Learning Iterative Reasoning through Energy Diffusion

Yilun Du [1 *] **Jiayuan Mao** [1 *] **Joshua Tenenbaum** [1]

Abstract

We introduce *iterative reasoning through energy diffusion* (IRED), a novel framework for learning to reason for a variety of tasks by formulating reasoning and decision-making problems with energy-based optimization. IRED learns energy functions to represent the constraints between input conditions and desired outputs. After training, IRED adapts the number of optimization steps during inference based on problem difficulty, enabling it to solve problems outside its training distribution — such as more complex Sudoku puzzles, matrix completion with large value magnitudes, and path finding in larger graphs. Key to our method's success is two novel techniques: learning a sequence of annealed energy landscapes for easier inference and a combination of score function and energy landscape supervision for faster and more stable training. Our experiments show that IRED outperforms existing methods in continuous-space reasoning, discrete-space reasoning, and planning tasks, particularly in more challenging scenarios. Code and visualizations are at https://energy-based-model.github.io/ired.

1. Introduction

Being able to solve complex reasoning tasks such as logic inference, mathematical proofs, and decision-making is one of the hallmarks of artificial intelligence. Researchers in various fields have been working on domain-specific algorithms for solving these tasks, typically utilizing various forms of search or optimization in iterative manners (*e.g.*, dynamic programming and gradient descent). These domain-specific algorithms are usually highly efficient and effective, but they usually can not directly handle sensory data and typically require users or experts to encode rules in domain-specific languages (such as axioms used in mathematical provers or domain theories in planning). Furthermore, it is usually hard for these systems to learn from experience to improve their performance on familiar tasks. A large body of work has been trying to address these limitations by incorporating machine learning in order to handle sensory inputs and learn to formulate and solve problems. Typical ideas include utilizing these domain-specific solvers as a submodule in a deep neural network (e.g., SAT solvers; Wang et al., 2019) or building structured neural networks that can realize algorithms (e.g., dynamic programming; Xu et al., 2019).

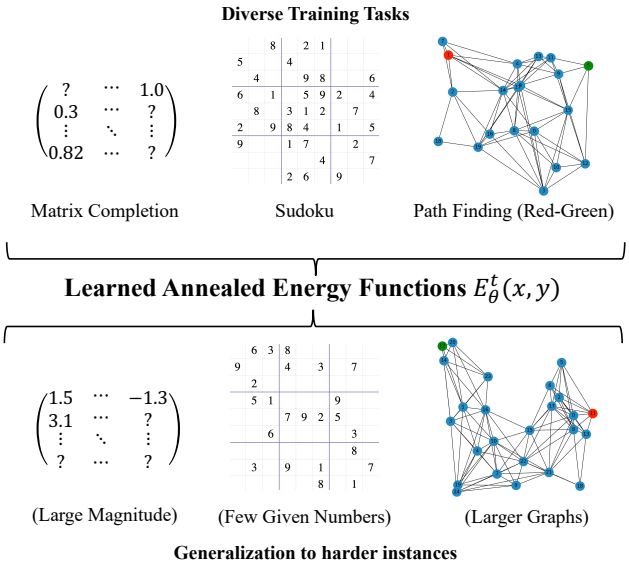

Figure 1. **Reasoning as Energy Diffusion** – IRED formulates reasoning problem with inputs $\boldsymbol{x}$ and output $\boldsymbol{y}$, as an energy minimization problem over a learned energy function. It can be trained stably for a wide variety of reasoning tasks and achieves strong generalization to harder problem instances, through adaptive computation in the optimization process.

Illustrated in Figure 1, we take a different approach to address the aforementioned challenges by formulating various kinds of reasoning and decision-making problems as an optimization problem. In particular, we consider the learning-to-reason problem as learning an energy function $E_\theta(\boldsymbol{x}, \boldsymbol{y})$ over input conditions $\boldsymbol{x}$ and desired output $\boldsymbol{y}$. For example, logical deduction can be cast as finding possible assignments to variables that satisfy all logical constraints; theorem proving can be cast as finding a sequence of valid

[*]Equal contribution [1]MIT. Correspondence to: Yilun Du <yilundu@mit.edu>, Jiayuan Mao <jiayuanm@mit.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

deduction steps that entails the goal; planning can be cast as finding a sequence of actions that respect the transition model of the environment and achieve the goal. This formulation directly allows us to learn the underlying constraints for a given task automatically from input-output data, without additional task-specific knowledge. Therefore, we can solve a wide variety of tasks across different domains using the same underlying training and inference paradigm, by only swapping out the neural network encoder for different data formats of $\boldsymbol{x}$ and $\boldsymbol{y}$. Another important feature of this optimization-based formulation is that during inference time, we can choose to apply a different amount of computation depending on the hardness of the problem by inspecting the value of the function $E_\theta(\boldsymbol{x}, \boldsymbol{y})$.

In particular, in this paper, we propose *iterative reasoning through energy diffusion* (IRED), a general framework for learning to reason. IRED is trained on a dataset of paired $(\boldsymbol{x}, \boldsymbol{y})$ data, and can recover the underlying energy function describing the objective function and constraints. During inference, because we are explicitly solving an optimization problem of finding the $\boldsymbol{y}^*$ that maximizes the energy function E_θ, we can run an adaptive number of optimization steps depending on the hardness of the problem. This enables us to solve problems that are beyond the training distribution, for example, Sudoku puzzles with a harder difficulty level, matrix manipulation under worse condition numbers, and sorting arrays with a larger size.

Our paper is not the first one to propose the use of energy-based models (EBMs) as a general framework for learning and reasoning (see, for example, Du et al., 2022). Although being a general framework for learning and reasoning, existing work falls short in its training speed, stability, and inference-time optimization hardness. These issues are critical and fundamentally hard because the learning of EBMs typically involves back-propagation through the entire iterative optimization process, and in general, the function landscape of E_θ can be complex with a large number of local optima. In this paper, we propose two important techniques to address these two challenges. Drawing inspiration from diffusion models and their relations to energy-based models (Ho et al., 2020; Du et al., 2023), instead of learning a single energy landscape, we instead learn a sequence of annealed energy landscapes, where smoother landscapes are being first optimized before optimizing for sharper ones afterward. Furthermore, in contrast to earlier work on EBM learning, IRED uses a combination of denoising supervision and direct supervision through negative sample mining. Both techniques can be implemented without the need to backpropagate through the optimization process, thereby making our learning algorithm both stable and fast.

We show the effectiveness of IRED on three groups of tasks: continuous-space reasoning (*e.g.*, matrix completion, inversion), discrete-space reasoning (*e.g.*, Sodoku solving, graph connectivity prediction), and planning (*e.g.*, finding paths on graphs). Compared with various domain-specific and domain-independent learning-to-reason baselines, including recurrent adaptive computation (Palm et al., 2018), EBM (Du et al., 2022) and diffusion-based models (Ho et al., 2020), IRED outperforms all of them, especially on test instances that are of higher difficulty levels, such as on matrices with larger value magnitudes, sudoku of higher difficulty levels, and larger graphs. Ablation studies show that the proposed optimization paradigm enables stable training and better generalization.

2. Related Work

Learning to reason with optimization. A wide variety of reasoning problems can be formulated as an optimization problem, including constraint satisfaction problems (CSPs), mathematical programs, discrete-space (Kautz et al., 2006) and continuous-space (*i.e.*, trajectory optimization, see Bryson, 2018, for a survey) optimization problems, and even algorithmic reasoning tasks (Brockett, 1991). The high-level idea is to cast these inference and decision-making problems as finding a set of variables that minimizes an objective function subject to constraints. Recently, there has been a growing interest in learning the objective and constraint functions instead of manually specifying them, which would be useful for domains where people do not have expert knowledge or simply the functions are too hard to be specified (*e.g.*, over high-dimensional sensory inputs).

Along this line, the first group of papers has explored using domain-specific optimization solvers as a computation block in neural networks. For example, Amos & Kolter (2017); Donti et al. (2017) integrates quadratic program solvers, Djolonga & Krause (2017); Wilder et al. (2019) studies submodular programs solvers, Wang et al. (2019) uses differentiable Max-SAT solvers, Yang et al. (2020) considers answer-set programming (ASP) solvers, Manhaeve et al. (2018) considers probabilistic logic programming solvers, and Rocktäschel & Riedel (2017) integrates symbolic theorem-proving solvers. However, due to the dependence on a particular problem formulation language, these frameworks are usually limited to solving problems of a particular kind.

The second group of papers has explored using a generic optimization framework as the underlying formulation. For example, Bai et al. (2019); Anil et al. (2022) utilizes equilibrium energy minimization inside a neural network to save memory, Rubanova et al. (2022); Comas et al. (2023) utilizes energy minimization to simulate physical dynamics by using neural networks to parameterize an energy function. Our paper falls into this group as well. Similar to our work, Du et al. (2022) uses energy-based models for learn-

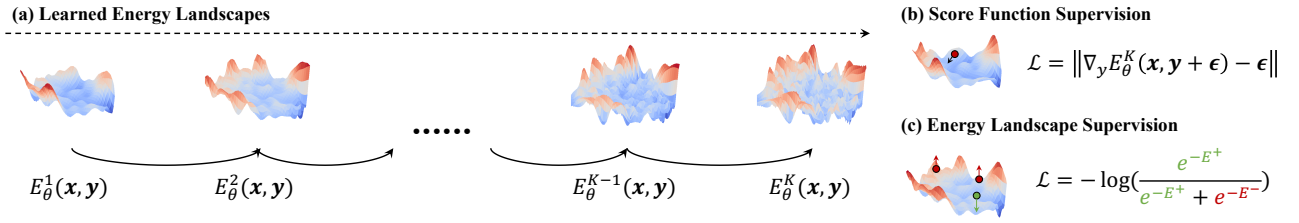

Figure 2. **IRED Learns a Sequence of Energy Landscapes.** During inference time, we optimize for $\boldsymbol{y}^*$ that minimizes the energy function, and we gradually increase the complexity of the energy optimization problem. The energy functions are trained with a combination of score function supervision and energy landscape supervision.

ing to reason. In contrast to it, this paper proposes to use a combination of denoising diffusion and supervised energy landscape training. This gives us stable training and strong performance. A concurrent work from Sun & Yang (2023) also considers using diffusion models for solving combinatorial optimization problems. However, their formulation is developed specifically for combinatorial optimization problems on graphs, but our diffusion formulation with energy parameterizations, landscape supervision, and substep optimizations can be generally applied to many optimization and decision-making domains.

Learning to reason with iterative neural computation. Another popular line of research studies using neural networks with iterative computation for reasoning. They draw motivation from the fact that many domain-specific constraint solvers are indeed iterative optimization algorithms (*e.g.*, gradient descent). At a high level, there are two groups of work: leveraging explicit program representations (Graves et al., 2014; Neelakantan et al., 2015; Reed & De Freitas, 2016; Cai et al., 2017; Chen et al., 2020b; Banino et al., 2021, ; typically with external memories) and using recurrent neural networks (Graves, 2016; Kaiser & Sutskever, 2016; Chung et al., 2017; Bolukbasi et al., 2017; Yang et al., 2017; Dong et al., 2019; Dehghani et al., 2019; Schwarzschild et al., 2021; Yang et al., 2023a). One of the key challenges in both types of approaches is when to halt the computation. Researchers have been tackling this problem through reinforcement learning (Chen et al., 2020a; Chung et al., 2017), leveraging hierarchical decomposition of programs (Cai et al., 2017), heuristic policies (Bolukbasi et al., 2017), and variational inference (Banino et al., 2021). However, these approaches are usually unstable, and many of them require manual hyper-parameter tuning (Banino et al., 2021) or additional human annotations (Cai et al., 2017). In this paper, we focus on an orthogonal approach by solving a broad set of reasoning problems by casting them as an optimization on learned energy landscapes. During optimization, the energy function of the landscapes naturally acts as a termination criterion.

Energy-based models and diffusion models. Our work is related to past work formulating prediction using Energy-Based Models (EBMs) (LeCun et al., 2006). Most recent EBMs have focused on learning probabilistic models over data (Xie et al., 2016; 2018; Du & Mordatch, 2019; Grathwohl et al., 2020; Du et al., 2021; Arbel et al., 2021; Xiao et al., 2020) but most similar to our work (Du et al., 2022) focuses on using energy minimization to solve reasoning tasks. Our work leverages the connection of energy based models and diffusion models (Du et al., 2023) to more effectively learn energy landscapes for solving reasoning problems.

An important difference between our proposed approach and standard diffusion models is that diffusion models usually focus on learning a particular sampling path transitioning from Gaussian noise to a target solution, where individual transition kernels across timesteps are learned. However, when obtaining a solution using these transition kernels, errors often accumulate across sampling timesteps, preventing a diffusion model from obtaining a precise answer to a reasoning problem. By contrast, we formulate predicting solutions as optimizing an annealed sequence of energy landscapes. In this setting, multiple steps of optimization are run at each energy landscape to ensure that we are at an energy minima at every landscape. These multiple steps of optimization prevent the accumulation of errors from using transition kernels in diffusion models, as they project the sample to an energy minima, which is likely "in distribution" to what has been seen during training.

3. Learning Iterative Reasoning through Energy Optimization

Let $\mathcal{D} = \{X, Y\}$ be a dataset for a reasoning task consisting of inputs $\boldsymbol{x} \in \mathbb{R}^O$ and corresponding solutions $\boldsymbol{y} \in \mathbb{R}^M$. We aim to learn a neural network-based prediction model $\text{NN}_\theta(\cdot)$ which can generalize execution $\text{NN}_\theta(\boldsymbol{x}')$ to a test distribution $\boldsymbol{x}' \in \mathbb{R}^{O'}$, where $\boldsymbol{x}'$ can be significantly larger and more challenging than the training data $\boldsymbol{x} \in X$ (*e.g.*, of higher dimensions, or with larger number magnitudes), by leveraging a possibly increased computational budget.

We formulate this adaptive model as an iterative energy optimization in Section 3.1. Our overall framework is illustrated in Figure 2. In particular, we construct an annealed sequence of energy functions to improve optimization. To involve

training stability, speed, and performance, we propose to shape the energy landscape to correctly assign minimal energy to ground truth solutions. We provide full pseudocode for training our approach in Section 3.4 with training following Algorithm 1 and inference following Algorithm 2.

3.1. Reasoning as Annealed Energy Minimization

A wide variety of reasoning and decision making problems can be formulated as an optimization problem. Traditionally, researchers have been focused on designing various domain-specific algorithms for solving different problems, typically with search, gradient-based optimization, or other forms of iterative computation, and also integrating machine learning to help. In this work, we take a different approach of formulating various kinds of reasoning and decision-making problems as an optimization process over a learned energy-based model (EBM): $E_\theta(\boldsymbol{x}, \boldsymbol{y}) : \mathbb{R}^O \times \mathbb{R}^M \rightarrow \mathbb{R}$. Under this formulation, the final prediction problem can be cast as finding the solution $\boldsymbol{y}$ according to:

$$\boldsymbol{y} = \arg\min_{\boldsymbol{y}} E_\theta(\boldsymbol{x}, \boldsymbol{y}). \tag{1}$$

One can use gradient descent to find such solutions:

$$\boldsymbol{y}^t = \boldsymbol{y}^{t-1} - \lambda \nabla_{\boldsymbol{y}} E_\theta(\boldsymbol{x}, \boldsymbol{y}^{t-1}), \tag{2}$$

where λ is the step size for optimization and the initial prediction $\boldsymbol{y}^0$ is initialized from a fixed noise distribution (*i.e.*, Gaussian throughout the paper). The final output of $\boldsymbol{y}^T$ is obtained after T steps of optimization.

In earlier work using a similar formulation Du et al. (2022), such EBM $E_\theta(\boldsymbol{x}, \boldsymbol{y})$ is trained by differentiating through the T steps of optimization and minimizing the MSE with the ground truth label $\boldsymbol{y}$ $\mathcal{L}_{\text{Opt}}(\theta) = \|\boldsymbol{y}_i^T - \boldsymbol{y}_i\|^2$. This approach requires the forward and backward computation of K steps of optimization at training, which makes it slow and unstable. Furthermore, because the EBM E_θ may have a complex optimization landscape[1], robustly finding solutions to Equation 2 is fundamentally hard.

As a general solution to stable training and better test-time optimization, instead of directly learning $E_\theta(\boldsymbol{x}, \boldsymbol{y})$, at a high-level, we propose to learn a sequence of annealed energy functions E_θ^k ($k = 0, 1, \cdots, K$), and supervise the EBM learning with the gradient of the energy function:

$$\mathcal{L}_{\text{MSE}}(\theta) = \|\nabla_{\boldsymbol{y}} E_\theta(\boldsymbol{x}, \boldsymbol{y} + \epsilon) - \epsilon\|^2, \quad \epsilon \sim \mathcal{N}(0, 1). \tag{3}$$

During training time, we obtain the ground truth for $\boldsymbol{y}$ by generating a noise-corrupted label $\boldsymbol{y} + \epsilon$, following a schedule of noise corruptions. By supervising on the gradient, our approach is substantially faster and more stable than earlier works using plain EBMs (Du et al., 2022) as it only supervises training of a single step of the optimization.

[1]For example, the 3-SAT problem exhibits steep energy minima surrounded by flat energy landscapes.

3.2. Learning Sequence of Annealed Energy Landscapes

Our key idea to mitigate the hard optimization problem of Equation 2 is to use simulated annealing — where smoother energy landscapes are first optimized before optimizing sharper ones afterward, as illustrated in Figure 2.

Similar to diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020), we propose to optimize and learn an annealed sequence of energy landscapes, with earlier energy landscapes being smoother to optimize and the latter ones more difficult. Given a ground truth label $\boldsymbol{y}$, we learn a sequence of K energy functions[2] $E_\theta^k(\boldsymbol{x}, \boldsymbol{y})$ over the ground truth label distribution $p(\boldsymbol{y}^*|x)$, where each energy function is learned to represent an EBM distribution

$$e^{-E_\theta^k(\boldsymbol{x}, \boldsymbol{y})} \propto \int_{\boldsymbol{y}^*} p(\boldsymbol{y}^*|x) \cdot \mathcal{N}(\boldsymbol{y}; \sqrt{1-\sigma_k^2}\boldsymbol{y}^*, \sigma_k^2 \mathbf{I}) \tag{4}$$

over a sequence of noise scales σ_k. Here, $\mathcal{N}(\cdot|\mu, \sigma)$ is the Gaussian density function. Larger values of σ_k correspond to smoother energy landscapes while smaller values lead to sharper landscapes, with the energy minima of landscape k corresponding to $\sqrt{1-\sigma_k^2}\boldsymbol{y}^*$ (which can be scaled by $\frac{1}{\sqrt{1-\sigma_k^2}}$ to obtain the ground truth prediction $\boldsymbol{y}^*$).

We can directly learn each energy landscape by supervising the gradient of energy function to denoise the corrupted ground truth label $\boldsymbol{y}^*$ from the dataset

$$\mathcal{L}_{\text{MSE}}(\theta) = \|\nabla_{\boldsymbol{y}} E_\theta(\boldsymbol{x}, \sqrt{1-\sigma_k^2}\boldsymbol{y}^* + \sigma_k \epsilon; k) - \epsilon\|^2, \tag{5}$$

where $\epsilon \sim \mathcal{N}(0, 1)$. Given a set of K different learned energy landscapes, we can initialize a data sample from Gaussian noise and sequentially run T steps of optimization following Equation 2 over each energy landscape k (starting with high noise levels and progressing to lower noise levels). The optimization result in the previous energy landscape is used to initialize optimization in the next landscape, after scaled by the appropriate scaling factor $\frac{\sqrt{1-\sigma_k^2}}{\sqrt{1-\sigma_{k-1}^2}}$.

3.3. Shaping the Energy Landscape

In the denoising training objective Equation 5, while the gradient of the energy landscape is locally trained to restore the ground truth label $\boldsymbol{y}$, it is not necessarily the case that the overall global energy minima $\arg\min_{\boldsymbol{y}} E_\theta(\boldsymbol{x}, \boldsymbol{y}, k)$ corresponds to the ground truth label $\sqrt{1-\sigma_k^2}\boldsymbol{y}^*$.

To enforce that the global energy minima of each of the k energy landscapes corresponds to the ground truth energy minima, we further propose a contrastive loss, where

[2]Empirically, in our experiments, we found that setting $K = 10$ was sufficient across all the domains we considered. With a total of 10 energy landscapes, we can smoothly transition from a Gaussian-like landscape (with $K = 10$) to a sharp and discontinuous landscape (with $K = 1$).

Algorithm 1 IRED Training

Input: Problem Dist $p_D(\boldsymbol{x}, \boldsymbol{y})$, EBM $E_\theta(\cdot)$, Noise Schedules $\{\sigma_k\}$, Corruption Function $c(\cdot)$, Landscapes k.

while not converged **do**

▷ *Supervise the Energy Landscape through Denoising:*

$\boldsymbol{x}_i, \boldsymbol{y}_i \sim p_D, \epsilon \sim \mathcal{N}(0,1), k \sim \{1, \ldots, K\}$

$\tilde{\boldsymbol{y}}_i \leftarrow \sqrt{1-\sigma_k^2}\boldsymbol{y}_i + \sigma_k \epsilon$

$\mathcal{L}_{\text{MSE}} \leftarrow \|\nabla_y E_\theta(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i, k) - \epsilon\|^2$

▷ *Shape the Energy Landscape Contrastively:*

$\boldsymbol{y}_i^- \leftarrow c(\boldsymbol{y}_i)$

$\tilde{\boldsymbol{y}}_i{}^- \leftarrow \sqrt{1-\sigma_k^2}\boldsymbol{y}_i^- + \sigma_k \epsilon$

$E_i^+ \leftarrow E_\theta(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i, k)$; $E_i^- \leftarrow E_\theta(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i^-, k)$

$\mathcal{L}_{\text{Contrast}} \leftarrow -\log\left(\frac{e^{-E_i^+}}{e^{-E_i^+}+e^{-E_i^-}}\right)$

▷ *Optimize objective $\mathcal{L}_{MSE} + \mathcal{L}_{Contrast}$ wrt θ:*

$\Delta\theta \leftarrow \nabla_\theta(\mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{Contrast}})$

Update θ based on $\Delta\theta$ using Adam optimizer

end while

Algorithm 2 IRED prediction algorithm

Input: Input task $\boldsymbol{x}_i$, Step Sizes λ_k, Number of Landscapes K, EBM $E_\theta(\cdot)$, Optimization Steps T.

$\tilde{\boldsymbol{y}}_i \sim \mathcal{N}(0,1)$

for each landscape $k = 1$ to K **do**

for run T steps of optimization $t = 1$ to T **do**

▷ *Optimize candidate solution $\tilde{\boldsymbol{y}}_i$ with gradient:*

$\tilde{\boldsymbol{y}}_i' \leftarrow \tilde{\boldsymbol{y}}_i - \lambda_k \nabla_{\boldsymbol{y}} E_\theta(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i, k)$

▷ *Check if the gradient descent step decreases energy:*

if $E_\theta(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i, k) > E_\theta(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i', k)$ **then**

$\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i'$

end if

end for

▷ *Scale optimized candidate solution:*

$\tilde{\boldsymbol{y}}_i \leftarrow \frac{\sqrt{1-\sigma_k^2}}{\sqrt{1-\sigma_{k-1}^2}}\tilde{\boldsymbol{y}}_i$

end for

return $\boldsymbol{y} = \tilde{\boldsymbol{y}}_i$

we construct a set of negative label $\boldsymbol{y}^-$ (formed by noise corrupting the ground truth label $\boldsymbol{y}^*$). Given an energy $E^+ = E_\theta(\boldsymbol{x}, \sqrt{1-\sigma_k^2}\boldsymbol{y}^* + \sigma_k\epsilon; k)$ of the ground truth label $\boldsymbol{y}^*$ and an energy $E_- = E_\theta(\boldsymbol{x}, \sqrt{1-\sigma_k^2}\boldsymbol{y}^- + \sigma_k\epsilon; k)$ of the negative label $\boldsymbol{y}^-$, $\mathcal{L}_{\text{Contrast}}(\theta) = -\log\left(\frac{e^{-E^+}}{e^{-E^+}+e^{-E^-}}\right)$. To reduce the variance of the contrastive loss, we use the same sampled noise value ϵ for both $\boldsymbol{y}$ and $\boldsymbol{y}^-$.

3.4. Combined Training and Inference Paradigms

We provide the overall pseudocode for training IRED in Algorithm 1 and executing algorithmic reasoning with IRED in Algorithm 2. We use a cosine beta schedule to train annealed energy landscapes and use a total of 10 energy landscapes (we empirically found that more landscapes did not lead to improved performance). At inference time, we can vary the number of optimization steps T for each energy landscape to make trade-offs between performances and inference speed.

In principle, when the solution is not well-defined, it is possible to use IRED to model multi-modal distributions, similar to how diffusion models have been proven effective in modeling multi-modal image distributions. Depending on the particular use case, one may also add additional inference-time constraints (*e.g.*, by composing the learned IRED energy function with other energy functions) to select favorable solutions.

4. Experiments

We compare IRED with both domain-specific and domain-independent baselines on three domains: continuous algorithmic reasoning, discrete-space reasoning, and planning. As we will break down in the following sections, the main advantages of IRED are twofold. First, compared with energy-based models (IREM), it is faster to train since it does not require backpropagation through the optimization process. Second, in terms of task performance, our focus is on generalization to "harder" problems, particularly leveraging the contrastive energy supervision and runtime iterative refinements. The idea is that after learning a correct energy landscape, the model can adaptively use more computation at test time to directly generalize to harder problems: we will focus on evaluating this generalization across all domains.

4.1. Continuous Algorithmic Reasoning

Setup. We first evaluate IRED on a set of continuous algorithmic reasoning tasks from Du et al. (2022). We consider three matrix operations on 20×20 matrices, which are encoded 400-dimensional vectors:

1. *Addition*: We first evaluate neural networks in their ability to add matrices together (element-wise). We also evaluate neural network on harder variants of the addition problems at test time by feeding input vectors with larger magnitudes.
2. *Matrix Completion*: Next, we evaluate neural networks on their ability to do low-rank matrix completion. We mask out 50% of the entries of a low-rank input matrix constructed two separate rank 10 matrices U and V, and train networks to reconstruct the original input matrix. We construct harder variants of the matrix completion problem at test time by increasing the magnitude of values in U and V.

Task	Method	Same Difficulty	Harder Difficulty
Addition	Feedforward	0.0448	0.7029
	Recurrent	0.3610	2.6133
	Programmatic	0.0111	0.3446
	Diffusion	0.0071	0.5931
	IREM	0.0003	0.0021
	IRED (ours)	**0.0002**	**0.0020**
Matrix Completion	Feedforward	0.0203	0.2720
	Recurrent	0.0266	0.3285
	Programmatic	0.0203	0.2637
	Diffusion	0.0219	0.2142
	IREM	0.0183	0.2074
	IRED (Ours)	**0.0174**	**0.2054**
Matrix Inverse	Feedforward	0.0112	0.2150
	Recurrent	0.0109	0.2123
	Programmatic	0.0124	0.2209
	Diffusion	0.0115	0.2132
	IREM	0.0108	0.2083
	IRED (Ours)	**0.0095**	**0.2063**

Table 1. **Continuous Algorithmic Reasoning.** Test evaluation performance on continuous algorithmic tasks. Inputs and outputs are 20 by 20 matrices. Error is reported using elementwise mean square error. Models are evaluated on test problems drawn from the training distribution (same difficulty) and a harder test distribution (harder difficulty). IRED outperforms comparisons.

3. *Matrix Inverse*: Finally, we evaluate neural networks on their ability to compute matrix inverses. We construct harder matrix inverse problems by considering less well-conditioned input matrices.

We report the underlying mean-squared error (MSE) between the predictions and the associated ground truth outputs on test problem instances. To more effectively generate negative samples for IRED in this domain, we first noise-corrupt ground truth labels and then run two steps of gradient optimization on the energy landscape to form negative samples. Details can be found in Appendix A.

Baselines. We compare our approach to a set of iterative reasoning baselines found in (Du et al., 2022): *(Feedforward):* an iterative reasoning approach where the same MLP is repeatedly applied, *(Recurrent):* an iterative reasoning approach where the recurrent network is repeatedly applied, and *(Programmatic):* an iterative reasoning approach which repeatedly applies a learned programmatic module (Banino et al., 2021). We further compare with the IREM method (Du et al., 2022), as well as using a denoising diffusion model directly to solve continuous tasks. All methods use identical architectures (with small differences due to recurrent layers or timestep conditioning).

Quantitative Results. We compare IRED with baselines across settings in Table 1. Similar to IREM, IRED is able to nearly perfectly solve the task of the addition, as well as generalize to larger addition matrices. On other tasks,

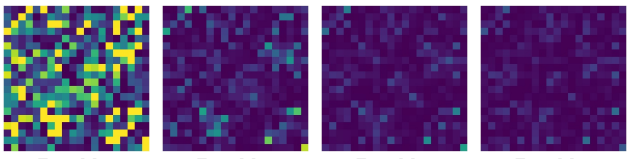

Figure 3. **Optimized Solutions Across Landscapes** – Error maps of intermediate optimized solutions. Optimized solutions at earlier landscapes are less accurate than later ones.

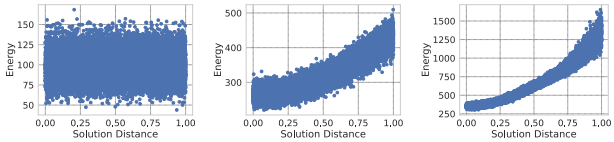

Figure 4. **Energy Landscape** – Predicted energy values for $\boldsymbol{y}$ and the corresponding MSE distance of $\boldsymbol{y}$ from the problem solution across different landscapes on the matrix inverse task. The earlier energy landscapes are smoother than the later ones.

IRED outperforms IREM and also generalizes better to harder problems. Furthermore, our approach substantially outperforms directly using a diffusion process to predict solutions, which lacks the iterative energy minimization procedure that explicitly learns the task constraints.

Qualitative Visualization. We provide a qualitative visualization of the error map of the optimized solution on the matrix inverse task at each different learned energy landscape in Figure 3. The error of optimized solutions at different energy landscapes decreases over time.

Energy Landscape. We visualize the learned energy landscape in Figure 4 as a function of the distance of an input label from the ground truth label. In early energy landscapes, the difference between energy values of solutions close and far from the ground truth solution is low, and therefore the energy landscape is relatively flat. At later landscapes, the energy value increases substantially as the input solution deviates from the ground truth solution.

Performance with Increased Computation. We analyze the performance on the matrix inverse task as a factor of an increased number of computational steps in Table 2. We find that running additional steps of optimization slightly improves performance on in-distribution tasks and substantially improves performance on harder problems.

Ablation. We ablate each component of IRED in Table 3. In the first two rows of Table 3, we compare our gradient-descent-based optimization with a noisy optimization procedure corresponding to the diffusion reverse process for each energy landscape. In the third row, we compare the difference between running multiple steps of optimization as opposed to a single energy optimization step. Finally, We then consider the effect of contrastively shaping the energy landscape. All components lead to improved performance.

Opt. Steps	Same Difficulty	Harder Difficulty
10	0.0096	0.2110
20	0.0096	0.2100
30	0.0096	0.2090
40	0.0095	0.2063

Table 2. **Continuous-Space Reasoning Performance vs Reasoning Steps.** More reasoning steps in IRED at inference time substantially improve generalization to harder difficulty tasks on the matrix inverse task. IRED is trained with 10 energy landscapes.

Gradient Descent	Optimization Refinement	Contrastive Shaping	Same Difficulty	Harder Difficulty
No	No	No	0.0158	0.2223
Yes	No	No	0.0097	0.2135
Yes	Yes	No	0.0097	0.2113
Yes	Yes	Yes	0.0095	0.2063

Table 3. **Continuous Ablations –** Ablations of proposed components of IRED on performance on the matrix inverse task. Leveraging gradient descent to optimize energy functions, using multiple steps of optimization at each energy landscape and contrastively shaping the energy landscape with ground truth labels all improve the performance on the Inverse task.

4.2. Discrete-Space Reasoning

Setup. The second group of tasks evaluates IRED on its reasoning in discrete spaces (*i.e.*, values are all binary or one-hot categorical). We run evaluations on two tasks: Sudoku solving and graph connectivity reasoning.

1. *Sudoku*: In the Sudoku game, the model is given a partially filled Sudoku board, with 0's filled-in entries that are currently unknown. The task is to predict a valid solution that jointly satisfies the Sodoku rules and that is consistent with the given numbers. We use the dataset from SAT-Net (Wang et al., 2019) as the training and standard test dataset. In SAT-Net, the number of given numbers is within the range of $[31, 42]$. Our harder dataset is from RRN (Palm et al., 2018), which is a different Sudoku dataset where the number of given numbers is within $[17, 34]$. We will show that our system, being trained on simpler Sudoku games with fewer blank entries, generalizes to harder instances.
2. *Connectivity*: In the graph connectivity task, the model is given the adjacency matrix of a graph (1 if there is an edge directly connecting two nodes). The task is to predict the connectivity matrix of the graph (1 if there is a path connecting two nodes). In literature (Dong et al., 2019), this task is an example that requires a dynamic number of reasoning "steps" (depending on the treewidth of the graph). Therefore, prior papers primarily focus on computing connectivity between nodes within k-steps away. Our training and standard test sets contain graphs with at most 12 nodes and our harder dataset contains graphs with 18 nodes. Since we do not limit the path lengths between nodes (in contrast to Dong et al. (2019)), on the training set, the maximum distance between two connected nodes is 9, and in the harder test set, the maximum distance is 16.

Task	Method	Test Dataset	Harder Dataset
Sudoku	IREM	93.5%	24.6%
	Diffusion	66.1%	10.3%
	SAT-Net	98.3%	3.2%
	RRN	**99.8%**	28.6%
	IRED (ours)	99.4%	**62.1%**
Visual Sudoku	SAT-Net	63.2%	0.0%
	RRN	**99.8%**	28.6%
	IRED (ours)	98.3%	**46.6%**
Connectivity	IREM	94.3%	89.8%
	Diffusion	61.6%	61.3%
	IRED (ours)	**99.1%**	**93.8%**

Table 4. **Discrete Reasoning Performance.** We evaluate models on the Sudoku task and the connectivity task. Sudoku:" the harder dataset has between 17 to 34 entries given, while models are trained with 31 to 42 entries given. Connectivity: the harder graphs have at most 18 nodes while the training graphs have only 12.

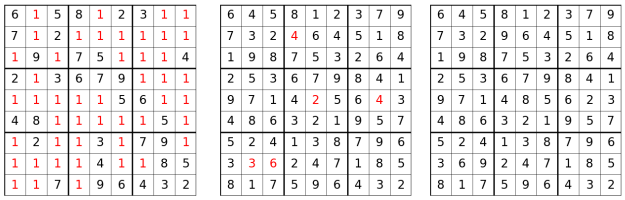

Figure 5. **Optimized Boards Across Landscapes** – Plot of the minimal energy board across energy landscapes, given the same initial board. Later energy landscapes lead to more accurate boards. We highlight inconsistent entries in red.

Quantitative Results. We compare IRED with both IREM and diffusion baselines. On the Sudoku task, we further compare with the domain-specific SAT-Net method. In Table 4, we find that our approach substantially outperforms all baselines across both evaluated discrete-space reasoning settings. In Sudoku, our approach generalizes substantially better than the SAT-Net model and the RRN model to the harder dataset consisting of fewer given Sudoku elements and is capable of obtaining an accuracy of roughly 62.1% compared to an accuracy of 3.2% obtained by SAT-Net and 28.6% by RRN. For cases in which our approach fails, we found that our approach sometimes erroneously assigns low energy to partially accurate answers. For example, there can be a Sudoku board that is not fully valid, but the model assigns lower energy to it than to the ground truth board.

Qualitative Results. We qualitatively visualize intermediate optimized samples across energy landscapes on Sudoku in Figure 6. Optimized boards are increasingly accurate in later landscapes.

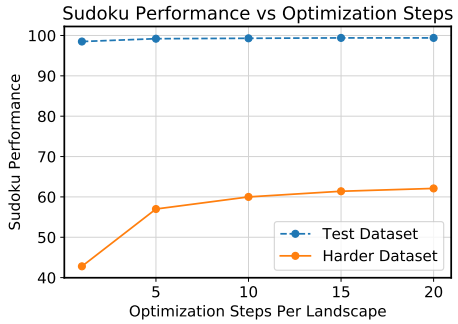

Figure 6. **Sudoku Performance with Optimization Steps** – Generalization to harder Sudoku problems significantly improves with a larger number of optimization steps in Sudoku.

Gradient Descent	Optimization Refinement	Contrastive Shaping	Same Difficulty	Harder Difficulty
No	No	No	97.0%	35.0%
Yes	No	No	98.8%	45.1%
Yes	Yes	No	99.3%	59.7%
Yes	Yes	Yes	99.4%	62.1%

Table 5. **Discrete Reasoning Ablations –** Ablations of proposed components of IRED on performance on the Sudoku task. Leveraging gradient descent to optimize energy functions, using multiple steps of optimization at each energy landscape and contrastively shaping the energy landscape with ground truth labels all improve the performance.

Performance with Computation. In Figure 6, we assess the impact of the number of optimization steps at each energy landscape on the performance in Sudoku on both the test and harder datasets. We find that performance substantially improves on the harder dataset with an increased number of optimization steps with more modest improvement on the test datasets. By formulating reasoning as energy optimization, we can adaptively change the number of optimization steps dependent on difficulty of task, enabling us to generalize substantially better on harder tasks.

Extension to Visual Sudoku. IRED can also be extended to deal with other input formats, such as images. To illustrate this, we conducte a new experiment on the Visual Sudoku dataset (Wang et al., 2019), where the board is not represented by one-hot vectors but now consists of MNIST digits written on a grid. We use a CNN to encode the image and fuse the image embeddings with the noisy answer to predict energy values. Shown in Table 4, we observed a similar performance advantage of our model compared to the baseline.

Ablation. In Table 5, we ablate each component of IRED on the Sudoku task. Similar to the continuous setting, we find that each component of IRED, gradient based optimization, multiple steps of optimization, and contrastive energy shaping all lead to improved performance. While performance is modestly improved on the test dataset, it is substantially improved on the harder generalization dataset with each added component.

Task	Method	Test Dataset	Harder Dataset
Shortest Path	IREM	90.4%	88.4%
	Diffusion	45.2%	46.9%
	IRED (ours)	**92.6%**	**91.9%**

Table 6. **Planning Performance.** Test evaluation performance on the shortest path task. The harder tasks consists of graphs of size 25 while models are trained on graphs of size 15.

Gradient Descent	Optimization Refinement	Contrastive Shaping	Same Difficulty	Harder Difficulty
No	No	No	80.8%	80.4%
Yes	No	No	80.7%	79.1%
Yes	Yes	No	88.6%	87.9%
Yes	Yes	Yes	**92.6%**	**91.9%**

Table 7. **Path Planning Ablations –** Ablations of proposed components of IRED on performance on the path planning task. Using multiple steps of optimization at each energy landscape and contrastively shaping the energy landscape with ground truth labels both improve path planning performance.

4.3. Planning

Setup. In this section, we evaluate IRED on a basic decision-making problem of finding the shortest path in a graph. In this task, the input to the model is the adjacency matrix of a directed graph, together with two additional node embeddings indicating the start and the goal node of the path-finding problem. The task is to predict a sequence of actions corresponding to the plan. Concretely, the output is a matrix of size $[T, N]$, where T is the number of planning steps and N is the number of nodes in the graph. Each entry (t, i) has a value of 1 if the t-th step of the shortest path is at node i and has a value of 0 otherwise. For all models, including ours, we use a spatial-temporal graph convolution network (STGCN; Yan et al., 2018) to encode the adjacency matrix and the prediction and predict energy function values or score functions. In short, the STGCN has multiple layers. At each layer, for each node, it fuses all features of nodes that are connected to it in the graph and from the current timestep or adjacent timesteps. Just like standard graph convolutional networks, it uses a sum-pooling mechanism to aggregate all embeddings from adjacent nodes. Since, in practice, such planning models are generally evaluated with closed-loop execution, here we only evaluate the success rate that the first action produced by the model shortens the distance between the current node and the goal node. This is the same planning and execution strategy as DiffusionPolicy (Chi et al., 2023).

Quantitative Results. We compare IRED with all baselines across settings in Table 6. IRED outperforms both

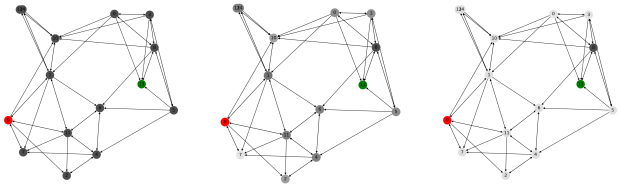

Figure 7. **Optimized Plans Across Landscapes** – Plot of next action prediction in plans across energy landscapes. In each visualization, the green/red nodes indicate start/goal nodes with connections between nodes indicated with arrows. The darkness of a node indicates the score for selecting the corresponding node as the next node to move to in the predicted plan. As landscapes are sequentially optimized, the correct next action is selected.

baselines, especially the diffusion model, by a large margin. Both methods based on energy based formulation (IREM and IRED) perform well on this task, validating the hypothesis that learning energy functions is an effective method for encoding planning problems such as the edge constraints in the path-finding task. Finally, since all methods uses the same STGCN encoder, there is no significant performance drop on generalization to the harder dataset.

Qualitative Results. Similarly to other tasks, we can also visualize the generated solutions by the model across different landscapes. Figure 7 visualizes the prediction of the first node to move to by the planning model. We normalize the prediction scores to 0 to 1. The darker the color, the higher the score. As can be seen in the figure, our IRED model is capable of gradually finding the immediate next action to take: at step 3 it excluded node 7, and the score gradually concentrates on node 8.

Ablations. We ablate each component of IRED in Table 7. We found that running multiple steps of optimization and shaping the energy landscape both lead to improved performance on both the same-difficulty test cases and harder difficulty ones.

5. Conclusion and Discussions

In this paper, we present IRED, a new approach to solving complex reasoning tasks by formulating it as an energy minimization process on a sequence of energy landscapes. We illustrate, on both continuous, discrete, and planning domains, how iterative computation utilizing IRED enables better algorithmic performance and generalization to more complex instances of problems. We further illustrate how the underlying algorithmic computation learned by IRED may be nested to implement more complex algorithmic computations.

Our current reasoning approach with IRED has several limitations. First, the inference time optimization procedure in IRED can still be improved because currently, it requires many steps of gradient descent to find an energy minima. For tasks with known specifications (e.g., shortest path), IRED will conceivably run slower than the algorithms designed specifically for them (e.g., polynomial algorithms for pathfinding), although it is worth noting that IRED is a general machine learning algorithm that does not assume a given task specification and can automatically learn the underlying task constraints from data. On the other hand, it would be interesting to explore how an amortized neural network generator for generating initial solutions or guided optimizers can speed up this procedure. Second, our sequence of annealed energy landscapes is defined through a sequence of added Gaussian noise increments — it would be further interesting to learn the sequence of energy landscapes to enable adaptive optimization. Another current limitation of IRED is that out of the box, IRED in its current form does not leverage any additional memory. Therefore, for tasks that would benefit from explicitly using additional memory to store intermediate results (analogous to chain-of-thought reasoning tasks in language or visual reasoning), IRED might not be as effective as other approaches.

So far we have been applying IRED in continuous and discrete reasoning tasks, and planning on discrete spaces (graphs). Other potential applications of IRED include general mathematical reasoning (Amini et al., 2019; Lu et al., 2021) and decision-making in hybrid discrete-continuous spaces (Garrett et al., 2021; Yang et al., 2023b; Fang et al., 2023). For example, IRED can serve directly as a policy model. In our experiments, we have supervised the energy functions using IID samples from a fixed dataset (as positive examples). However, the energy function can also be supervised with reward signals by modeling a distribution of actions that favor actions yielding high discounted returns (similar to REINFORCE).

Acknowledgements

We gratefully acknowledge support from NSF grant 2214177; from AFOSR grant FA9550-22-1-0249; from ONR MURI grant N00014-22-1-2740; and from ARO grant W911NF-23-1-0034; from MIT Quest for Intelligence; from the MIT-IBM Watson AI Lab; from ONR Science of AI; and from Simons Center for the Social Brain. Yilun Du is supported by a NSF Graduate Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

Impact Statement

No immediate negative social impacts can be derived from the proposed approach in its current form as our evaluation is carried out on relatively standard simple datasets.

References

Amini, A., Gabriel, S., Lin, S., Koncel-Kedziorski, R., Choi, Y., and Hajishirzi, H. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. In *NAACL*, 2019.

Amos, B. and Kolter, J. Z. Optnet: Differentiable Optimization as a Layer in Neural Networks. In *ICML*, 2017.

Anil, C., Pokle, A., Liang, K., Treutlein, J., Wu, Y., Bai, S., Kolter, J. Z., and Grosse, R. B. Path Independent Equilibrium Models Can Better Exploit Test-Time Computation. In *NeurIPS*, 2022.

Arbel, M., Zhou, L., and Gretton, A. Generalized Energy Based Models. In *ICLR*, 2021.

Bai, S., Kolter, J. Z., and Koltun, V. Deep Equilibrium Models. In *NeurIPS*, 2019.

Banino, A., Balaguer, J., and Blundell, C. Pondernet: Learning to Ponder. In *ICML Workshop on Automated Machine Learning*, 2021.

Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. Adaptive Neural Networks for Efficient Inference. In *ICML*, 2017.

Brockett, R. W. Dynamical Systems That Sort Lists, Diagonalize Matrices, and Solve Linear Programming Problems. *Linear Algebra and Its Applications*, 146:79–91, 1991.

Bryson, A. E. *Applied Optimal Control: Optimization, Estimation and Control*. Routledge, 2018.

Cai, J., Shin, R., and Song, D. Making Neural Programming Architectures Generalize via Recursion. In *ICLR*, 2017.

Chen, X., Dai, H., Li, Y., Gao, X., and Song, L. Learning to Stop while Learning to Predict. In *ICML*, 2020a.

Chen, X., Liang, C., Yu, A. W., Song, D., and Zhou, D. Compositional Generalization via Neural-Symbolic Stack Machines. In *NeurIPS*, 2020b.

Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *RSS*, 2023.

Chung, J., Ahn, S., and Bengio, Y. Hierarchical Multiscale Recurrent Neural Networks. In *ICLR*, 2017.

Comas, A., Du, Y., Lopez, C. F., Ghimire, S., Sznaier, M., Tenenbaum, J. B., and Camps, O. Inferring Relational Potentials in Interacting Systems. In *ICML*, 2023.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. Universal Transformers. In *ICLR*, 2019.

Djolonga, J. and Krause, A. Differentiable Learning of Submodular Models. In *NeurIPS*, 2017.

Dong, H., Mao, J., Lin, T., Wang, C., Li, L., and Zhou, D. Neural Logic Machines. In *ICLR*, 2019.

Donti, P. L., Amos, B., and Kolter, J. Z. Task-based End-to-End Model Learning in Stochastic Optimization. In *NeurIPS*, 2017.

Du, Y. and Mordatch, I. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

Du, Y., Li, S., Tenenbaum, B. J., and Mordatch, I. Improved Contrastive Divergence Training of Energy Based Models. In *ICML*, 2021.

Du, Y., Li, S., Tenenbaum, J., and Mordatch, I. Learning Iterative Reasoning through Energy Minimization. In *ICML*, 2022.

Du, Y., Durkan, C., Strudel, R., Tenenbaum, J. B., Dieleman, S., Fergus, R., Sohl-Dickstein, J., Doucet, A., and Grathwohl, W. S. Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC. In *ICML*, 2023.

Fang, X., Garrett, C. R., Eppner, C., Lozano-Pérez, T., Kaelbling, L. P., and Fox, D. DiMSam: Diffusion Models as Samplers for Task and Motion Planning under Partial Observability. *arXiv:2306.13196*, 2023.

Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., and Lozano-Pérez, T. Integrated Task and Motion Planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):265–293, 2021.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., and Zemel, R. Cutting Out the Middle-Man: Training and Evaluating Energy-Based Models Without Sampling. In *ICML*, 2020.

Graves, A. Adaptive Computation Time for Recurrent Neural Networks. *arXiv:1603.08983*, 2016.

Graves, A., Wayne, G., and Danihelka, I. Neural Turing Machines. *arXiv:1410.5401*, 2014.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. Hybrid Computing Using a Neural Network with Dynamic External Memory. *Nature*, 538(7626):471–476, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

Ho, J., Jain, A., and Abbeel, P. Denoising Diffusion Probabilistic Models. In *NeurIPS*, 2020.

Kaiser, Ł. and Sutskever, I. Neural GPUs Learn Algorithms. In *ICLR*, 2016.

Kautz, H., Selman, B., and Hoffmann, J. SATPlan: Planning as Satisfiability. In *IPC*, 2006.

LeCun, Y., Chopra, S., and Hadsell, R. A Tutorial on Energy-based Learning, 2006.

Lu, P., Gong, R., Jiang, S., Qiu, L., Huang, S., Liang, X., and Zhu, S.-C. Inter-GPS: Interpretable Geometry Problem Solving with Formal Language and Symbolic Reasoning. In *ACL*, 2021.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. Deepproblog: Neural Probabilistic Logic Programming. In *NeurIPS*, 2018.

Neelakantan, A., Le, Q. V., and Sutskever, I. Neural Programmer: Inducing Latent Programs with Gradient Descent. In *ICLR*, 2015.

Palm, R., Paquet, U., and Winther, O. Recurrent Relational Networks. In *NeurIPS*, 2018.

Reed, S. and De Freitas, N. Neural Programmer-Interpreters. In *ICLR*, 2016.

Rocktäschel, T. and Riedel, S. End-to-End Differentiable Proving. In *NeurIPS*, 2017.

Rubanova, Y., Sanchez-Gonzalez, A., Pfaff, T., and Battaglia, P. Constraint-Based Graph Network Simulator. In *ICML*, 2022.

Schwarzschild, A., Borgnia, E., Gupta, A., Huang, F., Vishkin, U., Goldblum, M., and Goldstein, T. Can You Learn an Algorithm? Generalizing from Easy to Hard Problems with Recurrent Networks. In *NeurIPS*, 2021.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. In *ICML*, 2015.

Sun, Z. and Yang, Y. Difusco: Graph-Based Diffusion Solvers for Combinatorial Optimization. In *NeurIPS*, 2023.

Wang, P.-W., Donti, P., Wilder, B., and Kolter, Z. SATNet: Bridging Deep Learning and Logical Reasoning using a Differentiable Satisfiability Solver. In *ICML*, 2019.

Wilder, B., Dilkina, B., and Tambe, M. Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization. In *AAAI*, 2019.

Xiao, Z., Kreis, K., Kautz, J., and Vahdat, A. VAEBM: A Symbiosis Between Variational Autoencoders and Energy-based Models. In *ICLR*, 2020.

Xie, J., Lu, Y., Zhu, S.-C., and Wu, Y. A Theory of Generative Convnet. In *ICML*. PMLR, 2016.

Xie, J., Lu, Y., Gao, R., Zhu, S.-C., and Wu, Y. N. Cooperative training of descriptor and generator networks. *T-PAMI*, 42(1):27–45, 2018.

Xu, K., Li, J., Zhang, M., Du, S. S., Kawarabayashi, K.-i., and Jegelka, S. What Can Neural Networks Reason About? In *ICLR*, 2019.

Yan, S., Xiong, Y., and Lin, D. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *AAAI*, 2018.

Yang, F., Yang, Z., and Cohen, W. W. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *NeurIPS*, 2017.

Yang, Z., Ishay, A., and Lee, J. NeurASP: Embracing Neural Networks into Answer Set Programming. In *IJCAI*, 2020.

Yang, Z., Ishay, A., and Lee, J. Learning to Solve Constraint Satisfaction Problems with Recurrent Transformer. In *ICLR*, 2023a.

Yang, Z., Mao, J., Du, Y., Wu, J., Tenenbaum, J. B., Lozano-Pérez, T., and Kaelbling, L. P. Compositional Diffusion-Based Continuous Constraint Solvers. In *CoRL*, 2023b.

Supplementary Material for Iterative Reasoning through Energy Diffusion

In this appendix we provide additional details on IRED. We first provide experimental details on our evaluated tasks in Appendix A. Next, we discuss individual model architectures used in Appendix B.

A. Experimental Details

Continuous Tasks We use dataset setups from (Du et al., 2022) for continuous tasks. Models were trained in approximately 2 hours on a single Nvidia RTX 2080 using a training batch size of 2048 and the Adam optimizer with learning rate 1e-4. Models was trained for approximately 50,000 iterations and evaluated on 20000 test problems.

Discrete Tasks For Sudoku, we train models for 50000 iterations using a single Nvidia RTX 2080 using a training batch size of 64 with the Adam optimizer with learning rate 1e-4 and are evaluated on the full test datasets provided in (Wang et al., 2019; Palm et al., 2018).

For Connectivity tasks, we generate random graphs using algorithms from Graves et al. (2016). Essentially, it first generates a set of random points on a 2D plane uniformly inside a unit square. Next, it samples the out-degree k (the number of out-going edges) for each node based on a uniform distribution. Finally, it connects each node to its k closest neighbors. The advantage of this generation process is that the generated graph will be close to a planar graph so that it can be easily visualized, and more importantly, it allows for fine-grained control of the connectivity of the graph by setting the out-degree range. In practice, we uniformly sample the out-degrees from $[1, n/2]$, where n is the number of nodes in the graph. Based on the sampled adjacency matrix, we use the floyd-warshall algorithm to compute its connectivity matrix. Note that there are no distances associated with the edges (*i.e.*, all edges are of unit length). We train models for 100000 iterations using a single Nvidia RTX 2080 with batch size 512 with the Adam optimizer.

Planning Task For planning, we use the same procedure as in the connectivity tasks to generate graphs. To create graphs with a large enough treewidth, we set the out-degree sample range to be $[1, n/5]$, where n is the number of nodes. All edges are of unit length. We use the floyd-warshall algorithm to compute the pairwise shortest distance. This enables us to evaluate whether the first action predicted by the model shortens the distance between the current node and the goal. We train models for 100000 iterations using a single Nvidia RTX 2080 with batch size 512 with the Adam optimizer.

B. Model Architectures

Continuous Task. For continuous tasks, we use the architecture in Table 8 to train both IRED and the IREM baseline. We use the architecture in Table 9 to parameterize the diffusion model baseline.

Discrete Task. For Sudoku, we use the architecture in Table 10. It encodes the Sudoku board with a convolutional neural network with the residual connection design, borrowed from He et al. (2016).

For Connectivity, we use the architecture adapted from Dong et al. (2019), as detailed in Table 11. It uses a relational neural network to fuse the connectivity information from neighboring nodes.

Planning Task. For planning tasks, we use the architecture the same architecture as the connectivity task, as detailed in Table 12. To encode temporal information, at each layer, we stack the node embedding from the previous time step, the current step, and the next step to implement temporal convolution. This is equivalent to the spatial-temporal graph convolution networks (STGCN; Yan et al., 2018).

We have also attached the code to reproduce all experiments in the supplementary material.

Linear 512
Linear 512
Linear 512
Linear $\rightarrow$ 1

Table 8. The model architecture for IRED on continuous-space algorithmic reasoning tasks

Linear 512
Linear 512
Linear 512
Linear $\rightarrow$ Output Dim

Table 9. The model architecture for diffusion baselines on continuous-space algorithmic reasoning tasks.

3x3 Conv2D, 384
Resblock 384
Resblock 384
Resblock 384
Resblock 384
Resblock 384
Resblock 384
3x3 Conv2D, 9

Table 10. The model architecture for IRED and diffusion baselines on the Sudoku task. The energy value is computed using the L2 norm of the final predicted output similar to (Du et al., 2023), while the output is directly used as noise prediction for the diffusion baseline.

NLM Arity=3, Hidden=64
NLM Arity=3, Hidden=64
Max-Pooling over All Edge Features
Linear, 1

Table 11. The model architecture for IRED and diffusion baselines on the connectivity task. For the diffusion baseline, we simply remove the pooling layer and apply the same linear layer on all edge embeddings to predict the noise value for each entry.

NLM Arity=2, Hideen=64
NLM Arity=2, Hideen=64
Max-Pooling over All Node Features
Linear, 1

Table 12. The model architecture for IRED and diffusion baselines on the shortest-path task. For the diffusion baseline, we simply remove the pooling layer and apply the same linear layer on all node embedding to predict the noise value for each entry.