
Energy-Based Transformers are Scalable Learners and Thinkers

Alexi Gladstone^{1,2}, Ganesh Nanduru¹, Md Mofijul Islam^{1,3}, Peixuan Han²,
Hyeonjeong Ha², Aman Chadha^{3,4}, Yilun Du⁵, Heng Ji³, Jundong Li¹, Tariq Iqbal¹

¹UVA ²UIUC ³Amazon GenAI[†] ⁴Stanford University ⁵Harvard University

 [energy-based-transformers.github.io](https://github.com/energy-based-transformers)  github.com/alexiglad/EBT

Abstract

Inference-time computation techniques, analogous to human System 2 Thinking, have recently become popular for improving model performances. However, most existing approaches suffer from several limitations: they are modality-specific (e.g., working only in text), problem-specific (e.g., verifiable domains like math and coding), or require additional supervision/training on top of unsupervised pretraining (e.g., verifiers or verifiable rewards). In this paper, we ask the question “*Is it possible to **generalize** these System 2 Thinking approaches, and develop models that learn to think solely from unsupervised learning?*” Interestingly, we find the answer is **yes**, by learning to explicitly **verify** the compatibility between inputs and candidate-predictions, and then re-framing prediction problems as optimization with respect to this verifier. Specifically, we train **Energy-Based Transformers (EBTs)**—a new class of Energy-Based Models (EBMs)—to assign an **energy** (un-normalized probability) value to every input and candidate-prediction pair, enabling predictions through gradient descent-based energy minimization until convergence. This formulation enables System 2 Thinking to emerge from unsupervised learning, making it modality and problem agnostic. Across both discrete (text) and continuous (visual) modalities, we find EBTs scale faster than the dominant Transformer++ approach during training, achieving an up to 35% higher scaling rate with respect to data, batch size, parameters, FLOPs, and depth. During inference, EBTs improve performance with System 2 Thinking (i.e., extra computation) by 29% more than the Transformer++ on language tasks, and EBTs outperform Diffusion Transformers on image denoising while using fewer forward passes. Further, we find that System 2 Thinking with EBTs yields larger performance improvements on data that is farther out-of-distribution, and that EBTs achieve better results than existing models on most downstream tasks given the same or worse pretraining performance, suggesting that EBTs generalize better than existing approaches. Consequently, EBTs are a promising new paradigm for scaling both the **learning** and **thinking** capabilities of models.

1 Introduction

In psychology, human thinking is often classified into two different types: System 1 (thinking fast) and System 2 (thinking slow) [1–4]. System 1 thinking is characterized by quick, intuitive and automatic responses, relying on previous experience to solve simple or familiar problems. Alternatively, System 2 Thinking is slow, deliberate and analytical, requiring conscious effort and logical reasoning to process more complex information. System 2 Thinking is essential for complex problems that go

Correspondence to Alexi Gladstone:  alexigladstone@gmail.com. [†]Work does not relate to position at Amazon.

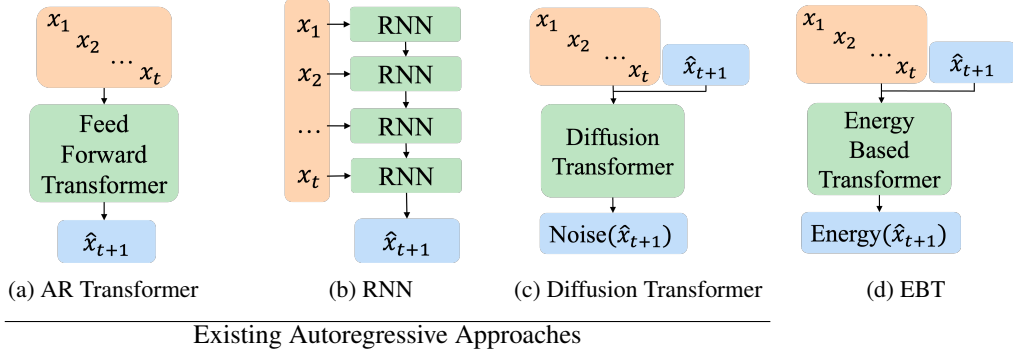


Figure 1: **Autoregressive Architecture Comparison.** (a) Autoregressive (AR) Transformer is the most common, with (b) RNNs becoming more popular recently [22, 23]. (c) Diffusion Transformers [26] (which are often bidirectional but can also be autoregressive) are the most similar to EBT, being able to dynamically allocate computation during inference, but predict the noise rather than the energy [27, 28]. Consequently, diffusion models cannot give unnormalized likelihood estimates at each step of the thinking process, and are not trained as explicit verifiers, unlike EBTs.

beyond automatic pattern recognition, such as in mathematics, programming, multistep reasoning, or novel out-of-distribution situations [5, 6], where precision and depth of understanding are important. Although current models perform well on tasks suitable for System 1 thinking [7], they continue to struggle with tasks that demand System 2 capabilities [8–10].

Consequently, the recent pursuance of System 2 Thinking capabilities has become a growing interest among AI researchers, leading to the rise of several foundation models such as O1 [11], R1 [12], Grok3 [13], and Claude 3.7 Sonnet [14]. These “reasoning models” excel on math and coding benchmarks, particularly by increasing the time models spend thinking. However, publicly available information on training methods, particularly from the open-source R1 model [12], suggests that the Reinforcement Learning (RL) based training approach for these models only works in domains where rule-based rewards can easily verify answers, such as math and coding. This limitation reduces applicability to a small range of problem types, and as a consequence, often deteriorates performance in other tasks such as writing [15–17]. Further, recent evidence suggests this RL-based approach may not induce new reasoning patterns, but rather just increase the probability of reasoning patterns already known to the base model [18], which limits performance on problems requiring exploration.

Along similar lines, there has been strong interest in achieving System 2 Thinking in both diffusion models and Recurrent Neural Networks (RNNs). Diffusion models support iterative inference through denoising steps, where increasing denoising steps can improve performance. However, they typically fail to benefit from denoising steps beyond what they were trained on [19], and aside from increasing denoising steps, diffusion models require an external verifier to improve System 2 Thinking capabilities [19–21]. RNNs also offer iterative computation via recurrent state updates [22–24], however, most modern RNNs only update their internal state with new information, meaning they cannot be used for thinking longer. Additionally, RNNs that do support recurrent depth still lack mechanisms for explicit verification [25], resulting in limited adoption for System 2 Thinking.

As one of the primary goals of AI is to figure out how we can create systems that **learn to think on their own** on any type of problem, these approaches ultimately bring about the following core research question: “*Can we **rely entirely on unsupervised learning** to develop **System 2 Thinking**?*” Such a capability would enable *generalization* of current System 2 Thinking approaches to *any problem, any modality, and avoid the reliance on external human, reward, or model supervision*.

In this work, we argue and demonstrate empirically that the answer to this core research question is **yes**, but that there are several limitations in existing model architectures that prevent this type of unsupervised System 2 Thinking from emerging. Particularly, when comparing the qualities of human System 2 Thinking with the current modeling paradigms (Figure 1, Table 1), we observe several key differences, outlined below as three key Facets of System 2 Thinking¹:

¹We acknowledge that these are not comprehensive for achieving System 2 Thinking, but rather, a good first step (more info is in Section C.1).

Table 1: **Architectures and Cognitive Facets.** For each prediction, Feed Forward (FF) Transformers and RNNs generally² have a finite amount of computation. While diffusion models have potentially more computation during inference by increasing the number of denoising steps, they do not learn to explicitly verify or estimate uncertainty for predictions. EBMs can use a dynamic amount of computation during inference by iterating for any number of steps, and give an energy scalar that can be used to evaluate uncertainty and verify the strength of predictions.

Architecture	Dynamic Compute Allocation (Facet 1)	Modeling Uncertainty (Facet 2)	Prediction Verification (Facet 3)
FF Transformers	✗	✗	✗
RNNs	✗	✗	✗
Diffusion Transformers	✓	✗	✗
EBTs	✓	✓	✓

Facet 1: Dynamic Allocation of Computation. Humans naturally allocate varying amounts of effort to different tasks depending on difficulty, which is widely supported by psychology and neuroscience [1, 31, 32]. As the difficulty of tasks humans face varies widely, the ability to adjust the magnitude of computational resources allocated towards a task is fundamental to success.³ For example, a decision regarding whether to change careers generally takes people much more time than deciding what to eat for lunch.

Facet 2: Modeling Uncertainty in Continuous State Spaces. While thinking longer is important for improving performance, humans also weigh how uncertain they are before committing to a decision. In language, LLMs can simulate this through token-level probabilities [33]. In the context of continuous state spaces, such as in vision, without the usage of discretization schemes such as Vector Quantization [34] or pseudo losses/objectives (such as ELBO [35]), standard implementations of the most successfully used approaches with Transformers, RNNs, or Diffusion models generally do not provide strong or reliable uncertainty estimates [36–39].⁴ EBMs can naturally model uncertainty without having to model exact likelihoods by modeling the relative unnormalized likelihoods of predictions [42], as demonstrated in Figure 3. As the real world often contains many inherently unpredictable elements, for instance, when a pedestrian might emerge from behind a parked vehicle, the ability to express uncertainty in predictions is essential to being cautious, and is a natural capability of humans [43–45].

Facet 3: Verification of Predictions. In addition to allocating computation and modeling uncertainty, effective thinking also benefits from the ability to verify predictions, which can guide decisions about when to stop thinking or to select the most accurate predictions. There is strong evidence that such verification is a core component of human thinking [46, 47]. In fact, it can be shown that verifying solutions is exponentially easier than generating solutions [48], which means that verifiers can often generalize better than explicit generators [48]. Training an explicit verifier means that at each step of the thinking process an estimate of the current prediction’s quality can be extracted. This supports more dynamic inference time behavior such as early stopping when a prediction is known to be correct or allocating more compute when a problem is difficult, which can intuitively be seen as adjusting resources according to the difficulty of a problem [1]. Training an explicit verifier also allows for capabilities such as Monte Carlo Tree Search [49] or sampling many times and choosing the best prediction, which traditionally have involved training new models on more data [19, 50].

For more information on additional Facets, please refer to Section F.

²Recent works attempt to enable dynamic computation per prediction [25, 29, 30], but these approaches are generally not modality agnostic and have not been widely adopted. While RNNs can theoretically support dynamic computation, they are typically updated only with new state information, except in specialized cases such as in [25].

³It’s important to note that we refer to this dynamic computation capability at the granularity of **each** prediction being made, meaning current LLMs built with traditional AR transformers/RNNs cannot dynamically allocate compute per token generated as they have a finite depth/width and are only updated with new text tokens. Please check Section G for more information.

⁴We acknowledge that there are approaches to achieve uncertainty with the models discussed, such as Mixture Density Networks [40] as well as score-based diffusion models [41]. However, these approaches have seen less widespread success and scalability than the current dominant approaches.

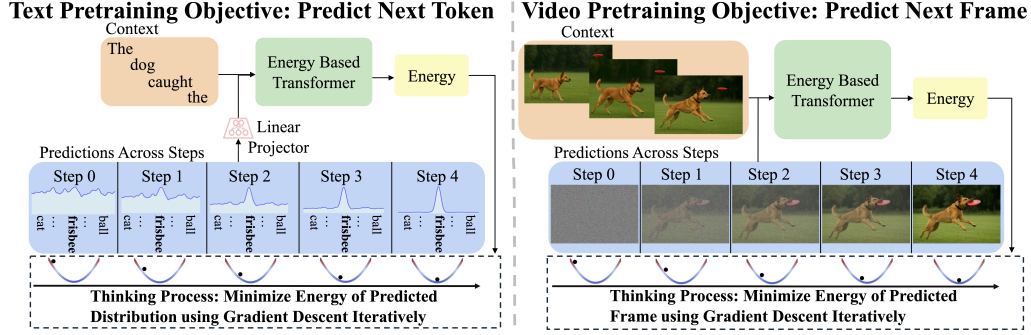


Figure 2: **EBT for Autoregressive Modeling.** Each blue box corresponds to a different prediction based on the current step of the thinking process, where the initial prediction starts as random. At each step, a new prediction is fed into the model, which gives an energy scalar for the prediction’s current *compatibility* (unnormalized likelihood) with the context (Facets 2 and 3). Then, the gradient of this energy with respect to the prediction is calculated and used to update the prediction. This gradient descent update is done iteratively to refine the prediction until convergence of the predicted energy, which allows for dynamic use of computation (Facet 1).

To achieve all facets described, we propose viewing thinking as an optimization procedure with respect to a learned verifier, which evaluates the compatibility (unnormalized probability) between an input and candidate prediction (Figure 2). More concretely, we train Energy-Based Models (EBMs) to learn an energy (unnormalized probability) landscape over all possible input-prediction pairs, where lower energy indicates higher compatibility. Then, thinking corresponds to starting from an initial random prediction, and progressively refining it through energy minimization until convergence (visualized in Figure 3). Optimizing over a learned energy landscape naturally allows for dynamic compute allocation (Facet 1), allowing models to think for longer on harder problems by iteratively performing more steps. Additionally, at each step of this thinking process, EBMs act as verifiers in the forward pass, giving an energy scalar which represents the *compatibility* of the context with the prediction. This energy scalar directly addresses Facets 2 and 3 by serving as an unnormalized likelihood as well as the score that verifies whether a prediction is correct (Figure 3).

While viewing thinking through the lens of inference-time energy minimization is a promising perspective, EBMs have traditionally struggled with scalability [51], with zero publicly known Foundation EBMs. This can be attributed to issues with EBM training stability [51–54] and long training times [53, 54]. To address these challenges and establish a scalable EBM training paradigm, we introduce Energy-Based Transformers (EBTs), Transformer implementations specifically for EBMs. We release two variants: a decoder-only EBT inspired by the GPT architecture [55], which parallelizes all predictions simultaneously; and a bidirectional EBT, enabling bidirectional attention across entire sequences, similar to BERT [56] and Diffusion Transformers (DiT) [26].

Additionally, we identify practical enhancements that improve the training efficiency of EBTs and provide theoretical insights into why our optimization-based training approach achieves strong scalability. Finally, we introduce *energy landscape regularization* techniques, which are methods that enhance the convexity and smoothness of learned energy landscapes, thereby fostering the emergence of strong System 2 Thinking capabilities during pretraining.

To investigate the learning and thinking scalability of EBTs, we compare EBTs to the Transformer++ for autoregressive modeling and the Diffusion Transformer (DiT) for bidirectional modeling, across both discrete and continuous modalities. During pretraining, we find that EBTs achieve an up to 35% higher scaling rate than the Transformer++ across several axes, including data, batch size, parameters, FLOPs, and depth. Notably, EBTs are the first approach to out-scale the standard feed-forward Transformer++ recipe across several modalities and axes, including improved data efficiency. At inference, EBTs outperform existing paradigms on System 2 Thinking, or solving more challenging problems with additional computation. For example, EBTs can improve language modeling performance 29% more than the Transformer++, and for image denoising EBTs exhibit a higher performance-compute scaling rate and improved performance over DiTs with 99% fewer forward passes. In investigations of the effect of System 2 Thinking on generalization, we consistently observe two phenomena. First, with the same or worse pretraining performance, EBTs still outperform

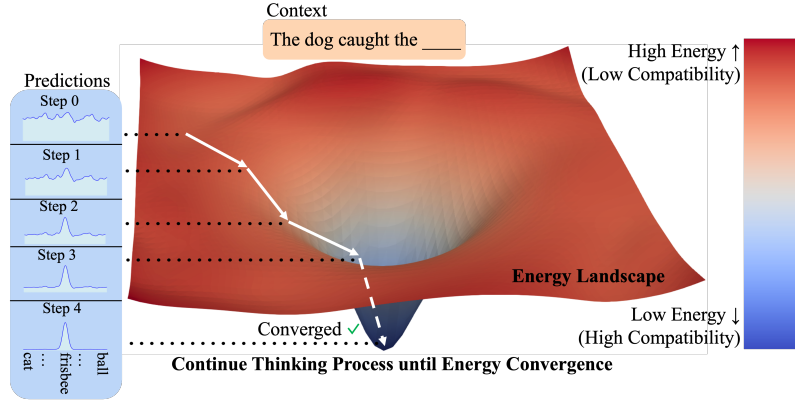


Figure 3: **Thinking Process Visualization.** A learned energy landscape and its optimization through gradient descent, interpreted as a thinking process. In this example, the model predicts a distribution over text tokens, progressively shifting from an initial random distribution toward the target distribution. At each step, the EBM assigns an energy scalar indicating how **compatible** the current prediction is with the context, visualized as the landscape’s height (Facet 3). This scalar’s convergence allows the model to determine whether the prediction is adequate or if further thinking is necessary. Uncertainty (Facet 2) can be represented by landscapes that are harder to optimize or by landscapes with many local minima, allowing the model to know when it requires more steps to think (Facet 1). Adapted from [57].

existing models at inference by performing System 2 Thinking, demonstrating its importance and how EBTs often generalize better than existing models. Second, System 2 Thinking with EBTs yields more substantial performance gains on data that is further out-of-distribution, aligning with observations in psychology where humans use System 2 Thinking on challenging, unseen tasks.

We believe the EBT implementations, along with novel techniques for EBMs to maximize the learning and thinking scalability, will advance the EBM paradigm by addressing key challenges in stable, parallelizable, and efficient training.

2 Energy-Based Transformers (EBT) Intuition

Energy-Based Models (EBMs) learn an energy function that assigns a scalar value to each input configuration, with lower energy indicating higher compatibility or likelihood between input variables [51], and high energy indicating lower compatibility or likelihood. Accordingly, the energy function acts as a **verifier** of input data coherence. Leveraging this principle, Energy-Based Transformers (EBTs) are trained to **verify** the compatibility of a given context-prediction pair (give an energy value), and then make predictions by optimizing with respect to this verifier (minimizing the energy), which is shown in Figure 2. Starting from an initial prediction, such as random noise, EBTs iteratively refine their output by progressively minimizing the energy, ultimately converging to predictions that are consistent with the given context. Performing energy minimization through this process simulates the thinking process during pretraining, unlike with traditional models, enabling each prediction (e.g., a token for LLMs) to have its own thinking process.

2.1 Learning to Verify

Verifying solutions is often substantially more tractable than generating them, a distinction well-established in complexity theory and foundational to developments in knowledge proofs and learning algorithms [58–60]. For example, in solving a maze, verifying the correctness of a given path is significantly easier than discovering such a path. This asymmetry has been recognized and utilized for several decades, notably in the field of cryptography [59, 61, 62]. EBMs are built on this principle that *verification is easier than generation*: rather than learning to generate directly, as in most existing paradigms, EBMs learn to generate by optimizing predictions with respect to this learned verification (energy function); this is depicted in Figure 3.

Recent works have attempted to leverage this characteristic of verifiers [19, 63–65], but these approaches decouple the verifier and generator, resulting in adversarial dynamics [19] and challenges in scalability [64]. For example, researchers combining tree search and LLMs required thousands or even millions of samples to achieve optimal performance [63]. In contrast, EBMs **combine** the verifier and generator into a **single model**, where the generator is defined implicitly by the gradient of the verifier [51]. We show that this coupling resolves scalability and adversarial issues (Figures 6b and B.1a).

An additional advantage of verifiers is generalization. Because verification is usually easier than generation [66], *prediction verification on Out-Of-Distribution (OOD) data is often easier than explicit prediction generation for OOD data* [48]. This characteristic often results in better generalization of verifiers than explicit generators [48]. As an example, models trained to explicitly solve mazes on a small grid may not generalize to larger grids, whereas verifiers for maze solutions learn whether a path is correct, which more easily transfers to larger mazes. This characteristic may be why EBMs often generalize better than existing models [48, 67], which we further support in our larger-scale experiments (Figure 6a and Table 4).

2.2 Learning to Understand

This verifier-centric perspective also relates to a deeper limitation in contemporary generative models, referred to as “The Generative AI Paradox [68].” Although current generative models achieve strong generative capabilities, they frequently lack basic discrimination skills, such as the ability to assess the plausibility or coherence of their own predictions [68, 69]. These limitations impede their ability to engage in reasoning, planning, and decision-making [9, 70]. In contrast, EBMs offer a potential solution to this challenge: as EBMs generate by learning a verifier (which is conceptually similar to a discriminator), they develop strong discrimination skills [71]. Experimental results further support this observation (Table 4).

3 Energy-Based Transformers (EBT) Approach

3.1 Energy-Based Models (EBM) Background

Energy-Based Models (EBMs) assign a scalar energy value to each configuration of input variables, enabling them to model the **compatibility** and **interactions** between variables, such as between a context and candidate-prediction. In the case of probabilistic EBMs, this defines a probability distribution using a Boltzmann distribution $p_\theta(x) = \frac{e^{-E_\theta(x)}}{Z(\theta)}$ where $Z(\theta) = \int e^{-E_\theta(x)} dx$ is the intractable partition function involving an integral over all possible values of x . Due to the negative exponential, lower energy corresponds to higher probability, and higher energy corresponds to lower probability. To avoid the intractability of the partition function, it is common to work with **unnormalized EBMs**, which dispense of the partition function in favor of representing relative unnormalized probabilities. This formulation shifts the focus from addressing the partition function, to simply assigning low energy to the true data manifold and high energy elsewhere [42, 51], offering benefits such as scalability to spaces where the true data manifold is thin and therefore a probabilistic EBMs would have an infinite score [42]. In supervised or predictive self-supervised learning (e.g., classification, autoregressive modeling, masked modeling), unnormalized EBMs can be formulated as: $p_\theta(x, \hat{y}) \propto e^{-E_\theta(x, \hat{y})}$, where the goal of the EBM is to learn to predict $\hat{y}$ given x .⁵ For an accessible and beginner-friendly introduction to EBMs, including intuitive explanations and pseudocode, please refer to Section H.

3.2 Scalable EBM Learning

While EBMs offer a flexible modeling framework, training them scalably remains an open research problem. Two primary training approaches exist—contrastive and regularized methods [72]. Contrastive methods increase the energy of negative samples while decreasing the energy of positive samples. Due to the curse of dimensionality, where the volume of spaces grows exponentially with

⁵In the case of self-supervised learning, x is some unmasked portion of the original x and $\hat{y}$ is the masked portion.

Algorithm 1: Training

Inputs: Context x , Target y , EBM $E_\theta(x, \hat{y})$ **Hparams:** Steps N , Step Size α , Loss $J(\cdot)$

```

1 Sample  $\hat{y}_0 \sim \mathcal{N}(0, I)$ ;
2 for  $i = 0, \dots, N - 1$  do
3    $\hat{y}_{i+1} \leftarrow \hat{y}_i - \alpha \nabla_{\hat{y}_i} E_\theta(x, \hat{y}_i)$ ;
4  $\mathcal{L} \leftarrow J(\hat{y}_N, y)$ ;
5 return  $\mathcal{L}$ , update  $E_\theta$ ;

```

Algorithm 2: Inference with Verification

Inputs: Context x , EBM $E_\theta(x, \hat{y})$ **Hparams:** Steps N , Step Size α , Samples M

```

1 for  $j = 1, \dots, M$  do
2   Sample  $\hat{y}_{0,j} \sim \mathcal{N}(0, I)$ ;
3   for  $i = 0, \dots, N - 1$  do
4      $\hat{y}_{i+1,j} \leftarrow \hat{y}_{i,j} - \alpha \nabla_{\hat{y}_{i,j}} E_\theta(x, \hat{y}_{i,j})$ ;
5 return  $\hat{y}^* = \operatorname{argmin}_j E_\theta(x, \hat{y}_{N,j})$ ;

```

their dimension, contrastive methods struggle to scale because they must increase the energy of an exponentially higher number of negative samples [42].

An alternative is to frame EBM learning as an optimization problem [48, 71], which avoids the curse of dimensionality by implicitly regularizing the energy landscape, enabling scalable learning. In this approach, EBMs are trained to optimize an initial prediction to the ground truth solution through gradient descent, as shown in Figure 3. This pushes the energy landscape to be convex surrounding the ground truth solution, thereby regularizing the energy landscape to only have low energy on the true data manifold. Intuitively, this training approach is similar to GANs [73]. During the forward pass, EBMs can be seen as a GAN discriminator by giving an energy “verification”; on the backward pass they can be seen as a GAN generator by optimizing predictions through energy minimization to try and fool the discriminator.

Training EBMs to perform optimization can be formalized as follows. We begin with an EBM E_θ , a prior (initial prediction) $\hat{y}_0$, an input (context) for the model x , and seek to predict y . We aim to find the minimum energy (most compatible/likely) $\hat{y}$ given an x , which we search for using gradient descent:

$$\hat{y}_{i+1} = \hat{y}_i - \alpha \nabla_{\hat{y}_i} E_\theta(x, \hat{y}_i), \quad (1)$$

where α is the step size (this is formalized in Algorithm 1). Then, the loss can be computed using any standard objective function. For this work, we use the same loss functions as existing papers to simplify experiments, so we use categorical cross-entropy for language modeling [74] and mean squared error for image denoising [75]. Importantly, this loss is backpropagated through the entire optimization process, requiring second-order derivatives (i.e., gradients of gradients). These are computed efficiently via Hessian-vector products, which scale linearly with model size [76], similar to standard first-order backpropagation in feed-forward models. More details and pseudocode can be found in Section I.2.

3.3 Scalable EBM Thinking

While this training approach is scalable, achieving smooth and convex energy landscapes, such as the landscape visualized in Figure 3, remains challenging on real-world problems. Because y is high-dimensional, the energy landscape spans a high-dimensional space, and must remain well-shaped throughout. To address this, we found three key *energy landscape regularization* techniques to be essential in ensuring the smoothness and convexity of learned energy landscapes, enabling strong thinking capabilities to be learned during training.

First, we found a replay buffer (following existing EBM works [48, 51]) helps simulate longer optimization trajectories, enabling energy landscapes to be well defined near their minimum. Second, a variant of Langevin Dynamics [51] (random noise), was found to be helpful for encouraging **exploration** of the energy landscape:

$$\hat{y}_{i+1} = \hat{y}_i - \alpha \nabla_{\hat{y}_i} E_\theta(x, \hat{y}_i) + \eta_i, \quad \eta_i \sim \mathcal{N}(0, \sigma), \quad (2)$$

where σ is the magnitude of the noise η . Without this random noise term, exploration is often limited to paths leading directly to the energy minimum, leaving other regions poorly defined. Third, varying the paths taken towards predicting solutions, by randomizing the gradient descent step size α and number of optimization steps, significantly improved generalization. Together, these techniques substantially improved the System 2 Thinking capabilities of models, as confirmed by ablation experiments in Table 2.

With these energy landscape regularization techniques established, to understand the System 2 Thinking capabilities of EBTs, we explored two main thinking approaches, corresponding to two of

the cognitive facets described. First, corresponding to dynamic computation allocation (Facet 1), we conduct experiments that involve changing the number of steps taken for optimization of a single prediction. This is conceptually similar to increasing the denoising steps performed with a diffusion model. Second, corresponding to the ability to verify predictions (Facet 3), we generate N predictions from an EBM and then choose the minimum energy prediction. This is conceptually similar to Best of N (BoN) sampling using language models [77]. However, EBMs generalize this approach to both discrete and continuous modalities, don’t require additional supervision (e.g., an external reward/verification model), and perform it on *every single prediction*, not just to entire sequences. We demonstrate performance improvements gained from both of these techniques in several Thinking experiments (Figures 6, 7, and 12), which further confirm the importance of the described cognitive facets. This thinking process is formalized in Algorithm 2 and we more formally define and justify usage of the term thinking in Section C.1.

3.4 Energy-Based Transformers (EBTs) Architecture

Transformers have demonstrated exceptional performance across numerous domains [12, 78–81]. Three primary advantages of Transformers include their parallelizability [82, 83], their stability [84], and their scalability [85]. Because Energy-Based Models (EBMs) have encountered difficulties with all three of these characteristics [51–54], Transformers represent a particularly suitable architecture for scaling EBMs. Consequently, to advance the EBM paradigm, we introduced Energy-Based Transformers (EBTs), which are Transformer implementations designed for EBMs. We developed two variants: a decoder-only EBT, inspired by the GPT architecture [55] for autoregressive modeling, as well as a bidirectional EBT with bidirectional attention across sequences, enabling capabilities such as infilling and masked modeling [26, 56]. Although the bidirectional EBT implementation is relatively straightforward, the autoregressive EBT presents greater implementation challenges, primarily due to the potential for information leakage in naïve implementations. Comprehensive details of this implementation are discussed in Section C.3.

4 Experimentation and Results

We experiment with EBT across both Autoregressive (AR) [82] as well as Bidirectional models [56] in discrete and continuous spaces.⁶ In discrete spaces, we focus primarily on the language modeling objective. In continuous spaces, we focus on vision tasks of next frame prediction and image denoising. We perform the most comprehensive experiments using Autoregressive Language Models trained to predict the next token, as this is the most extensively studied pretraining approach [86]. All models are pretrained from scratch under a tightly constrained compute budget, as the architecture of EBTs is incompatible with existing foundation models, making them incompatible for adaptation via fine-tuning. We focus on two primary types of results. First, we examine **learning scalability**, investigating how quickly models can fit the pretraining data, which is standard in the pretraining literature [22, 85–90].

Second, we study **thinking scalability**, or how the performance of models changes as we scale the System 2 Thinking of models (Definition C.1). To measure the thinking scalability, we use the Number of Function Evaluations (NFEs) [19, 91], which we deem to be one for every forward pass completed (for EBMs this is one function evaluation per optimization step). By scaling the number of forward passes during inference, we can determine whether model performance improves with increased thinking.

4.1 Autoregressive Language Modeling Experiments

In this section, we detail and discuss the results for all Natural Language Processing (NLP) experiments using Autoregressive (AR) Language Models trained to predict the next discrete token in a text sequence [82]. All language models are pretrained on the RedPajamaV2 text corpus [92, 93] 100B sample from HuggingFace using the GPT-NeoX tokenizer [94] (following [22]) to predict the next token. Following existing pretraining work, we compare AR EBT with the standard Transformer++ recipe [22, 87, 95]. We manually created a training and validation split of 66 million and 33 thousand samples, respectively.

⁶Here, Autoregressive and Bidirectional refer to the procedure for generation. It’s worth noting that autoregressive models are compatible with bidirectional attention, as in [27].

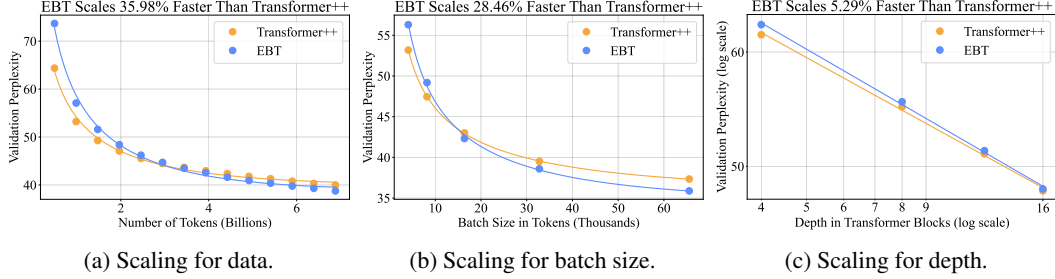


Figure 4: **Language Learning Scalability—Data, Batch Size, and Depth.** A comparison between the scaling of the Transformer++ recipe [87] and EBTs across data, batch size, and depth during pretraining. On all of these axes, EBTs out-scale the Transformer++ recipe significantly, indicating improved data efficiency and suggesting potential benefits for generalization. Additionally, the improved depth scaling offers promise for reasoning, where depth is crucial [96]. These results suggest that if these scaling trends persist, EBTs would likely outperform Transformer++ models at foundation model data scale.

For downstream evaluation, we used four key datasets in addition to the pretraining dataset, spanning reasoning, question answering, and syntax understanding. Ordered roughly by increasing perplexity difficulty, these include GSM8K [97], SQuAD [98], BigBench Elementary Math QA [99], and BigBench Dyck Languages [99]. We intentionally design the evaluation towards reasoning benchmarks due to their alignment with System 2 Thinking. Additionally, we focus on reporting perplexity as our relatively small models trained from scratch, when compared to current foundation models, do not achieve high accuracies on many of these benchmarks. Furthermore, perplexity often functions as a more linear metric than accuracy [22, 100], enabling a more comparable analysis of downstream performance as we scale compute during inference. Further details on the exact hyperparameters and setup used for all experiments are in Section D.

4.1.1 Autoregressive Language Model Learning Scalability

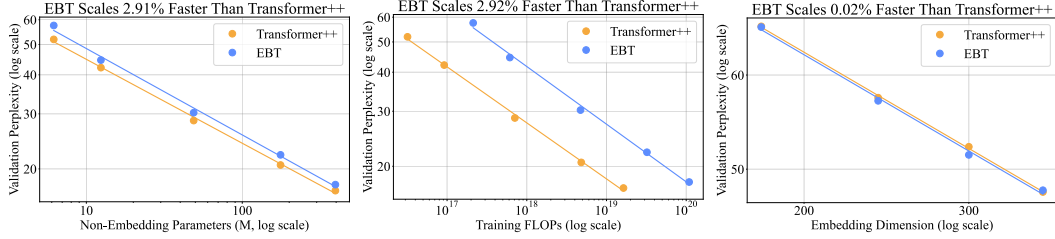
The scaling trends related to the learning speed of models, commonly referred to as “scaling laws,” are challenging to measure. For example, a recent survey [86] found that the “scaling laws” observed often depend on several implementation details and axes to measure, which can result in several different conclusions.⁷ Therefore, to be as comprehensive as possible in determining how EBTs scales compared to the Transformer++, we conduct scaling experiments for six different axes—including data, batch size, depth, parameters, FLOPs,⁸ and embedding dimension. The results for the data, batch size, and depth scaling are shown in Figure 4; and the results for parameters, FLOPs, and embedding dimension are visualized in Figure 5. Across all axes EBT consistently out-scales (has a higher scaling rate) the Transformer++ recipe, becoming the first model to achieve such a feat without using a different tokenizer ([101] was the first and only work to our knowledge to out-scale the Transformer++ recipe, but they use a different tokenizer, and do not out-scale the Transformer++ across multiple axes unlike EBTs). These results suggest that EBTs are more data efficient, batch size efficient, parameter efficient, depth efficient, and compute efficient than the Transformer++ recipe. Thus, at the scale of modern foundation models trained on $1,000\times$ more data with models $1,000\times$ larger (following [84]), we expect the pretraining performance of EBTs to be significantly better than that of the Transformer++ recipe.

4.1.2 Autoregressive Language Model Thinking Scalability

Building on the learning results, we investigate EBTs for thinking at inference time. We found that the thinking capabilities of EBT emerge with a sufficiently large data scale, and therefore, due to limited resources, we focus on conducting thinking experiments with smaller models trained on substantial amounts of data. We test thinking capabilities along two axes: thinking longer, which denotes more optimization steps, and self-verification, which denotes generating many candidate predictions and selecting the minimum energy prediction. In Table 2, we conduct ablation studies to confirm the benefits of our *energy landscape regularization* techniques for System 2 Thinking

⁷For more information on this perspective please refer to Section D.1.1

⁸The FLOP calculation is nuanced and depends on specific hyperparameters. For more information please refer to Section D.5.



(a) Scaling for number of Parameters. (b) Scaling for number of FLOPs. (c) Scaling for the embed. dimension.

Figure 5: Language Learning Scalability—Parameters, FLOPs, and Width. Pretraining scaling comparisons between the Transformer++ recipe [87] and EBTs across model size (parameters), compute (FLOPs), and width (embedding dimension). EBTs slightly out-scale the Transformer++ in FLOP and parameter scaling, becoming the first approach to achieve a higher **scaling rate** without modifying the tokenizer [101] to our knowledge. These results suggest that EBTs offer high promise as a pretraining paradigm in both parameter and FLOP efficiency as scale increases.

Table 2: System 2 Thinking Ablations. All energy landscape regularization techniques described in Section 3.3 and their impact on System 2 Thinking performance, measured by percent perplexity improvement. Thinking Longer denotes more optimization steps and Self-Verification denotes optimizing many predictions and choosing the best. Bolded highlights the default System 2 Hyperparameters, leveraging all energy landscape regularization techniques described. This configuration results in the best performance when thinking longer and doing self-verification. Removing regularization, such as Langevin Dynamics, results in less energy landscape exploration, which improves single path performance (thinking longer) at the expense of self-verification performance.

Model	Thinking Longer $\uparrow$	Thinking Longer and Self-Verification $\uparrow$
No Random Step Size	-1.47	0.19
No Random Num. Steps	0.00	9.65
No Langevin Dynamics	17.2	17.0
No Replay Buffer	14.8	17.8
Full System 2 Configuration	7.19	18.7

on Out-of-Distribution Data from the BigBench Dyck Languages benchmark [99]. We find that using all techniques yields the best System 2 Thinking performance when combining extended thinking and self-verification. Additionally, the results show that randomizing the step size is critical—removing it nearly eliminates Thinking gains, while disabling Langevin Dynamics degrades combined performance but improves results without verification, offering a performance-compute tradeoff.

Having established the importance of these landscape regularization techniques, in Figure 6, we analyze the scalability of thinking with EBTs, where the results yield two main insights. First, as shown in Figure 6a, EBTs are able to improve performance by as much as 29% by increasing the amount of forward passes (thinking time), whereas the Transformer++ cannot improve performance at all.⁹ This aligns with our claims that because traditional feed-forward Transformers cannot dynamically allocate additional computation for each prediction being made, they are unable to improve performance for each token by thinking for longer.

Second, as demonstrated in Figure 6b, the thinking capabilities of EBTs scale, showing that as EBTs are trained for longer, their ability to achieve improvements from verification improves, increasing up to 10%–14% from 4%–8%. This suggests that EBTs trained at the same scale as modern foundation models, such as the 15T tokens Llama3 [84] was trained on ($\approx 1000\times$ the current data scale), would have significantly more substantial results from self-verifying. Lastly, we visualize results from EBT at representing uncertainty while predicting tokens in Figure 8. The results demonstrate that for easier to predict tokens, such as “the” or “but”, EBTs optimize to lower energies faster, whereas for harder to predict tokens, such as “fox” or “problem” EBTs have higher energy that does not converge across

⁹Because we pretrained language models from scratch, and are unable to train models the size of modern foundation models, we find models did not benefit from inference time techniques such as Chain-of-Thought. However, we expect both EBT and Transformer++ models to benefit equally from existing techniques.

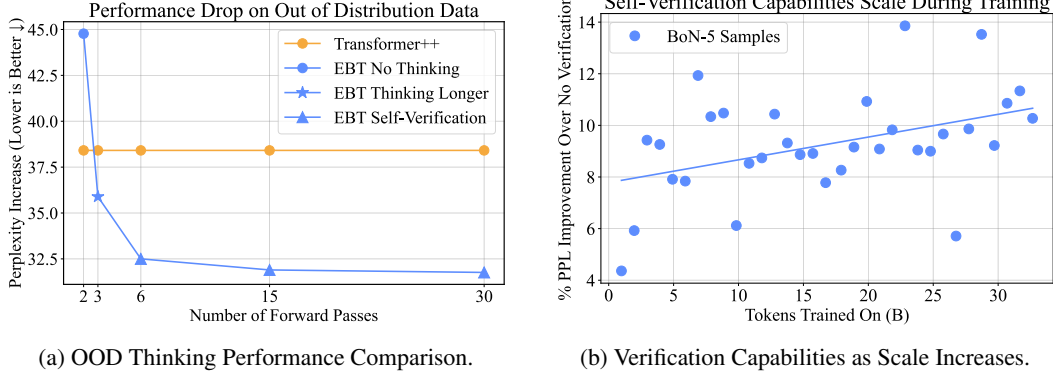


Figure 6: **EBT Thinking Analysis.** (a) Mean performance degradation of the standard Transformer++ recipe [87] and the Energy-Based Transformer (EBT) on four Out of Distribution (OOD) datasets. While the Transformer++ cannot reduce perplexity at a *per-token level*, EBTs can by performing more forward passes over a single token/sample (Thinking Longer) as well as generating many samples and choosing the minimum energy one (Self-Verifying/BoN in addition to longer thought). (b) The Self-Verification capabilities of EBTs using BoN-5 compared to not doing any self-verification. As data scale increases, the benefit from doing self-verification increases. These results suggest EBTs generalize OOD better than the Transformer++ because of their System 2 Thinking capabilities, and that the thinking capabilities of EBTs scale during training.

steps. This suggests that during pretraining EBTs learn to capture uncertainty regarding which tokens are harder or easier to predict, achieving Facet 2.

4.1.3 Autoregressive Language Model Generalization

As System 2 Thinking in humans is associated with generalization to novel scenarios, we conduct experiments directly aimed at measuring the effects of System 2 Thinking on generalization. In Figure 7, we visualize the performance of EBTs on the datasets described, which have varying levels of Out-of-Distribution (OOD) shift (measured as the ratio of downstream task perplexity to pretraining perplexity). We observe a strong linear trend: as the data becomes more OOD, thinking leads to greater performance improvements. Therefore, these findings suggest that the benefits of EBTs’ thinking are not uniform across all data but scale positively with the magnitude of distributional shifts, highlighting thinking as a critical mechanism for robust generalization beyond training distributions. These findings align with observations in psychology where humans rely on deliberate System 2 Thinking to tackle challenging OOD tasks.

Next, we investigate the relation between OOD generalization and pretraining performance. Pretraining performance is strongly correlated with downstream task performance in language models [102–104], and can even be a predictor of downstream performance [84, 105, 106]. Because we know that EBTs scale at a faster rate than the Transformer++, as demonstrated in Figures 4 and 5, it is reasonable to hypothesize that they may also perform better on downstream tasks at scale. To investigate this, we compare models with identical training setups, where EBTs have slightly worse pretraining perplexity than Transformer++ models. As shown in Table 3, despite achieving a higher

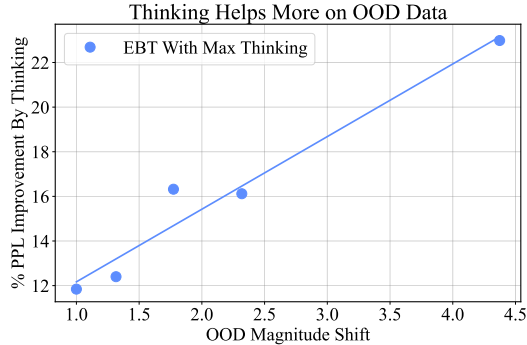


Figure 7: **OOD Thinking Performance.** As the data becomes more OOD, thinking leads to greater performance improvements, with a roughly linear trend. These findings highlight that EBTs thinking is especially critical for robust generalization to OOD data. Performance is measured on 5 different datasets varying in Out-of-Distribution (OOD) magnitude shift, which is measured as the ratio of downstream dataset perplexity to pretraining perplexity. Max Thinking denotes combining thinking longer and self-verification.

Table 3: **Language Model Task Generalization Comparison.** Despite exhibiting a slightly higher pretraining perplexity, EBTs usually achieve lower perplexity on downstream tasks than the Transformer++. This suggests that EBTs generalize better than the Transformer++. Additionally, because EBTs scale better than the Transformer++ during pretraining (Figure 4), these findings suggest that EBTs would outperform Transformer++ at foundation model scale. BB stands for BigBench.

Model	Pretrain	GSM8K ↓	SQuAD ↓	BB Math QA ↓	BB Dyck ↓
Transformer++	31.36	49.6	52.3	79.8	131.5
EBT	33.43	43.3	53.1	72.6	125.3

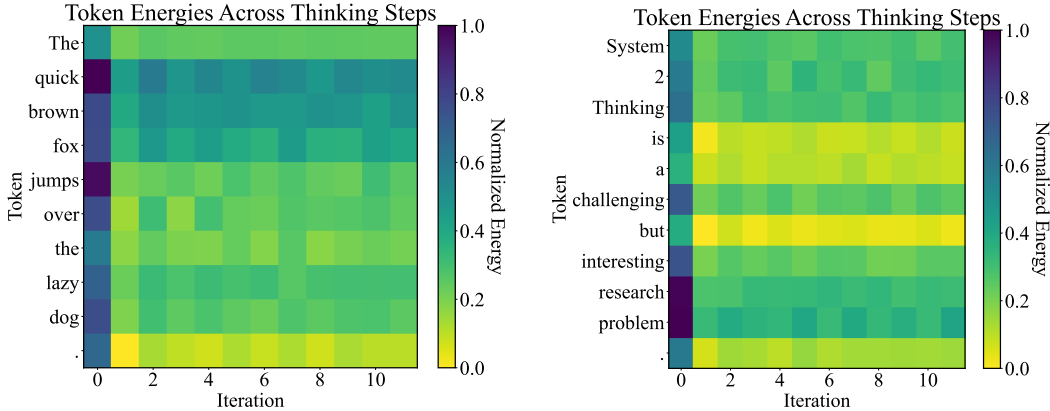


Figure 8: **Learning Uncertainty on Text Results.** EBTs learn to vary uncertainty across text tokens without any explicit supervision. As an example, in both (a) and (b), simple tokens such as “.”, “is”, “a”, “but”, or “the” have lower energies across inference-time optimization (thinking) steps, indicating lower uncertainty. On the other hand, harder to predict tokens such as “quick”, “brown”, “research”, and “problem” have higher energies across optimization steps, and more difficulty in achieving energy convergence, meaning the model is more uncertain. Inspired by [25].

pretraining perplexity, EBTs achieve lower (better) perplexity on most downstream tasks, suggesting stronger generalization, particularly to Out-of-Distribution (OOD) data. Together, with the better learning scalability results, and knowing that improved pretraining performance usually leads to improved downstream task performance [84, 105, 106], these results suggest that at scale EBTs would outperform the Transformer++.

4.2 Autoregressive Video Experiments

To assess how EBTs scale in continuous domains, we train models to predict the next image in a video conditioned on all previous frames—a common pretraining objective for video models [107–111]. Unlike in the NLP experiments, where models see each sample only once due to the dataset size, current popular video datasets are relatively small, requiring models to train repeatedly on the same data. As a result, this setting probes a different question: “*how well can models fit a fixed dataset?*” rather than how efficiently models scale under non-data-bound regimes. This distinction is especially important given the recent scarcity of high-quality datasets [112, 113] and the perspective that data will increasingly become a bottleneck.

For experiments, we encode all 224×224 images into 3136 dimensional features with the frozen SD-XL VAE [114, 115]. Then, all models are trained using a Smooth L1 loss with $\beta = 1.0$ on the Something Something V2 dataset [116], where we report the minimum validation loss achieved. In Figure 9 we report scaling results for the embedding dimension and non-embedding parameter count, as we found these axes behaved the most linearly. The results demonstrate that, despite achieving a higher initial loss, EBTs scale at a more than 33% faster rate than the Transformer++. This suggests that at foundation model scale EBTs would achieve significantly better performance than the Transformer++.

We believe this large scaling rate gap can be linked to the fact that EBTs more seamlessly model continuous distributions than standard feed forward transformers due to being able to express

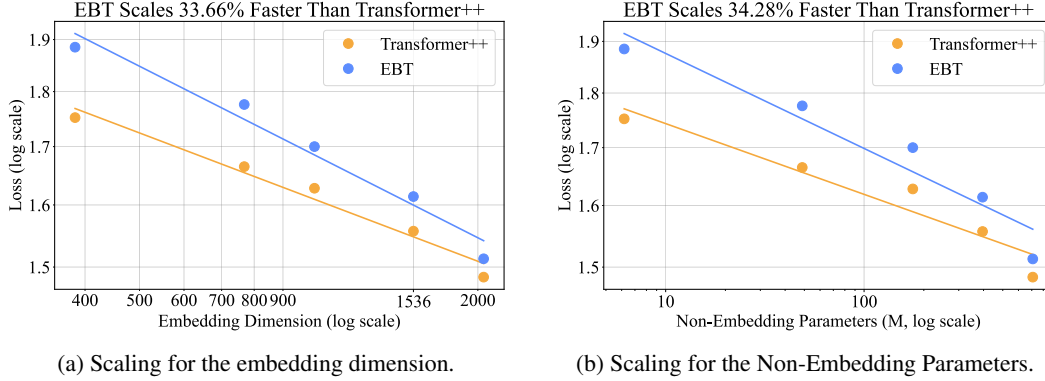


Figure 9: **Video Learning Scalability—Width and Parameters.** The minimum validation loss achieved on the Something Something V2 (SSV2) dataset. While EBTs achieve higher validation loss than the Transformer++ at smaller scales, the scaling rate is more than 33% higher, suggesting that at foundation model scale with hundreds of billions of parameters EBTs would perform much better than the Transformer++. Notably, scaling with respect to the embedding dimension behaves more linearly than for the number of parameters, likely due to the embedding dimension serving as a bottleneck for the image representation.

Table 4: **Image Denoising and Classification Comparison.** For image denoising, EBTs significantly outperform DiTs [26] in Peak Signal to Noise Ratio (PSNR), as well as MSE, on both in-distribution and Out-Of-Distribution (OOD) data, while using 99% less forward passes. This suggests that EBTs generalize better than DiTs while using less computation. On image classification, EBTs also perform better than DiTs, yielding around $10\times$ higher accuracy, suggesting that EBTs learn better image representations and therefore understand images better than DiTs.

Model	In Distribution Noise $\sigma = 0.1$		OOD Noise $\sigma = 0.2$		ImageNet-1k Classification	
	PSNR $\uparrow$	MSE Pixel $\downarrow$	PSNR $\uparrow$	MSE Pixel $\downarrow$	Top 1 Acc. $\uparrow$	Top 5 Acc. $\uparrow$
DiT	26.58	142.98	19.56	718.7	0.31%	1.36%
EBT	27.25	122.55	23.29	305.2	5.32%	13.2%

uncertainty (Facet 2) through their energy scalar. To confirm this, we visualize results for different energies when predicting video frames in Figure 11. The results demonstrate that EBTs successfully learn to capture uncertainty—where frames earlier on in the video have higher energy (higher uncertainty) due to no large objects being within the frame, and then as the major object in the scene becomes revealed more EBT predicts lower energy (lower uncertainty). EBTs learn to exhibit this behavior without any supervision using a Smooth L1 loss, whereas the standard feed-forward Transformer++ would require discretization schemes such as Vector Quantization [117] with a categorical loss, or other tricks to achieve the same effect.

4.3 Bidirectional Image Experiments

In addition to investigating autoregressive EBTs, we also explore the performance of EBTs trained bidirectionally. These experiments allow for a fairer comparison with diffusion models, which are not commonly trained autoregressively. Following [48, 118], models are trained to denoise images that have been noised. To make these denoising experiments compatible with diffusion models, we deviate from the original noising schemes performed in these works and use a noising scheme based on the noising schedule from diffusion models. Specifically, we follow [26], and use a linear variance schedule ranging from 1×10^{-4} to 2×10^{-2} . To control the noise level, we use a hyperparameter denoted σ representing the percentage of the diffusion schedule to noise samples; σ was set to 0.1 during training, and 0.2 during testing to test generalization. We use the COCO 2014 dataset [119, 120] with 128 by 128 images, a patch size of 16, and the Diffusion Transformer implementation from [26].

Following [48, 118], during inference we investigate denoising on the same level of noise performed during training ($\sigma = 0.1$), as well as at a higher noise level ($\sigma = 0.2$), representing Out-of-Distribution

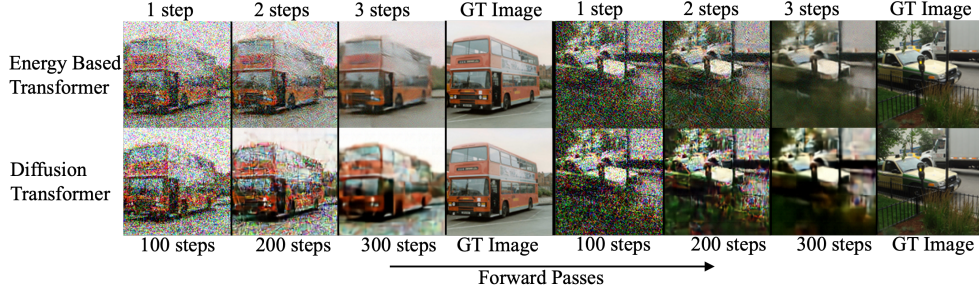


Figure 10: **Qualitative OOD Image Denoising.** EBTs achieve better denoising quality during inference while using one step for every 100 denoising steps of a DiT. The overall image quality of EBT denoised images is less blurry than images denoised by DiT.

(OOD) noisier images. The results are in Table 4, where we observe that EBTs perform better than DiTs at both in and out of distribution image denoising across various metrics, by as much as 3.5 in Peak Signal to Noise Ratio (PSNR). Following [91], we also plot the performance based on the number of forward passes (Number of Function Evaluations NFEs) in Figure 12. These results demonstrate that EBTs perform better at denoising while using 99% less denoising steps than DiTs, and that the System 2 Thinking scaling rate for EBTs is higher than for DiTs. Lastly, qualitative results for denoised out-of-distribution images for EBT compared to the DiT baseline are shown in Figure 10. These results further demonstrate that the visual quality of EBT denoised images is much better than the visual quality of DiT denoised images, while using less compute.

In an effort to understand whether the representations learned from denoising captured useful visual features, we perform a linear probe evaluation on ImageNet-1k [122], following common practice in visual representation learning [79, 80]. For both models, we take the average of all the final patch tokens, and for DiTs we feed in $T = 0$. The results are shown in Table 4, where the performance of EBTs is much higher than DiTs, achieving a Top-1 and Top-5 accuracy around $10\times$ higher than that of DiTs. This suggests that EBTs learn better image representations than DiTs, and therefore that EBTs offer promise in generative models with an **improved understanding** of what they generate.

5 Discussion

Across both discrete (text) and continuous (video) autoregressive models, the results show that EBTs scale at a faster rate than the standard Transformer++ approach during pretraining across all measured axes, including data, batch size, depth, parameters, FLOPs, and width. This is especially apparent with data and batch scaling for text, as well as width and parameter scaling for video, where the scaling rate was over 30% higher. These results are particularly important for two reasons: first, they suggest that at the scale of modern foundation models, EBTs would outperform the current Transformer++ approach *even without their System 2 Thinking capabilities*. Second, EBTs appear to be the first approach that has *better data ef-*

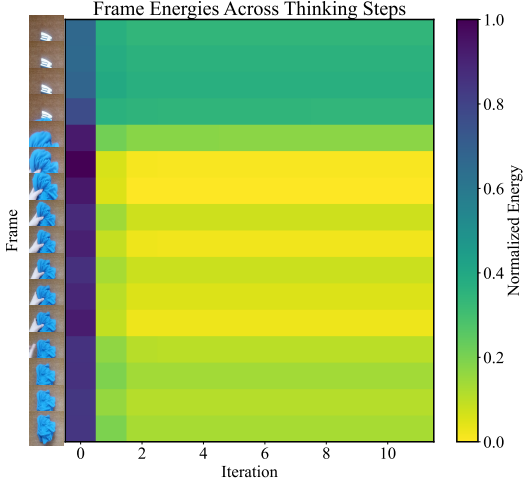


Figure 11: **Learning Uncertainty on Video Results.** In line with cognitive Facet 2, EBTs learn to express uncertainty across continuous video frames without supervision. At the start of the video, uncertainty is high (high energy) because the frame is mostly empty and the scene is highly unpredictable. As a blue garment is placed into the frame, uncertainty decreases (low energy), reflecting the greater predictability of the scene. When the blue garment is removed from the scene, uncertainty increases again, indicating a return to higher unpredictability. Such a capability is significantly more difficult to achieve in continuous spaces with traditional feed-forward transformers without discretization schemes [121].

efficiency than the Transformer++. As data has become one of the major limiting factors in further scaling [123], this makes EBTs especially appealing.

With System 2 Thinking, the EBT-Transformer++ performance gap would be expected to increase, as we found that System 2 Thinking could improve performance by as much as 29% for text (Figure 6a). Further, we found that the thinking capabilities of EBTs scale well, as they improve during pretraining (Figure 6b) and perform better for data that is more OOD (Figure 7). These findings imply that OOD generalization benefits from System 2 Thinking will further increase at larger scales, paving the way for principled generalization to OOD data. In addition, EBTs become increasingly robust to self-generated errors during verification (Figure B.1a), indicating that their self-verification process scales reliably and sidesteps the adversarial instabilities reported in prior works [19, 124].

We hypothesize that the superior scaling of EBTs compared to the Transformer++ can be attributed to EBTs learning to verify (Facet 3) rather than solely learning to predict. As discussed in Section 2, verification often generalizes better than amortized generation, which we believe leads to improved learning efficiency. The results in our generalization experiments further support this idea of verification leading to improved generalization, as we found that *given a slightly worse pretraining performance*, EBTs still outperform the Transformer++ recipe on downstream tasks. Additionally, corresponding to Facet 2 regarding prediction uncertainty, we find that the energy values learned by EBTs correlate strongly with more challenging to predict data, as shown in Figures 8 and 11. This is a promising characteristic of EBTs, as it enables uncertainty estimation within continuous state spaces, which would allow for principled inference-time behavior adaptation (Facet 1) when models determine a problem is more challenging.

Lastly, the results from our bidirectional experiments indicate that EBTs hold strong promise in bidirectional modeling. Particularly, we find that bidirectional EBTs outperform the DiT baseline in image denoising on both In and Out-of-Distribution performance, while using significantly fewer forward passes. This suggests that EBTs offer promise in tasks such as image generation or bidirectional text generation. Further, when evaluating the representations from DiTs and EBTs, we find that EBTs perform significantly better, achieving up to a 10 $\times$ improvement in accuracy, suggesting EBTs offer promise in developing a *better understanding* of what is being generated when performing generative modeling.

6 Related Work

6.1 Traditional Transformers

The Transformer architecture [83] has become ubiquitous across various domains [79, 82, 87, 125]. The most commonly used transformer variant of today makes predictions directly in the output space with a single forward pass, demonstrated in Figure 1a. Because these models have a finite depth and width, and make predictions in a single forward pass, they are unable to dynamically allocate more computation to *each* prediction being made. Furthermore, they cannot model uncertainty in continuous state spaces in the same way they can in discrete state spaces, because the normalization process for continuous state spaces is not as well-defined as it is for discrete spaces using softmax [42]. Rather, training these models to express uncertainty relies on tricks such as Vector Quantization [34] or pseudo losses/objectives (e.g., ELBO [35]). Finally, because these models are not trained to explicitly verify samples, improving inference-time performance at a per-prediction level often requires external models [50].

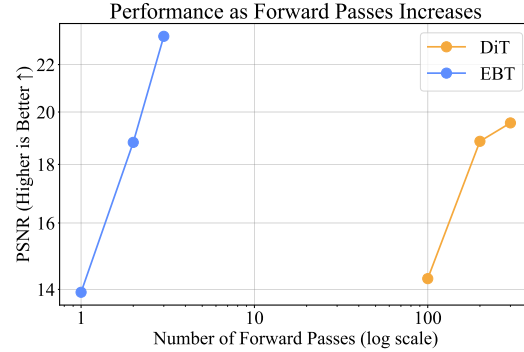


Figure 12: **Image Denoising Thinking Scalability.** A comparison between EBT and DiT on image denoising given a different number of forward passes. EBTs require only 1% of the forward passes used by DiT to achieve comparable or better PSNR. Further, the scaling rate of PSNR improvement given more forward passes is much higher for EBTs than it is for DiTs. These results suggest EBTs have superior thinking capabilities than DiTs on OOD data.

6.2 RNNs

Recently, several RNN variants (Figure 1b) have emerged to alleviate memory bottlenecks and achieve faster inference [22, 23]. These approaches have scaled similarly to Transformers in autoregressive sequence modeling and achieve better memory efficiency and reduced latency. However, traditional RNNs that update their internal state based solely on new information/data [22, 23] are not capable of allocating additional computation during inference, and thus suffer from the same flaws as traditional transformers in achieving human-like System 2 Thinking.

To resolve these issues, people have equipped RNNs with the ability to allocate computation dynamically, with architectures such as the Universal Transformer [126]. Recently, this type of RNN has also been applied to LLMs [25, 127], allowing LLMs to reason using additional computation in a continuous latent space through the depth of an unrolled RNN. However, like Diffusion models, these models learn to amortize gradient prediction of the energy function [25], meaning they cannot model uncertainty or explicitly verify predictions. Consequently, EBMs generalize these RNN-based architectures by offering explicit prediction verification capabilities [19]. Further discussion on this relationship is provided in Section E.

6.3 Dynamic Computation with LLMs

The ability to leverage a dynamic amount of computation in LLMs has been emulated using chain-of-thought prompting [128] and continuous latent space reasoning [29]. While these approaches can improve performance, they don’t seamlessly transfer to continuous modalities, and LLM chain-of-thought has been shown to be unreliable for reasoning [129–131]. More recently, models have been explicitly trained to perform reasoning using Reinforcement Learning [11–14]. These approaches allow LLMs to simulate additional computational depth based on the number of tokens decoded before making a prediction, and as a result, significantly improve performance [11, 12]. The main limitations of these approaches are that they currently apply only to discrete domains (i.e., LLMs), are effective on a narrow set of problems that are easily verifiable (e.g., math and coding), and require additional supervision, typically in the form of reward signals, making them incompatible with purely unsupervised pretraining [12].

6.4 Dynamic Computation with Diffusion

The most common instance of a model architecture specifically created to leverage dynamic computation is diffusion models (Figure 1c), where using multiple forward passes to generate a prediction is a core aspect of both training and inference [114, 132]. Although diffusion models implicitly define a likelihood through the reverse process [75], which could theoretically be used to verify predictions, in practice an external verifier is necessary to improve performance at inference time beyond increasing denoising steps [19–21]. This requirement limits the generalizability and scalability of diffusion models as an approach for System 2 Thinking, as they do not have two of the cognitive facets discussed in Table 1: the ability to model uncertainty in continuous state spaces and the ability explicitly verify predictions without additional models [19–21]. Furthermore, diffusion models rely on a fixed denoising schedule, which restricts their ability to adaptively halt or extend computation—unlike EBMs. Additionally, diffusion models can be seen as predicting the gradient of the data density/energy function [133], and therefore EBMs are a generalization of diffusion models that learn to explicitly verify predictions. More on this connection is in Section E, and a side-by-side comparison of diffusion models and EBMs is in Figure E.1.

6.5 Energy-Based Models (EBMs)

The perspective of energy minimization as thinking/reasoning has been known for some time [134]. Therefore, the most similar approaches to EBTs also train EBMs to do reasoning/thinking [48, 67]. While these works achieved impressive generalization results, they only focus on small-scale problems, and did not scale EBMs to high-dimensional real-world problems such as language or video. Additionally, these works did not perform an in-depth analysis on the types of System 2 Thinking that emerge with EBMs, more complex inference time procedures beyond just increasing the number of gradient descent steps, approaches towards improving EBM scalability, and required techniques for enhancing System 2 Thinking in EBMs.

7 Limitations and Conclusion

In this work, we proposed Energy-Based Transformers (EBTs), a new paradigm that frames System 2 Thinking as an optimization procedure with respect to a learned verifier (an Energy-Based Model), enabling System 2 Thinking to emerge across any problem or modality entirely from unsupervised learning.

Limitations. Despite demonstrating strong performance, EBTs have several current limitations. First, because EBTs generate predictions through an optimization process, they introduce additional hyperparameters, such as the optimization step size and the number of optimization steps. Tuning these hyperparameters is crucial for training stability, as we found poorly chosen values often lead to unstable training. Second, training and inference are more computationally expensive than standard feed-forward models, requiring additional gradient computations. Third, while EBTs scale well up to 800M parameters, we have not explored larger models due to resource constraints. However, experimental scaling trends demonstrate that EBTs scale faster than existing paradigms during pretraining, suggesting that EBTs would perform better at foundation-model scale. Finally, EBTs currently struggle with data distributions that have many modes, such as class conditional image generation, likely due to the convex energy landscape assumption made during training.

Conclusion. EBTs are the first instance of an approach that scales at a *faster rate* than the Transformer++ during pretraining across both continuous and discrete modalities. Additionally, our results suggest that EBTs scale better than existing approaches during inference at System 2 Thinking by dynamically allocating computational resources and self-verifying their own predictions; these System 2 Thinking capabilities enable improved generalization to out-of-distribution data. Ultimately, the superior scaling in both training and inference, coupled with improved generalization, positions EBTs as a promising new paradigm shift for advancing the intelligence of future foundation models.

8 Acknowledgement

We extend special thanks to Jeonghwan Kim and Cheng Qian for their helpful discussions. This research is based upon work supported by U.S. DARPA ECOLE Program No. HR00112390060 and DARPA ITM Program No. FA865023-C-7316. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. This research used the Delta and DeltaAI advanced computing and data resources, which are supported by the National Science Foundation (award OAC 2320345 and award OAC 2005572) and the State of Illinois. Delta and DeltaAI are joint efforts of the University of Illinois Urbana-Champaign and its National Center for Supercomputing Applications.

References

- [1] Daniel Kahneman. *Thinking, fast and slow*. macmillan, 2011. 1, 3, 30
- [2] Jonathan St BT Evans. Dual-process theories of reasoning: Contemporary issues and developmental applications. *Developmental review*, 31(2-3):86–102, 2011.
- [3] Daniel Kahneman, Shane Frederick, et al. Representativeness revisited: Attribute substitution in intuitive judgment. *Heuristics and biases: The psychology of intuitive judgment*, 49(49-81):74, 2002.
- [4] Keith Frankish. Dual-process and dual-system theories of reasoning. *Philosophy Compass*, 5(10): 914–926, 2010. 1
- [5] Wim De Neys. Dual processing in reasoning: Two systems but one reasoner. *Psychological science*, 17(5):428–433, 2006. 2
- [6] Vinod Goel, Christian Buchel, Chris Frith, and Raymond J Dolan. Dissociation of mechanisms underlying syllogistic reasoning. *Neuroimage*, 12(5):504–514, 2000. 2
- [7] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025. 2

- [8] Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024. 2
- [9] Yang Yan, Yu Lu, Renjun Xu, and Zhenzhong Lan. Do phd-level llms truly grasp elementary addition? probing rule learning vs. memorization in large language models. *arXiv preprint arXiv:2504.05262*, 2025. 6
- [10] Cheng Qian, Peixuan Han, Qinyu Luo, Bingxiang He, Xiushi Chen, Yuji Zhang, Hongyi Du, Jiarui Yao, Xiaocheng Yang, Denghui Zhang, Yunzhu Li, and Heng Ji. Escapebench: Pushing language models to think outside the box. In *arxiv*, 2025. 2
- [11] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helvar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. 2, 16, 30
- [12] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 2, 8, 16
- [13] xAI. Grok 3 Beta — The Age of Reasoning Agents, 2025. URL <https://x.ai/blog/grok-3>. Accessed: 2025-02-21. 2
- [14] Anthropic. Claude 3.7 sonnet and claude code, 2025. URL <https://www.anthropic.com/news/claude-3-7-sonnet>. Accessed: 2025-02-21. 2, 16
- [15] OpenAI. Learning to reason with llms, 2024. URL <https://openai.com/index/learning-to-reason-with-llms/>. Accessed: 2025-02-21. 2
- [16] Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Expanding rl with verifiable rewards across diverse domains. *arXiv preprint arXiv:2503.23829*, 2025.
- [17] Parshin Shojaei*, Iman Mirzadeh*, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity, 2025. URL <https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf>. 2
- [18] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025. 2
- [19] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025. 2, 3, 6, 8, 15, 16, 30, 36, 37
- [20] Fangfu Liu, Hanyang Wang, Yimo Cai, Kaiyan Zhang, Xiaohang Zhan, and Yueqi Duan. Video-t1: Test-time scaling for video generation. *arXiv preprint arXiv:2503.18942*, 2025.
- [21] Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025. 2, 16
- [22] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 2, 8, 9, 16, 27, 33, 34
- [23] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023. 2, 16
- [24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2
- [25] Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025. 2, 3, 12, 16, 41
- [26] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. 2, 4, 8, 13, 33, 34, 35, 43

- [27] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2025. [2](#), [8](#)
- [28] Chaorui Deng, Deyao Zhu, Kunchang Li, Shi Guang, and Haoqi Fan. Causal diffusion transformers for generative modeling. *arXiv preprint arXiv:2412.12095*, 2024. [2](#), [33](#)
- [29] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024. [3](#), [16](#)
- [30] Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023. [3](#)
- [31] Jochen Ditterich. Evidence for time-variant decision making. *European Journal of Neuroscience*, 24(12): 3628–3641, 2006. [3](#)
- [32] Nicolas P Rougier, David C Noelle, Todd S Braver, Jonathan D Cohen, and Randall C O’Reilly. Prefrontal cortex and flexible cognitive control: Rules without symbols. *Proceedings of the National Academy of Sciences*, 102(20):7338–7343, 2005. [3](#)
- [33] Christian Tomani, Kamalika Chaudhuri, Ivan Evtimov, Daniel Cremers, and Mark Ibrahim. Uncertainty-based abstention in llms improves safety and reduces hallucinations. *arXiv preprint arXiv:2404.10960*, 2024. [3](#)
- [34] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. [3](#), [15](#)
- [35] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013. [3](#), [15](#)
- [36] Karthik Abinav Sankararaman, Sinong Wang, and Han Fang. Bayesformer: Transformer with uncertainty estimation. *arXiv preprint arXiv:2206.00826*, 2022. [3](#)
- [37] Alvin Heng, Harold Soh, et al. Out-of-distribution detection with a single unconditional diffusion model. *Advances in Neural Information Processing Systems*, 37:43952–43974, 2024.
- [38] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don’t know? *arXiv preprint arXiv:1810.09136*, 2018.
- [39] Joan Serrà, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F Núñez, and Jordi Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. *arXiv preprint arXiv:1909.11480*, 2019. [3](#)
- [40] Christopher M Bishop. Mixture density networks. 1994. [3](#)
- [41] 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*, 2020. [3](#)
- [42] Anna Dawid and Yann LeCun. Introduction to latent variable energy-based models: a path toward autonomous machine intelligence. *Journal of Statistical Mechanics: Theory and Experiment*, 2024(10): 104011, 2024. [3](#), [6](#), [7](#), [15](#), [40](#)
- [43] A. Peters, B. McEwen, and Karl J. Friston. Uncertainty and stress: Why it causes diseases and how it is mastered by the brain. *Progress in Neurobiology*, 156:164–188, 2017. doi: 10.1016/j.pneurobio.2017.05.004. [3](#)
- [44] I. Vilares, J. D. Howard, Hugo L. Fernandes, J. Gottfried, and Konrad Paul Kording. Differential representations of prior and likelihood uncertainty in the human brain. *Current Biology*, 22:1641–1648, 2012. doi: 10.1016/j.cub.2012.07.010.
- [45] Issidoros C. Sarinopoulos, D. Grupe, Kristen L. Mackiewicz, J. Herrington, M. Lor, E. E. Steege, and J. Nitschke. Uncertainty during anticipation modulates neural responses to aversion in human insula and amygdala. *Cerebral cortex*, 20 4:929–40, 2010. doi: 10.1093/cercor/bhp155. [3](#)
- [46] Frank Loesche, Jeremy Goslin, and Guido Bugmann. Paving the way to eureka—introducing “dira” as an experimental paradigm to observe the process of creative problem solving. *Frontiers in Psychology*, 9: 1773, 2018. [3](#)

- [47] Zaid Alkouri. Using contents and containers to investigate problem solving strategies among toddlers. 2016. [3](#)
- [48] Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Learning iterative reasoning through energy minimization. In *International Conference on Machine Learning*, pages 5570–5582. PMLR, 2022. [3](#), [6](#), [7](#), [13](#), [16](#), [36](#)
- [49] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017. [3](#)
- [50] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023. [3](#), [15](#)
- [51] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in neural information processing systems*, 32, 2019. [4](#), [5](#), [6](#), [7](#), [8](#), [27](#), [42](#)
- [52] Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020.
- [53] Zengyi Li, Yubei Chen, and Friedrich T Sommer. Learning energy-based models in high-dimensional spaces with multiscale denoising-score matching. *Entropy*, 25(10):1367, 2023. [4](#)
- [54] Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. *arXiv preprint arXiv:2003.05033*, 2020. [4](#), [8](#), [42](#)
- [55] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. [4](#), [8](#)
- [56] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. [4](#), [8](#)
- [57] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018. [5](#)
- [58] Stephen A Cook. The complexity of theorem-proving procedures. In *Logic, automata, and computational complexity: The works of Stephen A. Cook*, pages 143–152. 2023. [5](#)
- [59] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. The knowledge complexity of interactive proof-systems. In *Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali*, pages 203–225. 2019. [5](#)
- [60] Kurt Gödel. Letter to john von neumann, 1956. URL <https://ecommons.cornell.edu/server/api/core/bitstreams/46aef9c4-288b-457d-ab3e-bb6cb1a4b88e/content>. Accessed: 2025-04-28. [5](#)
- [61] Ryan Lavin, Xuekai Liu, Hardhik Mohanty, Logan Norman, Giovanni Zaarour, and Bhaskar Krishnamachari. A survey on the applications of zero-knowledge proofs. *arXiv preprint arXiv:2408.00243*, 2024. [5](#)
- [62] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. [5](#)
- [63] AlphaCode Team. Alphacode 2 technical report. December 2023. [6](#)
- [64] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. [6](#)
- [65] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022. [6](#)
- [66] Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *arXiv preprint arXiv:2503.01067*, 2025. [6](#)
- [67] Yilun Du, Jiayuan Mao, and Joshua B Tenenbaum. Learning iterative reasoning through energy diffusion. *arXiv preprint arXiv:2406.11179*, 2024. [6](#), [16](#), [30](#), [37](#)

- [68] Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D Hwang, Liwei Jiang, Jillian Fisher, Abhilasha Ravichander, Khyathi Chandu, et al. The generative ai paradox: "what it can create, it may not understand". *arXiv preprint arXiv:2311.00059*, 2023. [6](#)
- [69] Gala Stojnić, Kanishk Gandhi, Shannon Yasuda, Brenden M Lake, and Moira R Dillon. Commonsense psychology in human infants and machines. *Cognition*, 235:105406, 2023. [6](#)
- [70] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*, 2024. [6](#)
- [71] Ze Wang, Jiang Wang, Zicheng Liu, and Qiang Qiu. Energy-inspired self-supervised pretraining for vision models. *arXiv preprint arXiv:2302.01384*, 2023. [6](#), [7](#), [38](#), [40](#), [43](#)
- [72] Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62, 2022. [6](#)
- [73] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. [7](#)
- [74] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003. [7](#)
- [75] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. [7](#), [16](#)
- [76] Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. How to compute hessian-vector products? In *ICLR Blogposts 2024*, 2024. URL <https://iclr-blogposts.github.io/2024/blog/bench-hvp/>. <https://iclr-blogposts.github.io/2024/blog/bench-hvp/>. [7](#), [35](#)
- [77] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020. [8](#)
- [78] OpenAI. Gpt-4 technical report, 2023. [8](#)
- [79] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. [14](#), [15](#)
- [80] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. [14](#)
- [81] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audioldm: a language modeling approach to audio generation, 2023. [8](#)
- [82] 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. [8](#), [15](#), [30](#)
- [83] 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. [8](#), [15](#), [31](#)
- [84] 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. [8](#), [9](#), [10](#), [11](#), [12](#), [28](#)
- [85] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. [8](#), [33](#), [34](#)
- [86] Margaret Li, Sneha Kudugunta, and Luke Zettlemoyer. (mis) fitting: A survey of scaling laws. *arXiv preprint arXiv:2502.18969*, 2025. [8](#), [9](#), [33](#)

- [87] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. 8, 9, 10, 11, 15, 33, 42
- [88] Ibrahim M Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. *Advances in Neural Information Processing Systems*, 35:22300–22312, 2022.
- [89] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.
- [90] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. 8
- [91] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018. 8, 14
- [92] Maurice Weber, Dan Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, et al. Redpajama: an open dataset for training large language models. *Advances in neural information processing systems*, 37:116462–116492, 2024. 8
- [93] Together Computer. Redpajama: an open dataset for training large language models, 2023. URL <https://github.com/togethercomputer/RedPajama-Data>. 8
- [94] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022. 8
- [95] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024. 8
- [96] Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. In *The Thirteenth International Conference on Learning Representations*, 2024. 9
- [97] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 9
- [98] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. 9
- [99] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022. 9, 10
- [100] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *Advances in Neural Information Processing Systems*, 36:55565–55581, 2023. 9
- [101] Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, et al. Byte latent transformer: Patches scale better than tokens. *arXiv preprint arXiv:2412.09871*, 2024. 9, 10
- [102] Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, et al. Language models scale reliably with over-training and on downstream tasks. *arXiv preprint arXiv:2403.08540*, 2024. 11

- [103] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don’t stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.
- [104] Tristan Thrush, Christopher Potts, and Tatsunori Hashimoto. Improving pretraining data using perplexity correlations. *arXiv preprint arXiv:2409.05816*, 2024. 11
- [105] Yangyi Chen, Binxuan Huang, Yifan Gao, Zhengyang Wang, Jingfeng Yang, and Heng Ji. Scaling laws for predicting downstream performance in llms. *arXiv preprint arXiv:2410.08527*, 2024. 11, 12, 34
- [106] Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. Scaling laws for downstream task performance of large language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024. 11, 12
- [107] Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. Autoregressive video generation without vector quantization. *arXiv preprint arXiv:2412.14169*, 2024. 12
- [108] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. *arXiv preprint arXiv:1906.02634*, 2019.
- [109] Ruslan Rakhimov, Denis Volkhonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. Latent video transformer. *arXiv preprint arXiv:2006.10704*, 2020.
- [110] Xi Ye and Guillaume-Alexandre Bilodeau. Video prediction by efficient transformers. *Image and Vision Computing*, 130:104612, 2023.
- [111] Yuchao Gu, Weijia Mao, and Mike Zheng Shou. Long-context autoregressive video modeling with next-frame prediction. *arXiv preprint arXiv:2503.19325*, 2025. 12
- [112] Pablo Villalobos, Jaime Sevilla, Lennart Heim, Tamay Besiroglu, Marius Hobbhahn, and Anson Ho. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. *arXiv preprint arXiv:2211.04325*, 2022. 12
- [113] URL <https://research.google/blog/data-centric-ml-benchmarking-announcing-dataperfs-2023-challenges/>. 12
- [114] 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*, pages 10684–10695, 2022. 12, 16, 33, 35, 40
- [115] URL <https://huggingface.co/stabilityai/sd-vae-ft-mse>. 12, 33, 35
- [116] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017. 12
- [117] Md Mofijul Islam, Alexi Gladstone, Riashat Islam, and Tariq Iqbal. Eqa-mx: Embodied question answering using multimodal expression. In *The Twelfth International Conference on Learning Representations*, 2023. 13
- [118] Xinshi Chen, Hanjun Dai, Yu Li, Xin Gao, and Le Song. Learning to stop while learning to predict. In *International conference on machine learning*, pages 1520–1530. PMLR, 2020. 13
- [119] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer vision—ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014. 13, 34
- [120] URL https://huggingface.co/datasets/AbdoTW/COCO_2014. 13, 34
- [121] Jiahuan Pei, Cheng Wang, and György Szarvas. Transformer uncertainty estimation with hierarchical stochastic attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11147–11155, 2022. 14
- [122] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 14

- [123] URL https://www.youtube.com/watch?v=6nJZopACRuQ&ab_channel=OpenAI. 15
- [124] Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Dylan J Foster, and Akshay Krishnamurthy. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. *arXiv preprint arXiv:2503.21878*, 2025. 15
- [125] Siddique Latif, Aun Zaidi, Heriberto Cuayahuitl, Fahad Shamshad, Moazzam Shoukat, and Junaid Qadir. Transformers in speech processing: A survey. *arXiv preprint arXiv:2303.11607*, 2023. 15
- [126] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018. 16
- [127] Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025. 16, 36
- [128] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022. 16
- [129] Tianhe Lin, Jian Xie, Siyu Yuan, and Deqing Yang. Implicit reasoning in transformers is reasoning through shortcuts. *arXiv preprint arXiv:2503.07604*, 2025. 16
- [130] Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don’t always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.
- [131] Chirag Agarwal, Sree Harsha Tanneru, and Himabindu Lakkaraju. Faithfulness vs. plausibility: On the (un) reliability of explanations from large language models. *arXiv preprint arXiv:2402.04614*, 2024. 16
- [132] Tobias Höppe, Arash Mehrjou, Stefan Bauer, Didrik Nielsen, and Andrea Dittadi. Diffusion models for video prediction and infilling, 2022. 16
- [133] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and memc. In *International conference on machine learning*, pages 8489–8510. PMLR, 2023. 16, 36, 38
- [134] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fugie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006. 16
- [135] Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: LLMs trained on "a is b" fail to learn "b is a". *arXiv preprint arXiv:2309.12288*, 2023. 26
- [136] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022. 26
- [137] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.
- [138] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024. 26
- [139] Anton Bakhtin, Yuntian Deng, Sam Gross, Myle Ott, Marc’Aurelio Ranzato, and Arthur Szlam. Residual energy-based models for text. *Journal of Machine Learning Research*, 22(40):1–41, 2021. 26, 38
- [140] Sumanta Bhattacharyya, Amirmohammad Rooshenas, Subhajit Naskar, Simeng Sun, Mohit Iyyer, and Andrew McCallum. Energy-based reranking: Improving neural machine translation using energy-based models. *arXiv preprint arXiv:2009.13267*, 2020. 26, 38
- [141] Rodney J Douglas and Kevan AC Martin. Recurrent neuronal circuits in the neocortex. *Current biology*, 17(13):R496–R500, 2007. 27
- [142] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017. 27
- [143] Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024. 28

- [144] Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: Lms can predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725*, 2024. 30
- [145] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. 30
- [146] Thomas Parr, Giovanni Pezzulo, and Karl J Friston. *Active inference: the free energy principle in mind, brain, and behavior*. MIT Press, 2022. 30
- [147] William A Falcon. Pytorch lightning. *GitHub*, 3, 2019. 33
- [148] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. Gpt-neox-20b: An open-source autoregressive language model, 2022. 33, 35, 36
- [149] Shengding Hu, Xin Liu, Xu Han, Xinrong Zhang, Chaoqun He, Weilin Zhao, Yankai Lin, Ning Ding, Zebin Ou, Guoyang Zeng, et al. Predicting emergent abilities with infinite resolution evaluation. *arXiv preprint arXiv:2310.03262*, 2023. 34
- [150] Adam Casson. Transformer flops. 2023. URL <https://adamcasson.com/posts/transformer-flops>. 35
- [151] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pages 423–439. Springer, 2022. 36
- [152] Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed Zaki, and Dmitry Krotov. Energy transformer. *Advances in Neural Information Processing Systems*, 36, 2024. 38
- [153] Yezhen Wang, Tong Che, Bo Li, Kaitao Song, Hengzhi Pei, Yoshua Bengio, and Dongsheng Li. Your autoregressive generative model can be better if you treat it as an energy-based one. *arXiv preprint arXiv:2206.12840*, 2022. 38
- [154] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011. 38, 40
- [155] Angela D Friederici and Jürgen Weissenborn. Mapping sentence form onto meaning: The syntax–semantic interface. *Brain research*, 1146:50–58, 2007. 38
- [156] Seijin Kobayashi, Simon Schug, Yassir Akram, Florian Redhardt, Johannes von Oswald, Razvan Pascanu, Guillaume Lajoie, and João Sacramento. When can transformers compositionally generalize in-context? *arXiv preprint arXiv:2407.12275*, 2024. 38
- [157] Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems*, 36:78723–78747, 2023. 38
- [158] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982. 41
- [159] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 42
- [160] Noam Shazeer. Glu variants improve transformer, 2020. 42
- [161] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 42

In this appendix, we provide additional insight and details on EBTs. First, we provide more insight on the broader impact/future work of EBTs in Section A. Next, in Section B, we include additional experiments. Then, we include additional approach details in Section C, as well as additional experimental details in Section D. After that, we include a comprehensive related work in Section E, additional facets of cognition in Section F, and a discussion of counterarguments in Section G. Finally, in the hopes of making EBTs more accessible to general audiences, in Section H we contain a general easier-to-understand intro to EBMs, and in Section I we describe a tutorial for getting started with EBTs.

A Future Works and Broader Impact

EBTs, being qualitatively different from existing approaches, open several future research directions.

A.1 Reversal Curse

Recently, a phenomenon known as “The Reversal Curse” has been observed where LLMs fail to learn a symmetric mapping [135] “B is A” despite learning “A is B”. For example, LLMs trained on an example such as “Q: Who is Tom Cruise’s mother? A: Mary Lee Pfeiffer” often fail to generalize to know the answer to the reverse question of “Who is Mary Lee Pfeiffer’s son?” Remarkably, the Reversal Curse has manifested itself in LLMs regardless of the size or scale [135]—probing researchers to investigate whether there are fundamental limitations to traditional feed-forward LLMs. One predicted cause of The Reversal Curse is the nature of gradient updates, where backpropagation only updates tokens within context. That is, while learning the mapping “A is B”, none of B’s tokens are within context, meaning they do not receive gradient updates when updating A’s tokens. We hypothesize that LLMs trained with EBTs rather than standard feed-forward Transformers could help reduce this phenomenon, as with EBTs the tokens of A and B are within context during gradient updates due to predictions being made in the input space. Therefore, an exciting research direction would be investigating whether this hypothesis is correct in LLMs trained with EBTs, allowing improved generalization.

A.2 Improved Stability

In this work, we primarily trained EBTs with either two or three optimization steps. While these parameters worked well, we suspect that increasing the number of steps would improve the System 2 Thinking capabilities and the scaling during pretraining of EBTs, as more steps enable a longer “thinking process” before needing to converge. However, because of challenges in stability when training with more steps, due to a larger gradient computation graph, we were unable to successfully increase past two or three steps. Future work could focus more on extending the number of steps by studying ways to improve the training stability of EBMs.

A.3 World Models

In this work, we focus on autoregressive and bidirectional models over just state information (no actions). EBTs offer high promise in modeling states and actions due to the nature of EBMs learning a distribution over all possible inputs. Particularly, given a model trained to estimate the unnormalized joint distribution of the current context, future, as well as future actions, such world models could implicitly be used as policies to generate actions to achieve a specific state, similar to [136–138]. This would involve holding the current context (past states) constant, and minimizing the energy by propagating the gradient back to the action inputs and future state predictions. Thus, world models trained in this manner become capable of more than just predicting the future, but also in decision making to achieve a specific goal state.

A.4 EBTs as Complementary Models

As demonstrated in [139, 140], EBMs can be used to improve the quality of generated text from language models. It’s possible EBTs could be used in a similar manner for a broad variety of tasks, serving as the verifier of predictions initialized by standard feed-forward models. Therefore, although we do a side-by-side comparison to existing models in this work, EBTs could be **complementary** to

existing model paradigms—being used as the System 2 Backbone for helping lighter models that perform System 1 thinking.

There exist several current real-world use cases, such as low-latency LLM serving, where doing a single forward pass is sufficient, and where the added inference overhead of gradients with EBTs would not be worth the extra computation. However, we also envision a world in which people use EBTs for long-term System 2 Thinking to solve challenging problems. How much computation would it be worth dedicating to prove a long-standing mathematical conjecture, or figuring out a cure to cancer?

A.5 Recurrent Energy-Based Models

While EBTs scale well, for latency-driven use cases, Transformers require significantly more memory than Recurrent Neural Networks. Additionally, there is strong evidence for recurrence in the human brain [141]. Therefore, we anticipate that recurrent Energy-Based Models, possibly leveraging the Mamba architecture [22], will eventually become common.

A.6 Improved Thinking Algorithms

The EBM thinking algorithms described have strong connections to or are derived from Markov Chain Monte Carlo (MCMC) sampling. Therefore, we broadly expect known MCMC samplers with more advanced techniques for traversing the energy landscape to be successful, such as Hamiltonian Monte Carlo [142] or annealed Langevin dynamics [51]. Additionally, we did not explore more advanced search algorithms such as Monte Carlo Tree Search, which we suspect could offer performance improvements and leave for future work.

A.7 Multimodal Energy-Based Models

We did not experiment with multimodal EBMs, however, EBMs offer several advantages for learning over multiple modalities. For example, multimodal EBMs would enable a single energy scalar to represent the alignment between modalities, and would simplify joint training across modalities by providing a unified objective that naturally captures inter-modal dependencies.

A.8 Thinking Scalability

Due to a lack of computational resources, we were unable to train models with more than 10^{21} FLOPs (≈ 1300 A100 GPU Hours). Therefore, training and thinking with EBTs remains untested at larger foundation model scale. We leave it to future work to scale with more GPUs and investigate the qualitative differences in training and thinking with EBTs.

A.9 Learning Multimodal Distributions

We found that EBTs, with the current training approach, struggle to capture distributions with many modes (e.g., unconditional image generation). Therefore, future work could explore approaches to improve the learning of distributions with many modes. More info is in Section B.2.

A.10 Understanding Predictions

We believe that there exists a fundamental distinction between the internal representations associated with model inputs and outputs. Specifically, models generate internal representations **of** inputs, as these serve as the foundation that dictate the model’s behavior at any given point in time. Conversely, as outputs do not affect a model’s behavior at a given point in time, we contend that models construct representations **for predicting** (and not necessarily understanding) outputs. This distinction leads us to an interesting insight: *models may not achieve a genuine understanding of outputs in the same way they understand inputs*. This implies that while models may develop an intricate understanding of input data, such understanding does not naturally extend to predictions that are made in the output space. Therefore, existing feed-forward models primarily making predictions in the output space may not **understand** their predictions in the same way they understand their inputs. This intuition further supports the principles behind EBT, where predictions are made in the input space, enabling

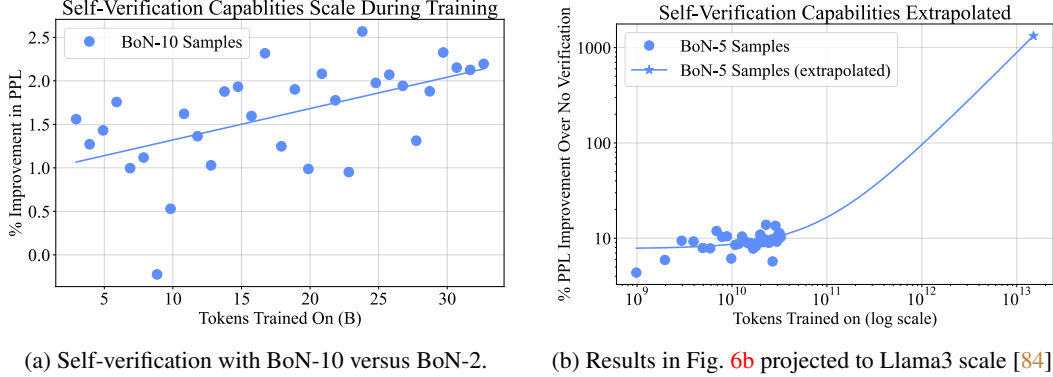


Figure B.1: **EBT Thinking Analysis for Data Scaling.** (a) Self-verification of BoN-2 compared to BoN-10. EBTs become less adversarial and thus benefit more from verifying an increasing number of samples during training. (b) A projection of the results from Figure 6b to the data scale of Llama3 [84], demonstrating that as data scale increases, improvements from self-verification can lead to potentially massive performance increases from System 2 Thinking.

representations of **predictions** to be developed. We leave the investigation of this hypothesis to future work.

B Additional Experimentation

B.1 Additional Natural Language Processing Experiments

We conduct experiments to confirm hypotheses on thinking results obtained in the main paper. First, we confirm that EBTs become less adversarial with scale by comparing the performance of using BoN with 2 versus 10 samples. The results in Figure B.1a demonstrate that when models are trained on less tokens, there is little performance improvement by verifying 10 samples instead of just 2. In fact, verifying 10 samples occasionally leads to *worse* performance than verifying 2 samples, likely because the EBT found an adversarial sample (a sample with low energy that is in fact not a good prediction). However, as data scale increases we observe that performance improvements from BoN-10 versus BoN-2 increase, and that these adversarial dynamics decrease. Together, these results suggest that with scale EBTs become less adversarial due to an improved energy landscape. In an effort to understand the impacts of thinking at the scale of modern foundation models, we project results from Figure 6b to the scale of modern foundation models [84] to extrapolate a projected performance gain from self-verification based thinking. The results are visualized in Figure B.1b, where they demonstrate that, because of the $1000\times$ data scale of modern foundation models, the performance improvement from self-verification increases drastically.

Additionally, in an effort to understand whether EBTs can capture epistemic uncertainty (uncertainty related to knowledge), in addition to aleatoric uncertainty (uncertainty related to inherent data randomness), we visualize the energies of different token sequences that are in versus Out-Of-Distribution (OOD) in Figure B.2. The results demonstrate that, for a more in-distribution sequence, EBTs have lower energy (less uncertainty), than for an OOD sequence. This suggests that EBTs learn to *know what they don't know*, as they learn to have higher energy for OOD sequences signifying harder predictability.

To confirm some of the scaling trends in the paper, as well as experiment with additional datasets, we conduct larger-scale experiments on the FineWeb dataset [143]. Particularly, for Figure 4a, we confirm that the generally better data scaling of EBTs compared to the Transformer++ holds at increased scale. We confirm this by training small-sized models with a batch size of 256, a context length of 1024, and for 500,000 training steps. These results are visualized in Figure B.3, where we observe that EBTs still continue to outscale the Transformer++ by as much as 35% in scaling rate. These results further reinforce that EBTs are more data-efficient than the Transformer++.

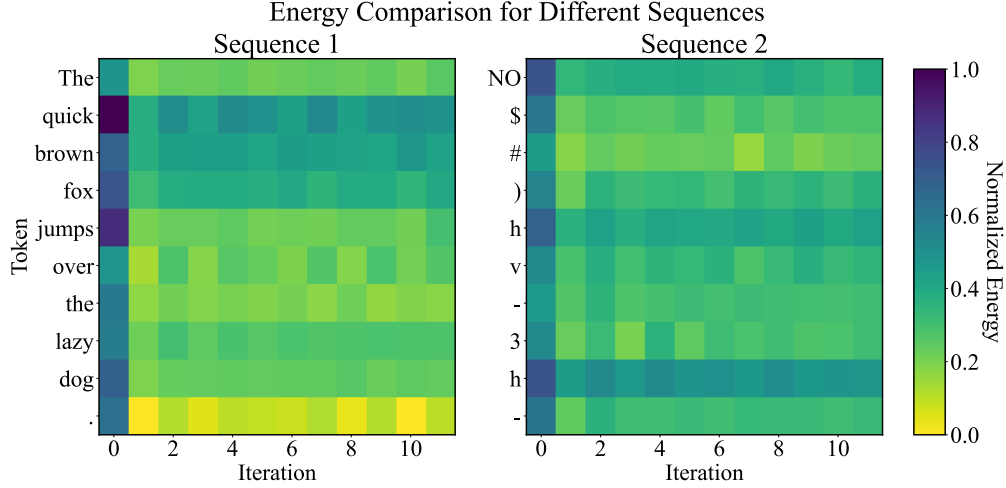


Figure B.2: **Epistemic Uncertainty Comparison.** EBTs learn to express epistemic uncertainty (uncertainty related to lack of knowledge) on unseen data. Particularly, the sequence on the left, which is a text sequence likely seen during training, has consistently lower energy (uncertainty) for tokens than the sequence on the right, which is a random text sequence not from the training distribution. This demonstrates that EBTs learn to “know what they don’t know.”

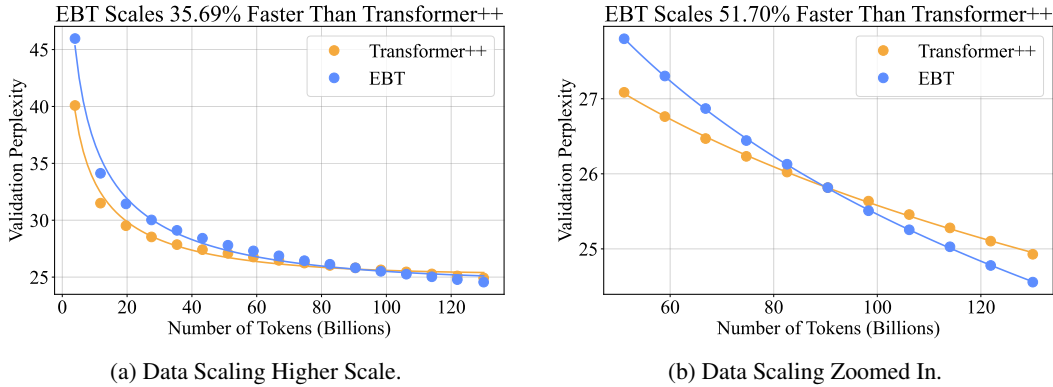


Figure B.3: **EBT Larger Scale Data Scaling.** (a) Data scaling results from Figure 4a scaled up to a larger model size, higher context length, increased batch size, and a higher number of training steps. (b) Results from (a) zoomed in.

B.2 EBT Failure Cases

During experimentation, along with image denoising experiments, we also conducted small-scale experiments with text-to-image generation. We found that, in datasets such as COCO with many different modes for a single condition (e.g., hundreds of images with a caption similar to “giraffe with a long neck”), that EBTs did not learn to generate high-quality novel images. Instead, EBTs often generated blurred images similar to the training distribution. We believe this is caused by the training approach pushing the energy landscape to be convex surrounding the training examples (modes). Therefore, when there are many different modes within the same region (same condition), this convex energy landscape “merges” to one landscape averaged around the different modes, resulting in blurriness. We believe that this is not a fundamental limitation of EBTs, and that future work could address this issue, such as by adding more conditioning.

C Additional EBT Details

C.1 Formalizing Thinking

Due to the recent surge of interest in scaling the performance of models during inference/test time, there are several common terms used to refer to these ideas. These terms include scaling the thinking capabilities of models [11], inference-time scaling [19], inference-time compute [144], and test-time compute [11, 145]. Therefore, to reduce confusion stemming from a wide variety of terminology and unite the community, in this work we broadly define these concepts as **System Two Thinking** or more concisely **Thinking**. We formalize improvements made by Thinking as the following:

Definition C.1 (System 2 Thinking). *Given a problem with data x , a model θ , and additional computational resources in the form of function evaluations F greater than the minimum number of function evaluations to get a valid prediction from the model F_0 , **System Two Thinking** $STT(\cdot)$ quantifies the expected percentage improvement in performance as F increases. Let $P(x, \theta, F)$ be the performance on input x when the model θ uses F function evaluations:*

$$STT(x, \theta, F) = \mathbb{E}_x \left[\frac{P(x, \theta, F)}{P(x, \theta, F_0)} - 1 \right],$$

This formalization is compatible with any type of metric (e.g., Accuracy, Perplexity, FID, etc), and uses more psychology-aligned terminology [1]. Further, avoiding terms such as “inference” or “test-time” makes the idea of Thinking more compatible with domains where the line between inference and training is blurry, such as real-world continual learning, domain adaptation, or actual human learning/thinking processes [146]. Just as **learning** has become a flexible term across machine learning representing several different ideas, we intend for **thinking** to similarly unify many diverse ideas under a common framework. For a greater justification on this perspective, please see Section G

C.2 Energy-Based Transformer (EBT) Thinking Types

All experiments in the paper are conducted with two main variants of EBTs, which we call System 1 (S1) and System 2 (S2) EBTs. S1-EBTs have hyperparameters specifically optimized for stability and learning convergence, whereas S2-EBTs have hyperparameters optimized for System 2 Thinking capabilities. Many of the pretraining scaling experiments conducted in Section 4 are with S1 models as to reduce the computational resources required for experimentation. To confirm that these results hold for S2 models, we plot the scaling trends of S1 and S2 models side by side; in Figure C.1, we find that S2 models scale *at the same or a higher rate* than S1 models during training, but have a higher Y-intercept. This Y-intercept offset does not affect asymptotic scaling behavior (as asymptotically the scaling rate dominates), and hence, scaling trends that hold for S1 models should generally hold for S2 models. In fact, S2 models may even perform better than S1 models during pretraining asymptotically because of the higher scaling rate. Intuitively, switching from S1 to S2 models allows for a compute trade-off between the model’s pretraining performance and the model’s System 2 Thinking capabilities.

S1 models have the gradient of predictions detached between optimization steps to increase training stability. Conversely, following [67], the S2 models truncate backpropagation and avoid detaching prediction tensors between optimization steps. Additionally, the S2 models have all the energy landscape regularization techniques described in Section 3.3, whereas the S1 models have none. We also find that S2 models require a different value for the optimization step size, that the optimization step size not be learned, and to perform a minimum number of optimization steps greater than one.

C.3 Autoregressive Causal Energy-Based Transformers Efficient Implementation

GPT-style decoder-only autoregressive Transformers are parallelizable due to making all next-state predictions simultaneously [82]. In this section, we detail the implementation of decoder-only autoregressive EBTs, which also parallelize all next-state predictions simultaneously, but involve more complexity due to EBTs making predictions in the input space rather than the output space.¹⁰

¹⁰For simplicity we often use the term matrix to refer to slices of tensors.

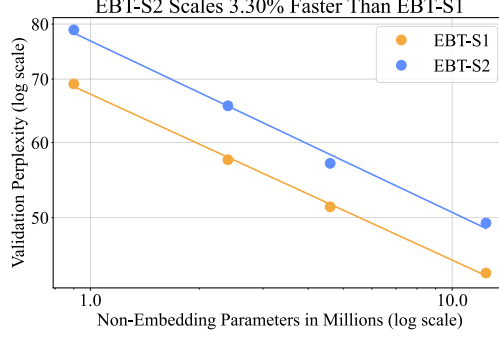


Figure C.1: **EBT S1 and S2 Scaling Comparison.** Scaling rate of System 1 (S1) compared to System 2 (S2) models. System 2 models have a higher Y-intercept but scale slightly faster than System 1 models. Therefore, as the scaling rate ultimately dominates asymptotic scaling behavior, and not the Y-intercept, scaling results in the main paper that hold for S1 models should generally hold for S2 models.

To demonstrate why this poses a challenge, for the decoder-only autoregressive Transformer, consider the case of the $N \times N$ attention scores matrix after the causal mask has been applied:

$$\text{scores} = \begin{bmatrix} \alpha_{z_1, z_1} & 0 & \dots & 0 \\ \alpha_{z_2, z_1} & \alpha_{z_2, z_2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \alpha_{z_n, z_1} & \alpha_{z_n, z_2} & \dots & \alpha_{z_n, z_n} \end{bmatrix},$$

where α_{z_i, z_j} represents the attention score (probability mass) from observed state z_i to observed state z_j . Now, in the case of an EBM, where predictions are made in the input space, the desired $N \times (N + 1)$ attention scores matrix would look like the following:

$$\text{scores} = \begin{bmatrix} \alpha_{z_1, z_1} & \alpha_{z_1, \hat{z}_2} & 0 & \dots & 0 \\ \alpha_{z_2, z_1} & \alpha_{z_2, z_2} & \alpha_{z_2, \hat{z}_3} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \alpha_{z_n, z_1} & \alpha_{z_n, z_2} & \alpha_{z_n, z_3} & \dots & \alpha_{z_n, \hat{z}_{n+1}} \end{bmatrix}, \quad (3)$$

where $\hat{z}_j$ denotes a predicted state (as opposed to an observed state). This is challenging to compute because each $\hat{z}_j$ along the superdiagonal is unique for its row, as we chose for each prediction to be made independently from each other prediction.¹¹ Consequently, unlike standard attention in feed-forward Transformers, this attention matrix cannot be computed with a matrix multiplication ($\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$), as every element of the superdiagonal is a prediction and not an observed state.

Additionally, in traditional transformers, if the context length is N , the size of manipulated tensors will generally be $B \times N \times D$ where B is the batch size and D is the embedding dimension. However, because EBMs make predictions in the input space, the input tensor needs to be different to allow for inputting predictions. Therefore, to distinguish these tensors, for a context length of N tokens (meaning we are given N tokens), we define a sequence of the first N elements as z_o , or the observed sequence representations, and the final N elements as z_p , or the predicted representations (for each next element in the sequence). This means that the manipulated tensors within the EBT are of size $B \times 2N \times D$ for a context length of size N .

The values for z_o , or the observed states, are computed identically as in the original transformer [83], as the attention scores of the observed states do not depend on the predicted states. This can be formalized as the following:

$$\text{Attention}(Q_o, K_o, V_o)_{z_o} = \text{softmax}\left(\frac{Q_o K_o^T}{\sqrt{d_k}}\right) V_o,$$

¹¹We note that it's possible to make each prediction not be independent, however, not making every prediction independent would mean that there is stochasticity for each prediction caused not only by its initial value but also by all initial values of all previous predictions.

where Q_o , K_o , and V_o are the Query, Key, and Value matrices of the observed states z_o . In every block of the transformer, the representations of all observed states are updated in this manner, independent of the representations of the predictions.

For the computation of prediction representations, three matrices are computed, which we notate as Q_p , K_p , and V_p .¹² First, we compute the self-attention scores of all predicted representations to all observed representations:

$$\tilde{S}_{z_o \leftarrow z_p} = \frac{Q_p K_o^T}{\sqrt{d_k}},$$

where $\tilde{S}_{z_o \leftarrow z_p}$ denotes the raw unnormalized attention scores of the predicted states to the observed states. Note, however, that the self-attention scores of each prediction with itself have not yet been calculated—due to the key matrix being from the observed states. Therefore, the superdiagonal of the $\tilde{S}_{z_o \leftarrow z_p}$ matrix needs to be replaced with the self-attention scores of each prediction with itself to achieve the attention score matrix shown in Equation 3.

To achieve this matrix, we first need to append a column to the right side of the $\tilde{S}_{z_o \leftarrow z_p}$ matrix, as the size of the matrix is currently $N \times N$, but the matrix needs to be $N \times (N + 1)$ (N for the observed states and then one for the predicted states along the second dimension). After doing this, we first mask out the superdiagonal with zeros to ensure that the probabilities in the superdiagonal of the score matrix only correspond to the values of predicted states with themselves. To make this operation differentiable, this masking operation is done through elementwise multiplication of a matrix with ones everywhere except the superdiagonal, which has zeros. Then, we compute the self-attention scores of each predicted state with itself, using the following equation:

$$\tilde{S}_{z_p \leftarrow z_p} = \frac{\text{sum}(Q_p * K_p)}{\sqrt{d_k}},$$

where the $*$ indicates the Hadamard product and the sum is across the feature dimension. Using a superdiagonal mask again, we set the diagonal of the $\tilde{S}_{z_o \leftarrow z_p}$ to these values. We denote the updated matrix as $\tilde{S}_{z_p}$, representing that this matrix is still unnormalized scores but takes into account interaction between z_p and z_o as well as from z_p to themselves. Now, after applying the softmax:

$$S_{z_p} = \text{softmax}(\tilde{S}_{z_p}),$$

we have the intended scores matrix shown in Equation 3. Note that for this softmax operation we adjust the standard causal attention mask to ensure all future information is masked out and the superdiagonal we added is not masked out.

One more barrier towards finally extracting all updated z_p representations is the fact that we cannot simply multiply this resulting scores matrix by the values matrix, as each element of the superdiagonal corresponds to a different predicted state. Thus, using similar techniques to before, we first clone and then extract the superdiagonal from this scores matrix using a diagonal mask.

After extracting out the superdiagonal, zeroing it out after extraction, and removing the earlier appended right-most column, we can multiply the resulting scores matrix by the V_o matrix to get all of the representations summed together of each predicted state with all observed states. This is computed using the following matrix multiplication:

$$z_p = S_{z_p} \cdot V_o.$$

As we also need to add the representation of each predicted state weighted with its own attention score (what was extracted on the superdiagonal), we perform another Hadamard product of the V_p matrix with the cloned superdiagonal to get these values, and then add these element wise to the z_p representations. Now, we have computed the intended representations involving the scores matrix shown in Equation 3. Now, z_o and z_p are updated by multiplying these tensors by the shared output weight matrix W_o .

¹²In practice, we shared the weights for the Q/K/V matrices for both observations and predictions to enable a one-to-one comparison to existing feed-forward transformers. It would be interesting to experiment with using different weight matrices to determine if that improves convergence and generalization at the cost of more parameters.

Table D.1: **Model sizes and hyperparameters for scaling experiments.** For applicable model sizes we follow Mamba [22].

Size	Non-Embedding Params	# layers	embed. dim	# heads
xxs	6.18M	6	384	6
xs	12.4M	12	384	6
small	48.8M	12	768	12
medium	176M	24	1024	16
large	396M	24	1536	16
xl	708M	24	2048	32

Table D.2: **Hyperparameters for Transformer++.**

Hyperparameter	CV	NLP
Optimizer	AdamW	
Optimizer Momentum	$\beta_1, \beta_2 = 0.9, 0.999$	
LR Schedule	Linear warm up cosine decay	
Warmup steps	1e4	
Minimum LR Scale	10	
Gradient Clip Value	1	
Weight Decay	0.01	
Context Length	16	256
Encoder	SD-XL VAE [114, 115]	-
Image Dimension	224x224	-
Tokenizer	-	EleutherAI/gpt-neox-20b [148]
Vocab Size	-	50277

C.4 Autoregressive Energy-Based Transformers Simplified Implementation

A more simplified implementation involves the entire attention matrices and a generalized causal mask, as described in [28]. However, because this implementation involves a matrix multiplication with 2 times the sequence length, this results in 4 times the number of FLOPs as normal attention, which is around double the number of FLOPs of our more efficient implementation.

D Experimentation Details

Tables D.2 and D.3 specify general model information and hyperparameters. We utilized the Llama 2 transformer implementation [87] for the Transformer++ and used this implementation as the backbone upon which we built causal decoder-only EBTs. We seed all libraries using PyTorch Lightning [147] for all experiments with a seed of 33. For the Diffusion Transformer, we use the implementation from [26]—for the bidirectional EBT we build upon this implementation.

Unless otherwise stated, for all scaling experiments we copy the popular Mamba paper pretraining settings for the model configuration (we use different batch sizes/data due to computational constraints), shown in Table D.1. Because we are compute-constrained, we also include two extra model sizes in this table, extra extra small (xxs) and extra small (xs), which allow us to collect more data points for the extrapolation of scaling trends. Because the hyperparameters used in these experiments were tuned for feed-forward Transformers in [85], we broadly expect that hyperparameters tuned for EBTs will further increase the performance gap between EBTs and the Transformer++ in autoregressive modeling. Future research should investigate the optimal width/depth/batch sizes for EBTs.

D.1 Autoregressive Language Modeling Experimental Details

D.1.1 Autoregressive Language Modeling Learning Scalability Details

Conducting thorough scaling experiments is extremely challenging—a recent survey on “scaling laws” [86] showed just how fragile many of these “scaling laws” are to hyperparameters, data, and other parameters, and how changing these variables slightly can lead to vastly different conclusions.

Being bottlenecked by a limited set of computing resources further exacerbates the issue of comparing two models. Therefore, we sought out to conduct controlled experiments that revealed the most information possible regarding the scaling of EBTs compared to different models.

Most existing works study scaling by varying several factors simultaneously, including depth (number of transformer blocks) [85], width (embedding dimension) [85], possibly batch size [105, 149], and the amount of data [85]. Therefore, to be more comprehensive in determining when EBTs scale differently than the Transformer++, we decided to conduct normal scaling experiments over all of these factors at the exact same time (as is standard), *as well as ablating over changing just one of these factors at a time*. Notably, conducting experiments in this manner allows for controlling a single independent variable at a time (i.e., just changing the number of Transformer Blocks), which allows for stronger conclusions regarding what aspects of model scaling different models perform better over (i.e., EBTs scale better than the Transformer++ when increasing the number of Transformer Blocks). Scaling all factors at once does not allow for such insight, as there are many independent variables changing at once. We believe experiments that involve changing a single independent variable are in broader alignment with traditional scientific principles.

For the parameter and FLOP scalability experiments, all models are pretrained for $105k$ steps. For each model size, following [105, 149], we scale the batch size. For NLP experiments we use the following batch sizes for the xxs, xs, small, medium, and large models respectively: 32, 46, 90, 170, and 256. These batch sizes were determined by scaling the batch size with the square root of the number of parameters and then rounding to an even number, similar to batch size trends in [149]. All model sizes use the same learning rate as in the Mamba paper [22] (0.0006, 0.0003, 0.00025, and 0.0002 for small, medium, large, and xl models respectively), where for the xxs and xs models we use a learning rate of 0.0012 and 0.0009 respectively. For the data scaling and batch size scaling experiments conducted in Figure 4, we use xxs models (we also report small models in Section B). The data scaling experiment used a batch size of 128 and the batch size experiments ran for $105k$ steps. All models were trained with a context length of 256 and no FFN multiplier (FFN dimension being equal to the embedding dimension) due to limited resources.

D.1.2 Autoregressive Language Modeling Thinking Scalability Details

We train xxs S2-EBT and Transformer++ models with the same setup as above, with the exception that models are trained with a batch size of 128 for $1M$ training steps (Table D.4 contains more hyperparameter details). Increasing the data scale enables us to better understand how thinking scales during pretraining. It’s worth noting that since we are training small language models, they could not benefit from modern techniques such as Chain of Thought (CoT) in improving performance. Figures 6, B.1, 7 and Table 3 use the best checkpoints of these two models. Figures 6a, 7 both use all four downstream datasets shown in Table 3. Figure 7 also uses the pretraining dataset in addition to these downstream datasets, where every dataset is represented as a separate OOD point.

Figure 6b was solely from the Dyck Languages benchmark. We did not observe this trend in other benchmarks, possibly due to perplexity not being a completely linear metric. Figure B.1a was on the RedPajamaV2 validation dataset.

D.2 Autoregressive Video Experimental Details

We use the same model parameter scaling, shown in Table D.1, as the NLP experiments. We also used a batch size of 256 for all models, as we found that it did not affect the scaling performance due to models training for many epochs. We also processed videos with 0.25 seconds between frames. For the Transformer++ baseline, we use the same learning rates as the NLP experiments. For EBTs, we found that it was necessary to use a lower learning rate by a factor of 3 for proper training stability. We use the standard SSV2 train and validation split for experiments. Other hyperparameters are shown in Figure D.2 and Figure D.3.

D.3 Bidirectional Image Denoising Experimental Details

We use the COCO 2014 dataset [119, 120] with 128 by 128 images, its train/validation split, a patch size of 16, and the Diffusion Transformer implementation from [26]. All models were trained using the large model size described in Table D.1, with a learning rate of $1e-4$ for 100,000 steps. For the DiT baseline, we used the same hyperparameters from [26], changing only the batch size to

Table D.3: Hyperparameters for EBT scaling experiments.

Hyperparameter	CV	NLP
Optimizer		AdamW
Optimizer Momentum		$\beta_1, \beta_2 = 0.9, 0.999$
LR Schedule		Linear warm up cosine decay
Warmup Steps		$1e4$
Minimum LR Scale		10
Gradient Clip Value		1
Weight Decay		0.01
Context Length	16	256
Encoder	SD-XL VAE [114, 115]	-
Image Dimension	224x224	-
Tokenizer	-	EleutherAI/gpt-neox-20b [148]
Vocab Size	-	50277
Normalize Input Distribution	-	✓
Optimization Steps	2	2
Optimization Step Size	30,000	500
Optimization Step Size LR Multiplier	90,000	1,500
Learnable Optimization Step Size		✓

128 from 256. We based our bidirectional EBT implementation on the code from this repository. We experimented with several different diffusion inference strategies to ensure a fair comparison, including DDPM, DDIM, increasing the number of diffusion steps at inference, and recursively applying the diffusion model on its own denoised output.¹³ Ultimately, we found that the combination of DDIM recursively applied on its own output performed best, hence we used this as the baseline in all experiments. We used the default denoising schedule from the DiT codebase [26].

For both DiTs and EBTs, we found that models performed best on OOD noise levels when denoising their own outputs twice, that is applying the model to denoise the image three times recursively. This is how we are able to achieve the results in Figure 12 demonstrating the performance for 300 forward passes from DiTs and 3 forward passes from EBTs. We found that for image denoising, it was not necessary to train EBTs with the S2 hyperparameters for System 2 Capabilities to emerge, although its possible these would further improve performance. Additionally, for image classification, for both models, we take the average of all the final patch tokens, and for DiTs we feed in $T = 0$.

D.4 Computational Resources

All experiments were conducted on either Nvidia A100s, H100s or GH200s, with the largest scale experiment requiring approximately ≈ 1300 A100 GPU Hours. The runtime for each experiment was dependent on the model sizes used as well as the amount of data trained on.

D.5 FLOP Calculations

We adopt the standard estimate of $6N$ FLOPs per token for the AR Transformer++ [150], where N denotes the number of non-embedding parameters. For AR EBTs, however, the per-token cost varies with the number of training optimization steps and chosen hyperparameters.

To derive the FLOPs for EBT training, we follow [76] for the Hessian-vector product (HVP), which EBTs require to backpropagate through a first-order derivative. Since an HVP has the same theoretical complexity as a gradient computation, we express the per-step FLOPs as

$$\text{FLOPs} = F + B + B.$$

Based on [150], the forward and backward passes require approximately $2N$ and $4N$ FLOPs per token, respectively, where N denotes the count of non-embedding parameters in the Transformer. In the autoregressive EBT implementation, the effective sequence length becomes twice that of the original Transformer (formally $2S - 2$ for an original sequence length S). Owing to the efficient scheme of Section C.3, this doubling of sequence length translates roughly into a two-fold increase

¹³This recursive application was only performed during testing due to the distribution shift.

Table D.4: **Hyperparameters for EBT System 2 Thinking experiments.**

Hyperparameter	NLP
Optimizer	AdamW
Optimizer Momentum (β_1, β_2)	0.9, 0.999
Model Size	Small
Learning Rate (LR)	0.0012
Learning Rate (LR) Schedule	Linear warm-up + cosine decay
Warmup Steps	1×10^4
Minimum LR Scale	10
Gradient Clip Value	1
Weight Decay	0.01
Context Length	256
Tokenizer	EleutherAI/gpt-neox-20b [148]
Vocab Size	50,277
Normalize Input Distribution	✓
Optimization Steps	2-3 (randomized)
Optimization Step Size	5
Optimization Step Size Multiplicative Random Factor	2
Langevin Dynamics Noise	3
Learnable Optimization Step Size	✗
No Detach Between Optimization Steps	✓
Truncate Optimization	✓
Replay Buffer	✓

in FLOPs, rather than a four-fold increase in FLOPs. Hence, each second-order optimization step demands roughly

$$(F + B + B) \times 2 = (2N + 4N + 4N) \times 2 = 10N \times 2,$$

making it $\approx 3.33\times$ more expensive than a standard feed-forward Transformer step.

The overall FLOP count also depends on one’s choice of hyperparameters. For S1 models, where the loss is evaluated at every iteration without gradient truncation, the total FLOPs simply multiply by the number of steps. Therefore, for our pretraining experiments using two optimization steps, we get that EBTs used $6.66\times$ the FLOPs of a comparable Transformer++ during training. In contrast, for S2 models, a random number of optimization steps is used, the gradient is truncated, the loss is only calculated at the last step following [48], and a Replay Buffer is used. Therefore, the FLOP count varies and can both decrease (as truncating uses less FLOPs for earlier steps) as well as increase (as using more steps and a replay buffer both use more FLOPs). These numbers also vary during inference, where the full EBT implementation parallelizing all predictions at once is not necessary.

Given the scarcity of published methods for computing higher-order derivative FLOPs and our inability to leverage existing libraries for Hessian-vector products, these estimates remain approximate. We welcome corrections or additional insights from readers familiar with FLOP calculations for second-order methods.

E Additional Related Works

E.1 Energy-Based Models (EBMs) as a Generalization of Diffusion Models and Recurrent Depth Models

Both diffusion models [133] and RNNs [127] can be seen as predicting the score, or the gradient of the energy function/data density, $\nabla_x E_\theta(x)$, where diffusion models do this with an additional time condition [151]. Thus, the largest benefit of explicit EBMs over these approaches, which can be seen as implicit EBMs (due to only implicitly defining an energy function), is that using an explicit EBM allows for explicit verification/likelihood modeling. We show that this enables the use of self-verification to improve predictions, whereas with diffusion models and RNNs an additional verifier model is necessary to achieve this capability [19].

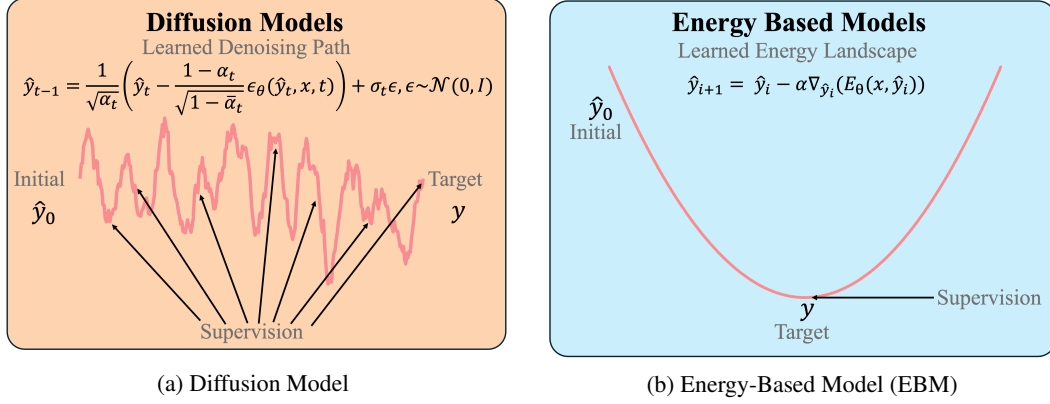


Figure E.1: EBM and Diffusion Comparison. Diffusion models receive supervision at each step of the denoising process (e.g., for one thousand steps), whereas EBMs only receive supervision at the end of the optimization process. This training procedure allows EBMs to learn an entire Energy Landscape over predictions, associating a scalar energy for every prediction according to its likelihood. Learning landscapes in this manner can reduce “error” accumulation throughout the denoising process [67] and makes EBMs more flexible by allowing unnormalized likelihood estimation at each step of the denoising process. Additionally, diffusion models update predictions by predicting the noise at each timestep, meaning they must follow a set denoising schedule. On the other hand, EBMs update predictions by performing gradient descent with respect to the energy scalar, allowing for flexible inference where this optimization process can be performed for any number of steps. x here refers to some condition (e.g., a class or text) whereas y is the generated prediction.

It’s also worth noting that RNN, diffusion models, and EBMs need not be incompatible with one another. For example, [67] combines EBMs and diffusion to reason over challenging problems. This can increase stability of the learned energy landscape by adding explicit score supervision.

E.2 In Depth EBM and Diffusion Model Comparison

Because of the similarity of diffusion models and EBMs, we present a side-by-side comparison of the training and inference approach for both in Figure E.1, where the primary difference lie in the supervision they receive during training and the update rule for predictions. Additionally, we provide more information to compare these approaches.

Under the assumption that the energy landscape is well formed and that optimization is well behaved, EBMs offer several distinct advantages over diffusion models. In EBMs, the energy function is trained to represent a meaningful landscape where the energy value of a sample directly corresponds to its relative unnormalized likelihood. Consequently, two samples can be directly compared to determine which is more likely, in a single forward pass. On the other hand, diffusion models require running samples through the entire reverse diffusion process to get likelihoods, which often requires hundreds to thousands of steps, and rely on likelihood approximations such as ELBOs or numerical solvers for SDEs/ODEs. In practice, these result in incomparable likelihoods, as ELBOs only give likelihood lower bounds and numerical solvers result in high approximation error [19].

Furthermore, the learning of an energy landscape over predictions means that any approximation errors at each individual step of the Markov Chain (optimization process) do not result in cumulative error, as the minimum of the energy landscape can still be reached. This differs from diffusion models, where any approximation error at each step will result in increasing accumulated error across the entire Markov Chain [67] (demonstrated in Figure E.1).

Lastly, EBMs, giving an unnormalized likelihood estimate at each step, are in practice much more flexible for generation than diffusion models. While diffusion models require running the entire reverse diffusion process with a specific denoising schedule to generate a sample, EBMs can be trained to directly predict the next sample in a single step, and give an unnormalized likelihood at each step indicating how likely they think this sample is. This better approximates human-like System

2 thinking, where humans naturally evaluate the strength of current predictions, and on the basis of knowing how good their predictions are, decide to dynamically allocate more or less computational resources.

E.3 Additional Energy-Based Models Related Works

One contribution of this work was the design of a custom architecture for EBMs called the Energy-Based Transformer (EBT). Roughly similar in naming is the work of the Energy Transformer [152]. However, despite similarity in the names of these architectures, they are very different—with the primary similarity in architectures being the usage of attention mechanisms as well as a global energy function. The existing work integrated ideas from Modern Hopfield Networks, including RNNs, whereas in our work the architecture is non-recurrent and does not use associative memories. Additionally, in this work with EBTs we focused on System 2 Thinking and scalability to high-dimensional problems, which the previous work did not experiment with.

Other loosely related approaches to EBTs involve autoregressive Energy-Based Models, including E-ARM [153], EBR [140], and Residual EBMs [139]. E-ARM involves adding an objective to the learning process to turn a traditional autoregressive model into an EBM, and as such does not achieve two of the cognitive facets discussed in Section 1. EBR and Residual EBMs involve the training of an EBM on top of an already pretrained autoregressive language model. Both works, however, leverage a contrastive objective, which suffers from the curse of dimensionality.

The optimization procedure used to train EBTs can be seen as a form of denoising score matching [71, 154]. Particularly, EBTs having predictions initialized at some Gaussian, and then optimizing predictions using the gradient of the energy function, can be seen as training EBMs to denoise by learning the score of the data starting from a Gaussian prior. However, we find the optimization perspective is more intuitive, and this denoising score matching perspective is more similar to the diffusion model training procedure than it is EBMs, involving multiple levels of noise rather than just one level.

F Additional Cognitive Facets

In this section, we describe in more detail the different facets of cognition, in addition to introducing additional facets.

Facet 1, Dynamic Allocation of Computation, can be formalized as Turing completeness (assuming an infinite length external memory). Because standard feed-forward Transformers have finite length programs (i.e., they have finite depth), they are unable to be Turing complete at the granularity of each prediction being made. This also holds for common RNN variants that are only updated with new sequence information. Diffusion models and EBMs, being able to iteratively compute functions, have potentially infinite length programs, and therefore when augmented with an infinite external memory can be Turing complete.

Facet 4: Compositional Reasoning and Systematicity. Humans routinely solve novel tasks by recombining familiar primitives (e.g., verbs with new arguments or visual parts into unseen objects), a hallmark of compositional generalization well-documented in neuroscience [155]. In contrast, state-of-the-art Transformers and diffusion models often fall short when evaluated on compositional generalization [156, 157]. Energy-Based Models (EBMs) seamlessly address these limitations: energies for individual factors are composable in several different manners [133], enabling zero-shot generation of novel combinations without retraining, where gradient-based sampling provides an intrinsic mechanism to verify and iteratively correct compositions [133]. Thus, EBMs offer a promising path toward human-like systematicity that remains elusive for existing paradigms and architectures.

G Counterarguments

G.1 System 2 Thinking

In this paper, strong claims were made regarding the capabilities of current models and their ability to perform System 2 Thinking. However, there are common counterarguments to our claims, which we address here.

G.1.1 System 2 Thinking and Inference-Time Compute Term Usage

Whether the computational effort spent at inference time fully captures what psychologists term System 2 Thinking is still actively debated. In Section C.1 we outline three reasons for preferring the broader terminology of System 2 Thinking: (i) it naturally extends to settings such as continual learning where terms such as “inference-time compute” becomes ambiguous, (ii) it connects our discussion to a substantial body of cognitive-science work, and (iii) it offers a conceptually straightforward entry point for readers beyond the machine-learning community.

We believe that Human System 2 Thinking encompasses a far greater depth and complexity than the specific approaches explored in this paper, such as “Thinking Longer” and “Self-Verification.” We wish to emphasize that our work does not claim current models replicate the full spectrum of human System 2 Thinking. Rather, we view the methods presented here as foundational steps toward that more ambitious long-term goal.

We propose that “System 2 Thinking” offers a useful umbrella term that can effectively encompass and generalize other existing terminologies, including “inference-time compute,” “test-time compute,” or “reasoning.” A parallel can be drawn with the term “learning” in our field. “Learning” itself has evolved to describe a wide array of processes, some of which, such as k-Nearest Neighbors (KNNs) or the specific mechanisms of weight matrix updates in Artificial Neural Networks (ANNs), represent distinct facets rather than the entirety of human-like learning, meaning these approaches may not encompass the complexity of true human-like learning. However, despite this breadth and these simplicities, “learning” has become a cornerstone term, upon which our entire field of machine learning is named upon.

In a similar vein, while the “thinking” exhibited by the models discussed in this paper may not yet capture the full nuance and intricacy of human cognition, we believe the term “System 2 Thinking” can still serve a valuable role. It offers a generalizing framework for existing vocabulary and can contribute to making complex concepts within the field more accessible and understandable to newcomers or experts from other fields. Our intention is to contribute to a constructive and unifying dialogue of intelligent systems.

G.1.2 Is Chain-of-Thought (CoT) Sufficient for System 2 Thinking?

CoT is commonly believed to be sufficient for advanced reasoning to emerge in LLMs. However, we argue that there are several flaws with CoT preventing advanced thinking capabilities. First, Chain-of-Thought (CoT) involves reasoning over a discrete state space, which limits the granularity of “thoughts.” Second, CoT is not an intrinsic architectural capability but an external procedure applied to token sequences. Ideally, such reasoning should be embedded within the model and learned during training (unsupervised training, particularly). Third, each token is produced with a fixed computational budget, restricting the depth of reasoning per step. In contrast, humans allocate variable effort across steps when reasoning “step by step”. Similarly, models should be able to spend a variable amount of computation per token, as enabled by EBTs. This aligns with the intuition behind the saying: “a chain is only as strong as its weakest link”—*each step in a chain of thought should receive sufficient computation to avoid failure points that result in bad reasoning.*

H Energy-Based Models (EBMs) Introduction

H.1 Simplified Energy-Based Model (EBM) Introduction

Feed-forward neural networks generally take the form of: given an x predict y (Fig. H.1a). Energy-Based Models (EBMs) are a family of models that learn the compatibility (unnormalized probability) over all possible combinations of x and y (Fig. H.1b). Intuitively, this can be seen as learning to verify the strength of y as a prediction with x as the input. Training models in this manner allows for representing multiple plausible versions of y compatible with a given x . The differences between these models is visualized in Fig. H.1.

Formulating models in this manner ultimately brings about two primary questions:

First question: Assuming we still care about ultimately predicting $\hat{y}$ *how do we use such an EBM to predict $\hat{y}$* ? With feed forward models, generally we can just input x and get the output of the model as $\hat{y}$, but we can't do this with EBMs?

With EBMs, what happens is conceptually similar to diffusion models [114], where we (commonly) initialize $\hat{y}$ as random noise. Then, we input x and $\hat{y}$ into the model, and get a single scalar energy output (our initial energy output) from the model. Now, because our entire model is differentiable, we can get the *gradient from this energy scalar to $\hat{y}$* and perform gradient descent along the *energy landscape* (energy landscapes are surfaces resulting from mapping all possible predictions to scalar values, visualized in Figure 3) using this gradient (this is the key)! This process is visualized in Figures 3, 2. This gradient can be seen as the opposite of the noise (e.g., denoising) and therefore EBMs have strong relations with Diffusion models predicting the noise. EBMs can be seen as a generalization of diffusion models, where diffusion models are predicting the gradient of the energy function/scalar (more on this in Section E).

Second question: *How do we train an EBM?* Generally, models are trained over a dataset of x and y pairs, but now there are several different possible y values that can be associated with any given x value—so how does that work?

It turns out that all the training techniques for EBMs boil down to two main categories: contrastive and regularizing approaches [42].

Contrastive approaches are more common for EBMs and are easier to rationalize about due to their similarity to GAN discriminators. The idea behind contrastive approaches is to push down on the energy of positive samples (i.e., the true data), and to push up on the energy of negative samples. While these positive samples are easy to rationalize about, as they are just the true data, the difficulty of contrastive EBMs is finding negative samples. Several approach exist, such as GANs, which use a generator to amortize negative sample generation, or running MCMC (similar to optimization) for some time. However, as discussed in Section 3.1, such approaches don't scale well due to the curse of dimensionality.

Therefore, to achieve a scalable EBM approach, we train EBMs through an optimization procedure (which has strong resemblance to Langevin Dynamics). That is, EBMs are trained to, starting from an initial prediction, optimize predictions to the ground truth solution (shown in Figures 3, 2). This pushes the energy landscape to be locally convex surrounding the ground truth solution, thereby regularizing the energy landscape to have low energy only on the true data. As mentioned in [71], this can be seen as being similar to denoising score matching [154].

Commonly, when people learn about EBMs and the iterative denoising/optimization procedure performed during training, they think of diffusion models, so we include a more in depth comparison between the two in Section E.2.

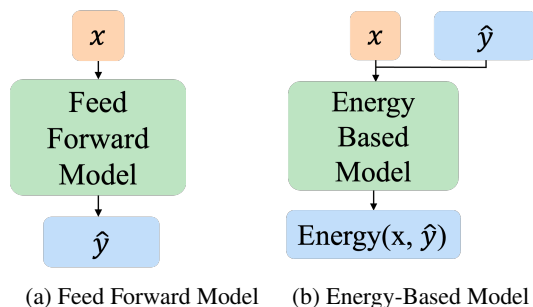


Figure H.1: Feed-Forward and Energy-Based Model Comparison. Feed-forward models (a), given an input x , directly try to predict $\hat{y}$. Instead of just getting x as an input, EBMs receive both x and $\hat{y}$ as an input and learn the **compatibility** of all possible values of $\hat{y}$ with x by outputting a scalar energy value for each combination. Low energy corresponds to high probability, and high energy to low probability. In practice, $\hat{y}$ is often initialized as random.

H.2 Energy-Based Model Types

Because EBMs are a broad modeling framework, and can generalize many existing approaches, we aim to provide precise language to distinguish EBM types. We broadly classify the EBMs described throughout this paper as **explicit EBMs**, meaning they explicitly define an energy function over inputs as the entire function being learned. In other words, explicit EBMs directly map all variables (inputs) to a single scalar energy as the output of the neural network. We define these in contrast to **implicit EBMs**, or EBMs where the energy function is not the learned model but rather some implicit definition of the learned model. For example, with diffusion models, the energy function is implicitly defined by the learned score network ($s_\theta(x, t)$) as the following:

$$\begin{aligned}\nabla_x E(x, t) &= -s_\theta(x, t) \\ E(x, t) &= -\int^x s_\theta(u, t) du + C(t).\end{aligned}$$

Other notable examples of implicit EBMs include Hopfield Networks [158], RNNs [25], and Boltzmann machines.

H.3 Energy-Based Model Frequently Asked Questions (FAQ)

- **What is energy/compatibility, what does it represent, and how is it learned? What energy corresponds to what probability?**

Energy is just a learned compatibility score between x and y (lower means more likely). The EBMs described in this paper learn it implicitly as described in Section 3.1 such that the true data (good pairs) have low energy and bad pairs (non-data) have higher energy. Probabilities follow:

$$\begin{aligned}p_\theta(x) &= \frac{e^{-E_\theta(x, y)}}{Z(\theta)} \\ p_\theta(x, y) &\propto e^{-E_\theta(x, y)}\end{aligned}$$

so the energy E is essentially the (unnormalized) negative log-likelihood up to an additive constant. The term compatibility is just a term used for intuition.

- **Does training EBMs require a full Hessian calculation?**

No—the approach described in the paper only requires Hessian-vector products. That makes training only about a constant $1.66\times$ as expensive as a vanilla feed-forward model given everything else remains constant and you use a single step.

- **Why is low energy good (and high energy bad)? Why not just use probability?**

Low energy is good because of the negative exponential. The reason we don't use probabilities is avoiding normalized probabilities makes the problem much more tractable in real-world high-dimensional continuous state spaces by removing the focus on explicit normalization via regularizing the partition function. EBMs come from a long line of work in statistical physics.

- **Is it okay that energies are unnormalized probabilities?**

Yes, for most real-world applications, you only ever need sample **relative likelihood** comparison; it's significantly less common to need the exact likelihood of samples. An example of this is reward models, which can be seen as EBMs (just multiplying the reward by -1), where all that really matters is the relative reward for choosing which sample to use or which behavior to perform.

- **Is it fine to not do Maximum Likelihood Training?**

Contrary to what your intuition may say, the answer is **yes!** For most real-world distributions, data lies completely concentrated on a very thin manifold with no defined distribution outside of this manifold. Thus, directly doing Maximum Likelihood Estimation (MLE) training would push EBMs to have low energy on the true data manifold and then infinite energy off that manifold (as the probability of such samples is 0). We don't want this as it would make the score (gradient of the energy function) undefined and the energy landscape untraversable—so not doing MLE makes the problem tractable.

I Energy-Based Transformers (EBTs) Tutorial

I.1 Improving Stability and Scalability

Energy-Based Models are notorious for instability during training [51–54]. Therefore, we experiment with several different hyperparameters to increase the stability and scalability of EBTs and EBMs in general.

I.1.1 Optimization Step Size and Stability

We found that the step size for gradient descent updates of predictions (α) was one of the primary factors affecting the stability of EBTs. Thus, for S1 models, we make the step size a learnable parameter (this is not the case for S2 models). We calculate its learning rate by multiplying the model’s learning rate by the step size learning rate multiplier. We found that the values for the step size have a large effect on the magnitude of gradients generated for the optimization of predictions. This is because the step size is directly multiplied by the prediction gradients. Particularly, a smaller step size results in larger generated gradients, whereas a larger step size results in smaller gradients. Therefore, the step size needs to be tuned per modality, as the update required for predictions depends on data. It’s also worth noting that we do not weight decay the step size in any of the models.

We also found that a relatively high optimization step size was necessary (30,000 for video and between 5 and 500 for text). Without a high optimization step size, gradient magnitudes continued to increase throughout training, resulting in unstable training dynamics.

I.1.2 Architecture Stability

For autoregressive models, we found that simply prepending a learnable “step” embedding to sequences significantly improved scalability and stability, especially for S1 models. This step embedding mapped a discrete step index (i.e., step 0, 1, etc.) of the current optimization step to an embedding the same dimension as the model’s embedding. We believe this helped improve stability by enabling the accumulation of attention mass, as well as enabling less steep energy landscapes conditioned on the optimization step.

Additionally, we experimented with adaptive layer normalization from the DiT architecture, but we found that the timestep embedding worked better. We also experimented with several different normalization and initialization approaches, where we found that the standard Llama2 [87] architecture and initialization worked best. This involves using RMSNorm, Xavier init [159], SwiGLU MLP [160], and RoPE [161].

I.1.3 System 2 Thinking Hyperparameter Stability

For S2 models, we found certain hyperparameters to be essential for the stability and scalability of models. First, we found that using a lower number of optimization steps resulted in more stability, as using more optimization steps necessitates longer gradient chains. Additionally, we found that the strategy used for the randomization of α to be very important for stability. Particularly, we found that randomizing α with a single value for an entire batch resulted in issues with training convergence. We believe this is because using the same value for every element in the batch resulted in high-variance gradients. Thus, when randomizing α differently for every batch and sequence element, we found training convergence to be more stable. It’s possible that randomizing the number of optimization steps would yield similar results in reducing gradient variance, however, we did not experiment with such a configuration.

I.1.4 Clamping Optimization Gradients for Stability

We found that clamping gradients of the energy function with respect to predictions (or prediction update gradients) could help improve training stability at the cost of some slight reductions in convergence speed. We do not conduct any experiments in the paper with clamped prediction gradients as we found that the other hyperparameters were sufficient for stable training, however, it’s a potentially useful trick worth discussing.

I.1.5 Normalizing Data for Stability

We found that normalizing/standardizing input data was crucial for the stability of EBTs. An example of this was in our NLP experiments, where we ran experiments with and without normalizing probability distributions. The experiments with unnormalized distributions often had extreme activations as well as large loss spikes, whereas the experiments with normalized distributions (by applying softmax) were stable.

I.2 How to Train Your EBT

The hyperparameters used for training EBTs are *extremely* important and can often be highly sensitive towards performance, so here we offer a guide toward hyperparameter tuning. First, we recommend starting off with training S1-EBTs, which are the easiest and most stable variant of EBTs not designed for System 2 Thinking. Then, once S1-EBTs can be trained successfully and scaled, we recommend changing the hyperparameters gradually towards the hyperparameters used for S2 models (we say gradually here as occasionally these parameters can cause instability and require additional tuning).

When training S1 EBTs, we recommend tuning hyperparameters in the following order:

- First, tune standard hyperparameters such as Learning Rate (LR), batch size, etc. Having a high batch size helps with stability by making gradients less noisy (because you initialize predictions from random noise this makes gradients noisier).
- Second, start tuning S1-specific hyperparameters—primarily alpha and its LR multiplier (we recommend keeping its LR multiplier around $3\times$ the value of alpha) and then tuning the number of optimization steps.
- Third, potentially tune whether the step size is learnable and try other EBT architectures (inspired by DiT [26] we tried a time embedding as well as adaptive layer normalization).

Once you have tuned these, the model should be stable and fine for most use cases. At which point, if you are desiring System 2 capabilities, you can proceed to the S2 models [I.3](#). Some potential metrics to monitor and look out for include the gradient magnitudes (if these increase too much or spike a lot that's a bad sign) and the gap between the initial and final energy after optimization (if this is too high or low it could be a sign your model's alpha value needs to be adjusted).

Following [71], we give pseudocode for training EBTs in natural language for language modeling (Listing 1) as well as in computer vision for autoregressive video modeling (Listing 2). The pseudocode is primarily for S2 models without any energy landscape regularization techniques. The first primary design decision in the presented pseudocode is whether or not to detach predictions in between steps. Not detaching predictions in between steps allows for more “Thinking Time” before making predictions, but makes the gradient computation graph longer and therefore increases the likelihood of stability issues with gradients. Similarly, calculating the loss at every step versus solely the last step enables model to “think for longer” before needing to make accurate predictions, and therefore affects the convexity and smoothness of the energy landscape. For S2 models, we found that **not** detaching between steps was best, and similarly that calculating the loss only at the last step was best. For S1 models, we found the opposite to be most stable. Generally, if one is calculating the loss only at the last step, then one should not detach between steps as it's best if the gradient propagates to previous steps in this case. For more details on these techniques, we refer the reader to the source code as well as Section [I.3](#).

```
# make sure to enable gradient tracking
with torch.set_grad_enabled(True):
    loss_fn = nn.CrossEntropyLoss(weight=None, ignore_index=
        tokenizer_pad_token_id)

    context_embeddings = self.embeddings(input_ids[:, :-1]) # B, S, D
    next_tokens = input_ids[:, 1:]
    next_embeddings = self.embeddings(next_tokens) # B, S, V; are just
        used for shaping next tensor
    predicted_distributions = torch.randn_like(next_embeddings) # B, S
        , V; initialize predictions as random
```

```

for _ in range(num_steps):
    # Can optionally detach predicted distributions so that no
    # gradient flows through past steps
    predicted_distributions = predicted_distributions.detach()

    predicted_embeddings = self.vocab_to_embed(torch.softmax(
        predicted_distributions)) # B, S, D; need to proj. to embed space
    # for transformer to work in, use linear layer, weighted sum, etc

    all_embeddings = torch.cat((context_embeddings,
        predicted_embeddings), dim=1) # B, 2S, D
    predicted_energies = self.transformer(all_embeddings) # B, S,
    1; this returns only energies for the predicted_embeddings

    # Compute the gradient of predicted energies w.r.t. predicted
    # distributions
    predicted_distributions_grad = torch.autograd.grad(
        predicted_energies.sum(),
        predicted_distributions,
        create_graph=True
    )[0] # B, S, V

    # Perform gradient descent w.r.t. the energy function where
    # self.alpha is the optimization step size
    predicted_distributions = predicted_distributions - self.alpha
    * predicted_distributions_grad

    # Calculate cce loss based on predicted and ground truth
    # distributions, optionally at each optimization step or only at the
    # end
    cce_loss = loss_fn(predicted_distributions, next_tokens)

```

Listing 1: Autoregressive Language Model Training Pseudocode in PyTorch

```

# make sure to enable gradient tracking
with torch.set_grad_enabled(True):
    loss_fn = torch.nn.SmoothL1Loss(beta=1.0) # use whichever loss
    # function desired

    context_embeddings = embeddings[:, :-1] # B, S, D
    next_embeddings = embeddings[:, 1:] # B, S, D
    predicted_embeddings = torch.randn_like(next_embeddings) # B, S, D
    ; initialize predictions as random

    for _ in range(num_steps):
        # Can optionally detach embeddings so that no gradient flows
        # through past steps
        predicted_embeddings = predicted_embeddings.detach()

        all_embeddings = torch.cat((context_embeddings,
            predicted_embeddings), dim=1) # B, 2S, D
        predicted_energies = self.transformer(all_embeddings) # this
        # returns only energies for the predicted_embeddings # B, S, 1

        # Compute the gradient of predicted energies w.r.t. predicted
        # embeddings
        predicted_embeddings_grad = torch.autograd.grad(
            predicted_energies.sum(),
            predicted_embeddings,
            create_graph=True
        )[0] # B, S, D

```

```

        # Perform gradient descent w.r.t. the energy function where
        self.alpha is the optimization step size
        predicted_embeddings = predicted_embeddings - self.alpha *
        predicted_embeddings_grad

    # Calculate reconstruction loss based on predicted and ground
    truth embeddings, optionally at each optimization step or only at
    the end
    reconstruction_loss = loss_fn(predicted_embeddings,
    next_embeddings)

```

Listing 2: Autoregressive Video Model Training Pseudocode in PyTorch

I.3 How to Think Using Your EBT

Once an S1 EBT has been trained, we recommend tuning the System 2 hyperparameters in the following manner:

- Remove detaching tensors between optimization steps and add loss truncation so the loss is only calculated at the final step.
- Then, tune alpha, as it's by far the most important EBT-specific hyperparameter. But, do not make it learnable.
- Next, tune the number of optimization steps, including potentially a minimum and maximum number when randomizing the number of steps.
- Then add a replay buffer, Langevin Dynamics, and eventually a randomized alpha (step size). Tune all of these in tandem while tweaking the earlier parameters (particularly alpha).

It's possible that randomizing the number of steps for each sample within a batch would work better (similar to randomizing the alpha value within a batch). Additionally, it's worth mentioning that all optimization steps are performed along the same energy landscape (same time embedding condition), unlike with S1-EBTs using multiple time steps.