an off-diagonal finite-particle average and radius 0.78, whereas the MMD curves use finite-cloud statistics after normalization to unit radius. Equation (11) concerns population pair integrals with a *radius-matched* reference, so it does not identify these finite-sample ordinates. Rescaling the normalized vectors by r also rescales the Gaussian bandwidth by r^2. Row centering each hidden vector confines it to $\mathbf{1}^\perp$; for an exact spherical-uniform comparison, the uniform reference must be drawn in that same subspace. Finally, MMD and energy derived from the same hidden states are mathematically related diagnostics, not independent confirmation that pretrained transformers execute EFS.

D.4 What the normalization and trajectory experiments test

Unit-vector normalization retains angular structure but removes radial variation. A reduction in spherical MMD under this transformation does not by itself establish that the *unnormalized* representations are approaching any uniform law. A direct analysis-time control would instead subtract the *cloud* mean $\bar{h} = N^{-1}\sum_i h_i$ and divide all vectors by the *same* root-mean-square radius:

$$\widetilde{h}_i = \frac{h_i - \bar{h}}{\left(N^{-1}\sum_j \|h_j - \bar{h}\|^2\right)^{1/2}}.$$

This preserves relative radii. A complementary directional control compares the centered, unnormalized vectors with $r_i u_i$, where $r_i = \|h_i - \bar{h}\|$ and the u_i are independently uniform directions in the relevant subspace. Both controls require new measurements before robustness to *analysis-time* normalization can be claimed for the pretrained models in this paper.

Separately, in the controlled 16-layer two-moons setting, we compared a LayerNorm transformer to a transformer trained without LayerNorm using T-Fixup-style initialization. The latter avoids the collapse observed when LayerNorm is simply deleted. Projecting the input stage, each layer, and the output through the output head gives 18 clouds. Against a circle fitted with one center and radius for each cloud, the reported RBF-MMD minima occur at layers 5 and 8, respectively; sliced Wasserstein and angular Fourier discrepancies show the same qualitative decrease followed by an increase. Fitting a circle does not project individual points onto it, so this control retains radial deviations. This architectural ablation does not replace the distinct analysis-time controls above, and its qualitative agreement across three discrepancies does not demonstrate robustness for every model or potential.

The choice of the EFS potential is specified in advance by the algorithm. Equation (11) explains why its energy can track multiscale MMD on a matched sphere; it does not show that other attractive–repulsive potentials would fail or that the EFS potential is uniquely supported by the plots. Evaluating other s, ϵ values and separating the attractive and repulsive contributions would test that question. Likewise, the intermediate two-moons clouds illustrate a transport-like path, but the endpoint matching objective does not minimize a quadratic Wasserstein action and does not imply a W_2-optimal trajectory. Neither those plots nor the pretrained-model MMD values establish that a stronger U-shape improves generation: endpoint quality requires a separate controlled, preferably within-family comparison with fixed prompts and explicit sampling metrics.

D.5 Relationship to prior in-context denoising

Rosu et al. (Rosu et al., 2026) already relate attention to Gaussian-weighted denoising and analyze RBF attention as well as standard softmax under equal-norm conditions. Our closed-form diffu-

sion calculation instead supplies the norm feature explicitly and computes the empirical Gaussian responsibility with ordinary dot-product softmax for every admitted prompt and state; its output is then unrolled through the prescribed Euler schedule. This is a precise difference in assumptions and implementation, rather than a claim that the broad attention–denoising connection is new. The EFS-inspired theorem establishes expressivity for a different finite interacting-particle recurrence. Neither construction identifies the computation learned by a pretrained language model; the layerwise experiments are descriptive evidence of geometry compatible with several algorithms.

E Experiments Details

E.1 Smile Experiment

Figure 1 provided a controlled visual example of the sampling behavior studied in this work. We construct a two-dimensional smiling emoji distribution with four components: a noisy circular face boundary, two Gaussian eye clusters, and a crescent-shaped smile. The smile component is removed from the training distribution. We use a small GPT-2-style decoder-only transformer trained from scratch with 16 layers and 32-dimensional hidden embeddings. Each training example contains 64 input points, and the model is trained on 1,000 IID training sequences for 2,000 optimization steps.

Because the architecture is decoder-only and causal, we formulate the task in the same way as next-token prediction. For each IID sequence $(x_1, \ldots, x_{T+1})$, the model receives $(x_1, \ldots, x_T)$ as input and is trained to predict the shifted sequence $(x_2, \ldots, x_{T+1})$. The final target point x_{T+1} is not included in the input context, so it acts as a fresh IID sample from the same distribution. Since the samples are unordered, we train with a full-sequence optimal-transport matching loss rather than an index-wise regression loss. This encourages the predicted point cloud to match the target distribution, while remaining compatible with the next-token structure of the transformer.

For visualization, we plot only the prediction at the last position. It is produced after the model has attended to the entire in-context sequence. Earlier positions are also trained, but each of them is conditioned only on a shorter prefix because of the causal mask. The held-out smile gives the central qualitative test. Although smile-shaped samples are never included in the training distribution, the model can generate new IID samples along the same smile geometry.

E.2 Synthetic layerwise geometry

Figure 3 used the two-moons distribution to show how the learned sampler transforms an in-context point cloud across depth. We generate a pool of 5,000 2-dimensional points and split it at the point level. Using the same causal next-token sampling setup as in Subsection E.1, we train the model for 2,000 optimization steps on 1,000 IID training sequences, where each input sequence contains 64 points. To inspect the internal computation, we project the hidden states after selected transformer layers back to the two-dimensional output space using the model's output head. In early layers, the predicted points are concentrated in a small region, then expand in the middle layers toward a more uniform geometry, and finally contract back onto the structured two-moons distribution specified by the in-context samples. This layerwise trajectory visually supports the view of transformer depth as an iterative transport process. Full layer-by-layer visualization is provided in Fig 7.

E.3 Layerwise Uniformization in Large Language Models

Figure 2 shows that pretrained autoregressive language models reshape in-context semantic distributions across depth. We construct prompts from three topic vocabularies: animals, foods, and cities. For each model, we keep only tokenizer-eligible single-token words and use unique tokens, so each prompt contains up to 256 distinct words from one topic. We run 5 independent prompts per topic and record the raw hidden states after each transformer block, excluding the embedding layer, final layer normalization, and vocabulary logits. The complete semantic-token experiment was conducted across nine pretrained autoregressive language models; the corresponding layerwise MMD results are reported in Figure 8.

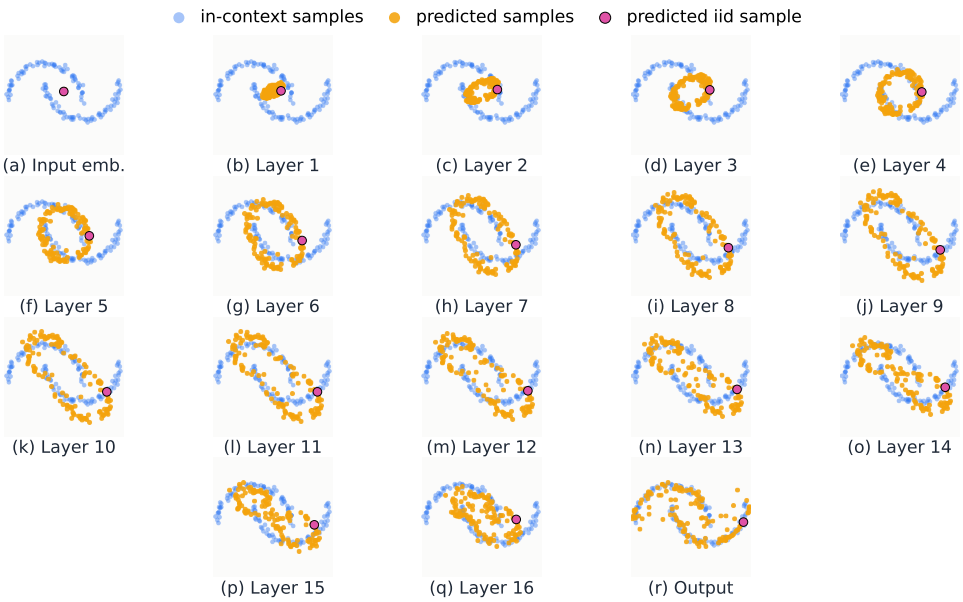


Figure 7: Geometry evolution of one test sample across the transformer layers.

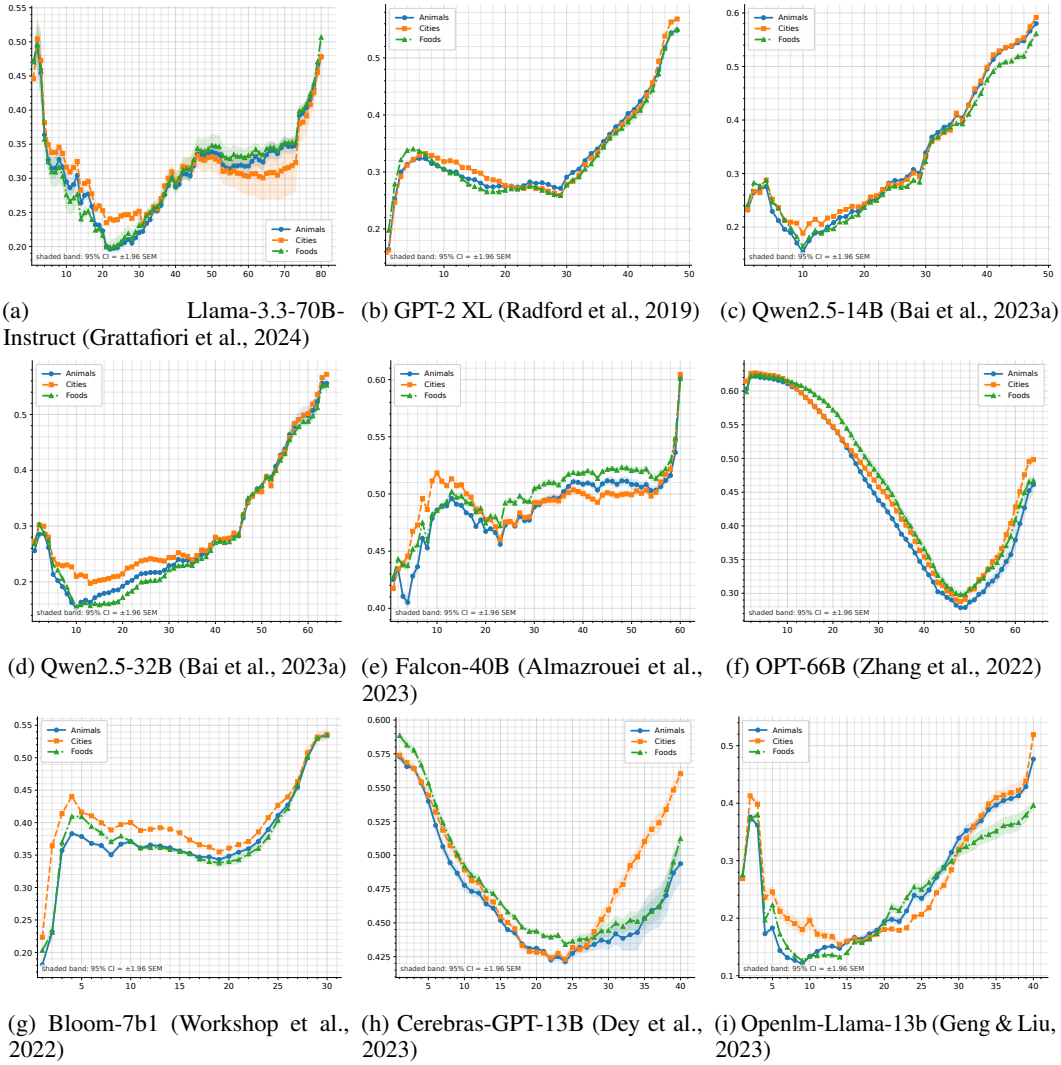
In Figure 4, we also repeat the same analysis on natural text prompts from the CBT dataset (Hill et al., 2015): for each story, we feed the first 256 tokens to the model, but compute the layerwise metrics only on the first occurrence of each unique token.

At each layer, we treat the hidden states as an empirical particle cloud. We apply row-center normalization by subtracting the coordinate mean of each hidden vector and projecting it to the unit sphere. We then compare the resulting cloud to samples from a uniform spherical reference using the off-diagonal RBF MMD^2 statistic. We use this statistic descriptively, without an i.i.d.-based unbiasedness claim for the dependent hidden states. Row centering followed by normalization places each nonzero centered vector in $\mathbf{1}^{\perp} \cap S^{d-1}$, a sphere of dimension $d-2$; a uniform reference on the ambient sphere has different support. An exact uniformity interpretation would require a reference matched to this subspace. Accordingly, the reported curves are relative finite-cloud diagnostics, not evidence of equality with an ambient uniform distribution. The curves report the mean over trials, with shaded bands denoting 95% confidence intervals.

Across several large language models, the curves show a U-shaped profile: the measured discrepancy to the chosen spherical reference decreases in middle layers and increases in later layers. In Figure 5, we further measure the corresponding EFS-style logarithmic interaction energy. For this metric, the normalized hidden states are rescaled to radius 0.78, and the energy is computed directly on the layerwise particle cloud. The energy curves support the same interpretation as the MMD plots: transformer layers first regularize the in-context distribution toward a uniform-like configuration and later recover structured semantic geometry.

E.4 Failure mode

The layerwise uniformization pattern is not equally strong across all models. This effect is visible not only within the Qwen2.5 family (Qwen et al., 2024), but also within the GPT-2 family (Radford et al., 2019), and it appears in both the semantic-topic setting and the CBT story setting; see Figure 9, Figure 10, Figure 6, and Figure 11. Figure 6 shows this effect within the Qwen2.5 model family (Qwen et al., 2024). Smaller models, especially the 1.5B, 3B, and 7B variants, show only a short or weak movement toward the uniform spherical reference before the MMD^2 distance increases again. In these models, the middle-layer uniformization phase is compressed, suggesting that the model may not have enough depth to fully realize the two-stage sampler-like computation. The larger 14B and 32B models show a clearer separation between the early movement toward uniformity and the later movement away from it.



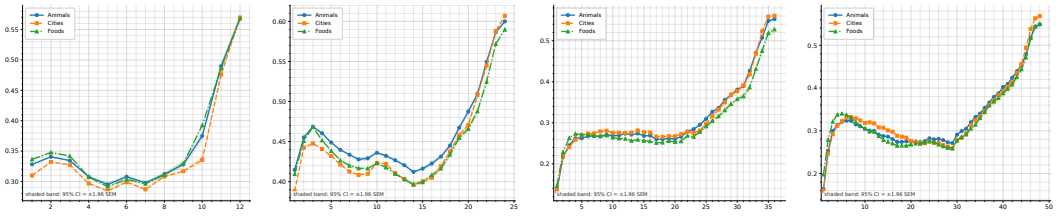
(a) Llama-3.3-70B-Instruct (Grattafiori et al., 2024) (b) GPT-2 XL (Radford et al., 2019) (c) Qwen2.5-14B (Bai et al., 2023a)

(d) Qwen2.5-32B (Bai et al., 2023a) (e) Falcon-40B (Almazrouei et al., 2023) (f) OPT-66B (Zhang et al., 2022)

(g) Bloom-7b1 (Workshop et al., 2022) (h) Cerebras-GPT-13B (Dey et al., 2023) (i) Openlm-Llama-13b (Geng & Liu, 2023)

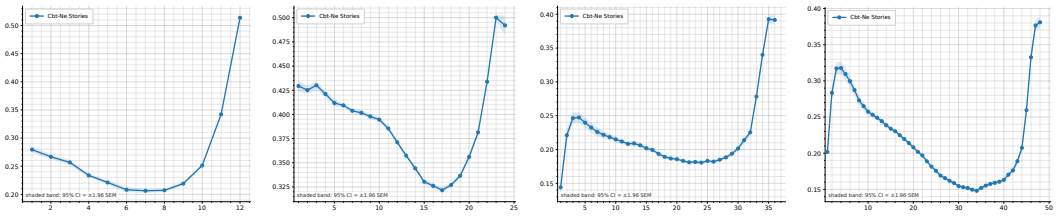
Figure 8: Additional large-model results for semantic-category prompts sampled without replacement. The curves show the layerwise MMD^2 distance to the uniform spherical reference distribution.

This behavior is consistent with the theoretical construction. In Theorem 1, the transformer simulates an iterative sampler by assigning transformer blocks to sampler steps. Therefore, depth is not only an architectural detail; it controls how many test-time computational steps the model can perform. If the model has too few layers, the transport process may be incomplete: the representation starts moving toward a reference geometry, but the model does not have enough intermediate computation to form a stable uniform-like phase before reconstructing semantic structure. We therefore interpret shallow or smaller models as a failure mode of in-context sampling.



(a) GPT-2 (b) GPT-2 Medium (c) GPT-2 Large (d) GPT-2 XL

Figure 9: GPT-2 family (Radford et al., 2019) on semantic-category prompts sampled without replacement. Across model sizes, the layerwise distance to the uniform spherical reference distribution decreases in the middle layers and increases near the output layers.



(a) GPT-2 (b) GPT-2 Medium (c) GPT-2 Large (d) GPT-2 XL

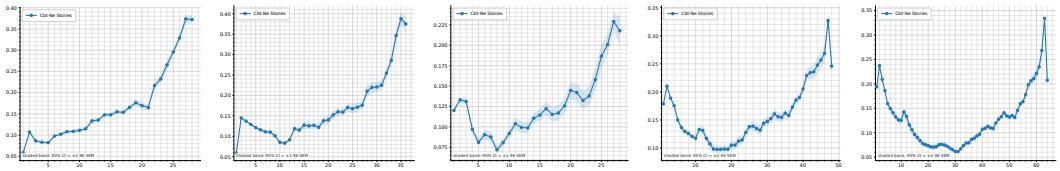
Figure 10: GPT-2 family (Radford et al., 2019) on CBT story-token prompts. The same middle-layer movement toward a uniform spherical reference distribution appears for natural text prompts.

E.5 Computational Resources

All experiments were run on a single NVIDIA H100 GPU machine with 128 GB of system memory. The controlled synthetic experiments in Appendix D.1 and D.2 are computationally lightweight: they use a small GPT-2-style decoder-only transformer. If training is mentioned, the model is trained for 2,000 optimization steps on 1,000 IID training sequences. For the pretrained-language-model analyses in Appendix D.3, no model fine-tuning is performed. The experiments require only forward passes through publicly available autoregressive language models in order to extract hidden states after each transformer block.

E.6 Existing Assets and Licenses

We use publicly available pretrained language models only for evaluation and do not redistribute their weights. For each model family, we cite the original paper, technical report, or official release, and follow the license or model-card terms associated with the official release. The evaluated models include GPT-2 (Radford et al., 2019), Llama/Llama-3.3 (Touvron et al., 2023a; Grattafiori et al., 2024), OpenLLaMA/OpenLM-Llama, Cerebras-GPT (Dey et al., 2023), Qwen2.5 (Qwen et al., 2024), Falcon (Almazrouei et al., 2023), OPT (Zhang et al., 2022), and BLOOM (Workshop et al., 2022). The CBT dataset is cited as (Hill et al., 2015). License and access terms are taken from the corresponding official repositories or model cards.



(a) 1.5B (b) 3B (c) 7B (d) 14B (e) 32B

Figure 11: Qwen2.5 family (Qwen et al., 2024) on CBT story-token prompts. The U-shaped profile becomes visible across model scales, indicating a middle-layer movement toward a uniform spherical reference distribution followed by a later movement away from it.

Table 1: Existing pretrained models and datasets used in the experiments. We use these assets only for evaluation and do not redistribute model weights or dataset copies. License information is taken from the corresponding official release pages or model cards.

Asset	**Use in this paper**	**Source / citation**	**License or terms**
GPT-2	Pretrained autoregressive language model for layerwise hidden-state analysis	OpenAI release; cited as (Radford et al., 2019)	MIT / Modified MIT license, according to the official model release or model card.
Llama / Llama-3.3-70B-Instruct	Pretrained autoregressive language model for semantic-topic and CBT experiments	Meta Llama releases; cited as (Touvron et al., 2023a; Grattafiori et al., 2024)	Meta Llama community/model license; access and use governed by the official Llama license terms.
OpenLLaMA / OpenLM-Llama-13B	Pretrained autoregressive language model for semantic-topic experiments	OpenLLaMA/OpenLM Research official release or model card; cited as (Geng & Liu, 2023)	Apache-2.0 or permissive open-source license, according to the official model card.
Cerebras-GPT-13B	Pretrained autoregressive language model for semantic-topic and CBT experiments	Cerebras-GPT release; cited as (Dey et al., 2023)	Apache-2.0 license.
Qwen2.5 family	Pretrained autoregressive language models for model-scale comparison	Qwen release; cited as (Qwen et al., 2024)	Apache-2.0 for most open-source Qwen2.5 variants; some variants, including 3B and 72B, use model-specific Qwen license terms.
Falcon-40B	Pretrained autoregressive language model for additional model-family experiments	Falcon release; cited as (Almazrouei et al., 2023)	Apache-2.0 license, according to the official release/model card.
OPT-66B	Pretrained autoregressive language model for additional model-family experiments	OPT release; cited as (Zhang et al., 2022)	Meta OPT license terms, according to the official model card or license file.
BLOOM-7B1	Pretrained autoregressive language model for additional model-family experiments	BigScience BLOOM release; cited as (Workshop et al., 2022)	BigScience RAIL License v1.0.
CBT dataset	Natural-text prompts for layerwise MMD experiments	Children's Book Test; cited as (Hill et al., 2015)	Dataset release terms associated with the original CBT release; the dataset is used only for evaluation.