Enabling Compositional Generalization of AI Systems

by

Shuang Li

B.Eng., Dalian University of Technology (2015)
M.Phil., The Chinese University of Hong Kong (2018)

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

Authored by: Shuang Li
Department of Electrical Engineering and Computer Science
August 31, 2023

Certified by: Antonio Torralba
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Enabling Compositional Generalization of AI Systems

by

Shuang Li

Submitted to the Department of Electrical Engineering and Computer Science
on August 31, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

A vital aspect of human intelligence is the ability to compose increasingly complex concepts out of simpler ideas, enabling both rapid learning and adaptation of knowledge. Despite their impressive performance, current AI systems fall short in this area and are often unable to solve tasks that fall outside of their training distribution. The work contained in this thesis aims to bridge this gap by incorporating compositionality into deep neural networks, thereby enhancing their ability to generalize and solve novel and complex tasks, such as generating 2D images and 3D assets based on complicated specifications, or enabling humanoid agents to perform a diverse range of household activities. The implications of this thesis are far-reaching, as compositionality has numerous applications across fields such as biology, robotics, and art production. By significantly improving the compositionality ability of AI systems, this research will pave the way for more data-efficient and powerful models in different research areas.

Thesis Supervisor: Antonio Torralba
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Antonio Torralba. Without his help, support, and unparalleled brilliance in research, the completion of my Ph.D. would not have been possible. Antonio has not only provided me with invaluable guidance and suggestions for my research but has also served as a life mentor, assisting me in making crucial decisions. Working with him has been a privilege, and I consider myself incredibly fortunate to have had this opportunity.

I would also like to extend my sincere thanks to Igor Mordatch, who has provided me with significant support throughout my research. Whenever I presented a new idea, Igor was instrumental in helping me validate and justify it. Working with Igor has been a truly enriching experience, and I can confidently say that he is one of the kindest individuals I have had the pleasure of collaborating with.

I must express my gratitude to Phillip Isola. Even though I never officially joined his lab, I have always felt welcomed as part of the team. Phillip's warmth and approachability make him an exceptional mentor who never adds unnecessary pressure to his students and is always friendly and supportive to everyone.

I am also very grateful for the opportunity to work with Jacob Andreas. Jacob's exceptional talent has left a lasting impression on me. I still vividly remember how he significantly improved my paper within mere hours. He serves as a role model for me as a top researcher, and I am grateful for the opportunity to have worked alongside him.

Special acknowledgment goes to my collaborators, labmates, and friends, whose engaging discussions and assistance have been vital throughout my journey: Joshua B. Tenenbaum, Pulkit Agrawal, Vincent Sitzmann, David Bau, Leslie Kaelbling, Bryan Russell, Josef Sivic, Sanja Fidler, Yuke Zhu, Anima Anandkumar, Florian Shkurti, Yilun Du, Yunzhu Li, Xavier Puig Fernandez, Tianmin Shu, Krishna Murthy Jatavallabhula, Nan Liu, Tongzhou Wang, Hengshuang Zhao, Tao Chen, Sarah Schwettmann, Tamar Rott Shaham, Joanna Materzynska, Anurag Ajay, Bolei Zhou, Jun-Yan Zhu, Hang Zhao, Adrià Recasens, Jonas Wulff, Wei-Chiu Ma, Ching-Yao Chuang, Manel Baradad, Pratyusha Sharma, Yichen Li, George Cazenavette, Adrián Rodríguez Muñoz, Kabir Swain, Jingwei Ma, William Peebles, Ethan Weber, Steven Liu, Zilin Wang, Sirui Li, Pingchuan Ma, Lucy Chai, Caroline Chan, Yen-Chen Lin, Minyoung Huh, Yonglong Tian, Toru Lin, Yanwei Wang, Tiancheng Yu, Anthony Simeonov, Gabe Margolis, Brian Cheung, Aviv Netanyahu, and Clinton Wang. This dissertation is not possible without your help.

Lastly, I am profoundly grateful to my parents. Your unwavering support, encouragement, and belief in me have shaped me into who I am today. Your influence has been instrumental in every step of my academic journey, and I cannot thank you enough for your endless love and faith in me.

This dissertation is a testament to the collaboration, guidance, and encouragement of many remarkable and generous people who shaped both me and my research. It is to them that I dedicate my work, grateful for the collective effort that made this achievement possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Compositionality is a crucial aspect of human intelligence missing from modern AI systems. The ability to compose concepts: to combine patterns, ideas, and subgoals to build a structured representation of the world, and then to reason about the world by manipulating individual components, manifests in crucial cognitive abilities. Humans can incorporate individual observations into sophisticated knowledge and belief structures, make small targeted adjustments to complex plans, imagine alternatives to a base scenario, and create new technology or art inspired by existing work. Such compositional abilities have largely failed to materialize in AI systems, yet achieving this could be one of the keys to unlocking major AI capabilities such as continual learning, controllable and robust behavior, high-level planning, counterfactual reasoning, and stronger generalization.

This dissertation focuses on developing neural networks that exhibit compositional abilities to solve a wide range of tasks, such as image generation, question answering, mathematical reasoning, robot manipulation, and embodied decision-making. The goal is for the network to solve tasks with combinations of concepts, goals or skills beyond what it was exposed to during training.

Our study on compositional AI spans the following two axes: **Prior Knowledge** and **Compositional Structure**. Prior knowledge describes the set of basic concepts and capabilities that the model learns over the course of training. The large deep learning models [131, 125, 13] trained on a large amount of data contain rich prior knowledge, but they miss the compositional structure, which is another key component to achieve compositionality. To build the compositional structure, we propose compositional operators to compose the basic concepts. Applying the compositional operators to pre-trained models enables us to significantly improve the compositional generation ability of AI systems.

This dissertation incorporates the introduction of how to build the *Compositional Structure* in the first two parts. **Part I: Composing concepts and goals**: We develop neural networks that can compose concepts or goals to give rise to highly controllable and complex, fine-grained behaviors. **Part II: Composing models**: We compose pre-trained models from different domains to yield powerful crossmodal capabilities without any training or finetuning. In **Part III: Transferring Compositionality**, we introduce how to achieve *Prior Knowledge* in a data-efficient way by transferring knowledge from pre-trained models.

1.1 Composing Concepts and Goals

Humans have limitless capacity to recombine concepts in unconventional ways. For example, given a 3D scene, we can easily imagine changing the color of each object beyond their natural distribution, and we can separate the color changes of one object from color changes of other objects. But neural networks seem to lack such compositional ability, and often fail to generalize to new patterns beyond their training distribution. As an example, consider the state-of-the-art image generation model DALL-E 2 [125]. Given an input language description "a red house and a blue car", this model often generates images that either omit the car or apply a color to the wrong objects (Fig. 1-2 first column), even when the model is perfectly capable of generating separate images of red houses and blue cars.

We try to solve this problem by equipping the model with three compositional operators [25, 88] – AND, OR, and NOT – that allow the model to combine concepts such as facial attributes, objects, object relations, and natural language descriptions (Fig. 1-1). The key insight is to use the inherent compositional behavior of probability

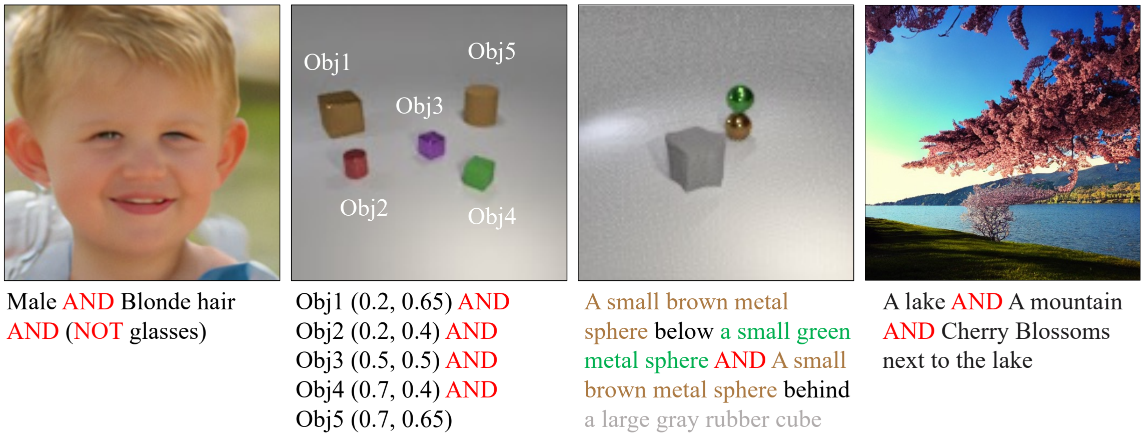

Male AND Blonde hair AND (NOT glasses)
Obj1
Obj2
Obj3
Obj4
Obj5
Obj1 (0.2, 0.65) AND Obj2 (0.2, 0.4) AND Obj3 (0.5, 0.5) AND Obj4 (0.7, 0.4) AND Obj5 (0.7, 0.65)
A small brown metal sphere below a small green metal sphere AND A small brown metal sphere behind a large gray rubber cube
A lake AND A mountain AND Cherry Blossoms next to the lake

Figure 1-1: **Composing concepts and goals.** We develop neural networks that can compose various concepts or goals for highly controllable visual generation, such as facial attributes, objects, object relations, and natural language descriptions, to give rise to highly controllable and complex, fine-grained behaviors.

distributions to model concept composition within iterative frameworks such as Energy-based Models (EBMs) [79] and Diffusion Models [145, 147, 51]. Producing an image that matches an individual concept can be understood as a "subgoal" of the generative model, and hence this framework naturally extends to composing subgoals in goal-directed settings. The proposed compositional operators allow us to compose concepts or goals using pre-trained models, resulting in remarkable generalization abilities and fine-grained user control over the model's outputs.

Composable energy-based model [25, 87]. EBMs [79] are trained to match an unnormalized probability distribution $p_\theta(\boldsymbol{x}) \propto e^{-E_\theta(\boldsymbol{x})}$ to a given dataset of images. Its energy function E_θ, which is parameterized by a neural network, maps an image $\boldsymbol{x}$ to a scalar energy. At inference time, the model optimizes $\boldsymbol{x}$ to minimize $E_\theta(\boldsymbol{x})$ under a stochastic process called Langevin sampling: $\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \frac{\lambda}{2}\nabla_{\boldsymbol{x}} E_\theta(\boldsymbol{x}_{t-1}) + \omega_t$, $\omega_t \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. When the EBM is well-trained, it can generate photorealistic images by running Langevin sampling for many iterations.

We want to generate samples from some conditional distribution $p_\theta(\boldsymbol{x}|\boldsymbol{c})$, where $\boldsymbol{c}$ is the conditioned concept which may be complex and far from the training distribution, such as a long sentence that is never seen during training. The proposed

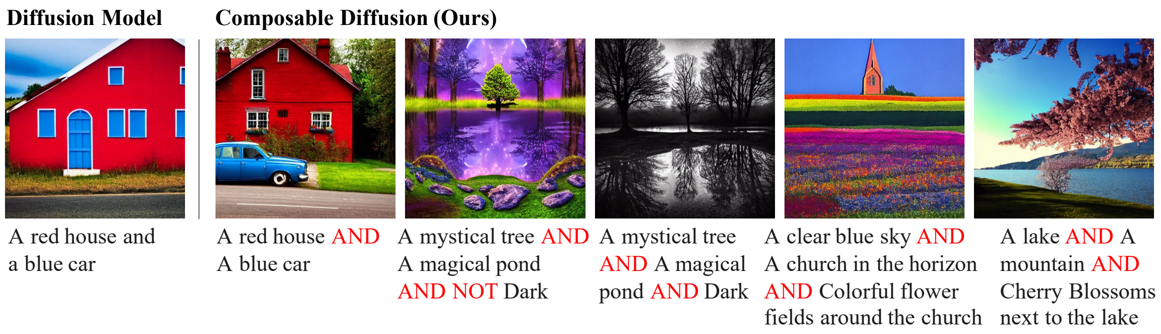

Diffusion Model
Composable Diffusion (Ours)
A red house and a blue car
A red house AND A blue car
A mystical tree AND A magical pond AND NOT Dark
A mystical tree AND A magical pond AND Dark
A clear blue sky AND A church in the horizon AND Colorful flower fields around the church
A lake AND A mountain AND Cherry Blossoms next to the lake

Figure 1-2: **Results of Stable Diffusion [131] and our composable diffusion model.** Our method can compose concepts using compositional operators on pre-trained diffusion models without training or fine-tuning. It generates images that reflect the combined concepts more accurately.

compositional operators [25] leverage inherent structure in $\boldsymbol{c}$, without needing to modify the architecture or training scheme of the EBM. We discover that the EBM energy of an image conditioned on two concepts can be set to the sum of the energies of the image conditioned on each individual concept. This **concept conjunction (AND)** operator produces images that convincingly combine the concepts. Using a similar strategy, we introduce the **concept disjunction (OR)** operator to generate images containing at least one of several concepts, and **concept negation (NOT)** to remove concepts from an image. In Chapter 2, we show that chaining energy functions in this manner can generate images with conditions far more complex than those observed at training time. Remarkably, these compositional operators perform well even when they are only introduced at inference time: the EBM can be trained entirely on individual objects or concepts, and generalize to multiple concepts at test time without additional finetuning.

Composable diffusion model [88]. Diffusion Models [145, 147, 51] have shown amazing image generation results. Starting from random noise, diffusion models generate images through an iterative denoising process: $\boldsymbol{x}_{t-1} = \boldsymbol{x}_t - \epsilon_\theta(\boldsymbol{x}_t, t) + \omega_t$, $\omega_t \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$, where $\epsilon_\theta(\boldsymbol{x}_t, t)$ is the noise pattern to be filtered out. In Chapter 3, we show that diffusion models can be interpreted as implicitly parameterized EBMs, where the noise term $\epsilon_\theta(\boldsymbol{x}_t, t)$ represents the gradient of a time-dependent energy function.

With such an interpretation, we can generate images conditioned on combinations of concepts by composing their individual noise terms using the proposed compositional operators. Similarly, we can define the **AND** operator and the **NOT** operator on top of diffusion models. As with EBMs, employing compositional operators enhances the robustness, user control, and generalization ability of these powerful models. They produce high-quality images matching detailed specifications (Fig. 1-2).

1.2 Composing Models

Humans continually expand their capabilities over their lifetime, readily integrating new knowledge and skills with existing ones. Compositionality is a powerful framework for imbuing neural networks with a similar capacity. Large pre-trained models exhibit distinct and complementary capabilities dependent on the data they are trained on. Language models are capable of textual reasoning but cannot process visual information, while vision models such as DALL-E 2 [125] can generate photorealistic images but fail to understand complex language descriptions. However, there is no effective way to robustly combine the capabilities of these models without additional training. Prior approaches [2, 174] for composing models are task-specific or scale poorly.

In Chapter 4, we propose a unified framework [83] to compose different pre-trained models in a zero-shot manner* to solve diverse multi-modal tasks without any training or finetuning. The framework consists of a single generator model that proposes feasible solutions and an ensemble of scorer models that guide the generator to the best solution. The generator's output is iteratively refined to optimize the sum of the scores in a manner reminiscent of our earlier Concept Conjunction operator.

This iterative closed-loop feedback between the generator and scorers results in stable performance, and enables each scorer to compensate for the potential weaknesses of other scorers. We demonstrate that guiding the generator with an ensemble of scorers significantly outperforms a generator guided by a single scorer.

*By zero-shot, we mean the composed models are never trained together on the evaluation task.

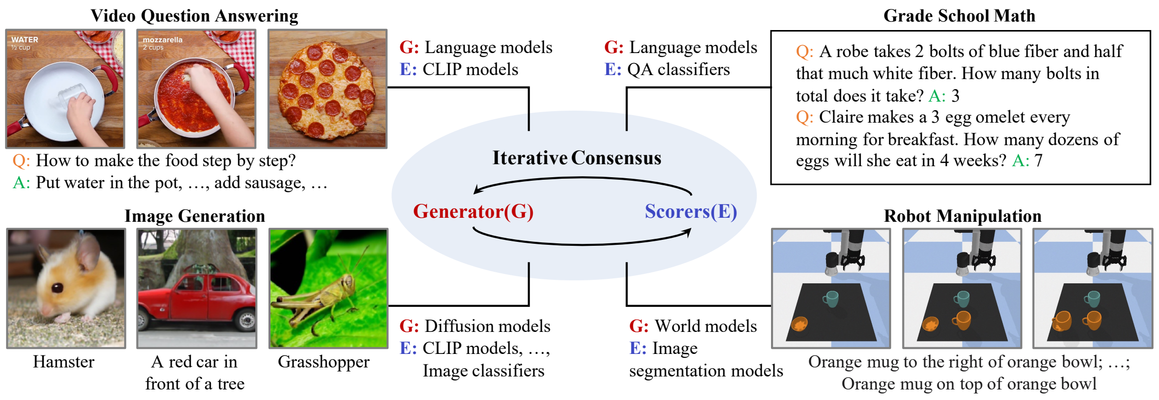

Video Question Answering
WATER
mozzarella
Q: How to make the food step by step?
A: Put water in the pot, …, add sausage, …
Image Generation
Hamster
A red car in front of a tree
Grasshopper
G: Language models
E: CLIP models
G: Language models
E: QA classifiers
Iterative Consensus
Generator(G)
Scorers(E)
G: Diffusion models
E: CLIP models, …, Image classifiers
G: World models
E: Image segmentation models
Grade School Math
Q: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? A: 3
Q: Claire makes a 3 egg omelet every morning for breakfast. How many dozens of eggs will she eat in 4 weeks? A: 7
Robot Manipulation
Orange mug to the right of orange bowl; …; Orange mug on top of orange bowl

Figure 1-3: **Composing models.** We develop a unified framework to compose pre-trained models from different modalities to solve diverse tasks in a zero-shot manner.

This framework achieves strong zero-shot performance on diverse tasks, including image generation, video question answering, mathematical reasoning, and robot manipulation (Fig. 1-3). For example, in the video question-answering task, the generator is a language model which is trained on pure text data and the scorers are CLIP [119] models which are trained on image-caption pairs. Their combination can be used to answer questions about videos even though none of them has ever seen video data. In addition, the models can be trained on different data and tasks in an incremental learning manner, presenting an avenue for applications in lifelong and continual learning settings.

1.3 Transferring Compositionality

Composing concepts and composing models aim at building compositional structures. An equally crucial factor in achieving compositionality is enhancing its prior knowledge, which amplifies the models' capacity to grasp a wide range of concepts.

Recent advancements demonstrate that sufficiently massive models, trained on extensive datasets, exhibit more than just an understanding of basic concepts; they also showcase the emergence of compositional abilities. After being exposed to images or text aggregated from billions of web pages, image and text generation models can often succeed on unconventional prompts such as generating an image with "teddy

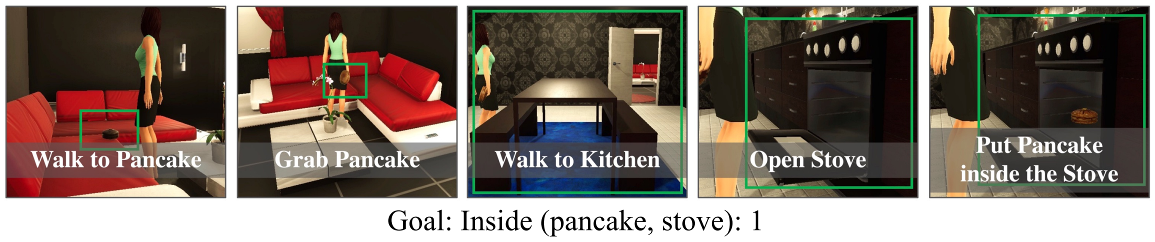

Walk to Pancake
Grab Pancake
Walk to Kitchen
Open Stove
Put Pancake inside the Stove
Goal: Inside (pancake, stove): 1

Figure 1-4: **Decision-making tasks in VirtualHome [118].** VirtualHome is a large-scale embodied environment to simulate household activities, such as "making breakfast". Agents are represented as humanoid avatars which can interact with the environment through high-level instructions.

bears swimming at the Olympics", suggesting that despite some outstanding failure modes (as described before), these models have arrived at some implicit structure that allows them to reason in a compositional manner. Such large pre-trained models can be further combined with the compositional structures described above to enhance compositionality.

Nonetheless, numerous tasks like decision-making, embodied behavior, and policy learning pose considerably greater challenges for such scaling. The collection of extensive data on a large scale for many real-world tasks remains intricate or, in certain cases, unattainable. In this part, rather than relying on training extensive models with massive datasets to establish a solid foundation of prior knowledge for compositionality, we study how to transfer the compositionality from large pre-trained models to solve new tasks without the necessity of vast amounts of data.

Language models are one of the most commonly used models for studying compositionality because of the compositional nature of natural language. Large language models such as GPT-4 [109] have shown remarkable compositionality on language tasks. One question to ask is whether we can transfer the compositionality in pre-trained language models to solve other tasks, such as embodied decision-making tasks shown in Fig. 1-4, without using extensive training data.

In Chapter 5, we use large pre-trained language models as a general scaffold for interactive decision-making across a variety of environments by converting policy inputs into sequential data. We demonstrate that language modeling induces compositional generalization in learned policies: initializing a policy with a pre-trained language model substantially improves out-of-distribution performance on novel combinations. These results highlight the promise of leveraging existing large pre-trained models and transferring their compositionality to solve new tasks.

We believe that compositionality is a crucial component of next-generation AI systems. Our research presents a new direction for overcoming the constraints of current AI models and paving the way for more composable AI systems. By building AI systems that can learn and compose from multiple pre-trained models, we can unlock the potential for AI systems to continually improve performance and effectively handle new tasks in a dynamic world. Through model composition and continual learning, we are one step closer to realizing the full potential of AI.

1.4 Dissertation Structure

This dissertation investigates compositionality from three distinct but interrelated perspectives: the composition of concepts and goals, the combination of models across various domains, and the transfer of compositional strategies to novel applications.

Part I focuses on exploring methods for composing concepts and goals, such as facial attributes, objects, and natural language descriptions, using two types of generative models, *i.e.*, energy-based models and diffusion models.

– Chapter 2 composes energy-based models using compositional operators, *i.e.*, conjunction, disjunction, and negation, for image generation. The compositional operators allow us to generalize to new combinations that are outside the training distribution.

– Chapter 3 interprets diffusion models as energy-based models, and the proposed compositional operators can be directly applied to diffusion models. As diffusion models are more stable to train, composing concepts over diffusion models enables us to generate more complex and photorealistic images.

Part II delves into methods for composing models from different domains. This process aims to construct a robust multi-modal framework capable of tackling a wide range of tasks that may not have been encountered during training.

- In Chapter 4, we introduce an approach that enables the composition of pre-trained models from diverse domains, such as vision models, language models, and mathematical problem solvers. The method facilitates communication and collaboration between pre-trained models, harnessing their collective intelligence. Models learn from each other, exchange feedback, and iteratively improve their performance.

Part III takes a different perspective on compositionality by investigating its transferability for solving novel tasks without relying on extensive amounts of data.

- Chapter 5 explores pre-trained models as a way to bootstrap compositional reasoning capabilities, facilitating the transfer of accumulated knowledge and insights from well-established models to unexplored tasks. This approach substantially lessens the requirement for intensive training on large datasets, thus circumventing a common bottleneck in AI development.

Part I

Composing Concepts and Goals

Chapter 2

Compositional Visual Generation with Energy-Based Models

Yilun Du, Shuang Li, Igor Mordatch; NeurIPS 2020.

Nan Liu*, Shuang Li*, Yilun Du*, Joshua B. Tenenbaum, Antonio Torralba; NeurIPS 2021.

A vital aspect of human intelligence is the ability to compose increasingly complex concepts out of simpler ideas, enabling both rapid learning and adaptation of knowledge. We find that energy-based models (EBMs) can exhibit this ability by directly combining probability distributions. Samples from the combined distribution correspond to compositions of concepts. For example, given one distribution for smiling face images, and another for male faces, we can combine them to generate smiling male faces. This allows us to generate natural images that simultaneously satisfy conjunctions, disjunctions, and negations of concepts.

2.1 Introduction

Humans are able to rapidly learn new concepts and continuously integrate them into prior knowledge. The core component in enabling this is the ability to compose increas-

*Equal Contribution

ingly complex concepts out of simpler ones as well as recombining and reusing concepts in novel ways [35]. By combining a finite number of primitive components, humans can create an exponential number of new concepts, and use them to rapidly explain current and past experiences [78]. We are interested in enabling such capabilities in machine learning systems, particularly in the context of generative modeling.

Past efforts have attempted to enable compositionality in several ways. One approach decomposes data into disentangled factors of variation and situates each datapoint in the resulting - typically continuous - factor vector space [157, 46]. The factors can either be explicitly provided or learned in an unsupervised manner. In both cases, however, the dimensionality of the factor vector space is fixed and defined prior to training. This makes it difficult to introduce new factors of variation, which may be necessary to explain new data, or to taxonomize past data in new ways. Another approach to incorporate compositionality is to spatially decompose an image into a collection of objects, each object slot occupying some pixels of the image defined by a segmentation mask [155, 39]. Such approaches can generate visual scenes with multiple objects, but may have difficulty in generating interactions between objects. These two incorporations of compositionality are considered distinct, with very different underlying implementations.

In this chapter, we introduce how to implement compositionality via energy-based models (EBMs). Instead of using an explicit vector of factors that is input to a generator function, or object slots that are blended to form an image, our unified treatment defines factors of variation and object slots via energy functions. Each factor is represented by an individual scalar energy function that takes an image as input and outputs a low energy value if the factor is exhibited in the image. Images that exhibit the factor can then be generated implicitly through a Markov Chain Monte Carlo (MCMC) sampling process that minimizes the energy. Importantly, it is also possible to run the MCMC process on some *combinations* of energy functions to generate images that exhibit multiple factors or multiple objects, in a globally coherent manner.

There are several ways to combine energy functions. One can add or multiply

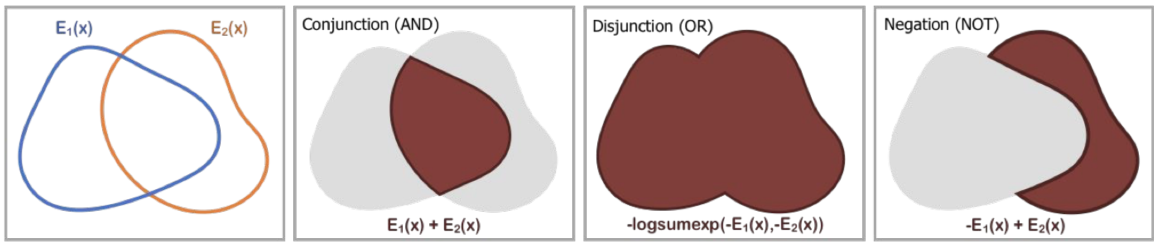


Figure 2-1: **Illustration of logical composition operators over energy functions E_1 and E_2** (drawn as level sets where red = valid areas of samples, grey = invalid areas of samples).

distributions as in mixtures [139, 39] or products [49] of experts. We view these as probabilistic instances of logical operators over concepts. Instead of using only one, we consider three operators: logical conjunction, disjunction, and negation (illustrated in Figure 2-1). We can then flexibly and recursively combine multiple energy functions via these operators. More complex operators can be formed out of our base operators.

EBMs with such composition operations enable a breadth of new capabilities - among them is a unique approach to continual learning. Our formulation defines concepts or factors implicitly via examples, rather than pre-declaring an explicit latent space ahead of time. For example, we can create an EBM for the concept of "black hair" from a dataset of face images that share this concept. New concepts (or factors), such as hair color can be learned by simply adding a new energy function and can then be combined with energies of previously trained concepts. This process can repeat continually. This view of few-shot concept learning and generation is similar to the work of [127], with the distinction that instead of learning to generate holistic images from few examples, we learn *factors* from examples, which can be composed with other factors. A related advantage is that finely controllable image generation can be achieved by specifying the desired image via a collection of logical clauses, with applications to neural scene rendering [34]. In summary, our contributions are:

- First, while the composition of energy-based models has been proposed in abstract settings before [49], we show that it can be used to generate plausible natural images.

- Second, we propose a principled approach to combine independently trained energy models based on logical operators, which can be chained recursively, allowing controllable generation based on a collection of logical clauses at test time.
- Finally, by being able to recursively combine independent models, we show our approach allows us to extrapolate to new concept combinations, continually incorporate new visual concepts for generation, and infer concept properties.

2.2 Related Work

Our work draws on results in energy-based models - see [79] for a comprehensive review. A number of methods have been used for inference and sampling in EBMs, from Gibbs Sampling [50], Langevin Dynamics [166, 30], Path Integral methods [29] and learned samplers [71, 149]. In this work, we apply EBMs to the task of compositional visual generation.

Compositionality has been incorporated in representation learning (see [4] for a summary) and generative modeling. One approach to compositionality has focused on learning disentangled factors of variation [45, 77, 157]. Such an approach allows for the combination of existing factors, but does not allow the addition of new factors. A different approach to compositionality includes learning various different pixel/segmentation masks for each concept [39, 40]. However, such a factorization may have difficulty capturing the global structure of an image, and in many cases, different concepts cannot be explicitly factored using segmentation masks. Individual compositions of factors can also be seen as a domain translation problem [10, 116, 101]. Such a formulation, however, requires separate retraining of models for each considered composition.

In contrast, our approach towards compositionality focuses on composing separate learned probability distributions of concepts. Such an approach allows viewing factors of variation as constraints [98]. In prior work, [48] show that products of EBMs can be used to decompose complex generative modeling problems into simpler ones. [157, 46]

further apply products of distributions over the latent space of VAE to define compositions. Both of them rely on joint training to learn compositions of a fixed number of concepts. In contrast, in this work, we show how we can realize concept compositions using completely independently trained probability distributions. Furthermore, the proposed three compositional logical operators, *i.e.*, conjunction, disjunction, and negation, can be realized and nested together through the manipulation of probability distributions of different concepts.

Our compositional approach is inspired by the goal of continue/lifelong learning - see [110] for a thorough review. New concepts can be composed with past concepts by combining their probability distributions. Many methods in continual learning are focused on how to overcome catastrophic forgetting [73, 85], but do not support dynamically growing capacity. Progressive growth of models [132] has been considered, but is implemented at the level of the model architecture, whereas our method composes independent models together.

2.3 Method

In this section, we first give an overview of the energy-based Model formulation employed, along with the introduction of three logical operators that operate on these models. Subsequently, we delve into the distinctive properties that arise from this particular form of compositionality.

2.3.1 Energy-based models

EBMs represent data by learning an unnormalized probability distribution across the data. For each data point $\boldsymbol{x}$, an energy function $E_\theta(\boldsymbol{x})$, parameterized by a deep neural network, outputs a scalar energy value such that the model distribution can be written as:

$$p_\theta(\boldsymbol{x}) \propto e^{-E_\theta(\boldsymbol{x})}. \tag{2.1}$$

To train an EBM on a data distribution p_D, we use contrastive divergence [48]. In particular we use the methodology defined in [30], where a Monte Carlo estimate of maximum likelihood $\mathcal{L}$ is minimized with the following gradient:

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{\boldsymbol{x}^+ \sim p_D} \nabla_\theta E_\theta(\boldsymbol{x}^+) - \mathbb{E}_{\boldsymbol{x}^- \sim p_\theta} \nabla_\theta E_\theta(\boldsymbol{x}^-). \tag{2.2}$$

To sample $\boldsymbol{x}^-$ from p_θ for both training and generation, we use MCMC based on Langevin dynamics [162]. Samples are initialized from uniform random noise and are iteratively refined using

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \frac{\lambda}{2} \nabla_{\boldsymbol{x}} E_\theta(\boldsymbol{x}_{t-1}) + \omega_t, \ \omega_t \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}), \tag{2.3}$$

where t is the iteration step, λ is the step size, ω_t is sampled from the Gaussian distribution, and $\boldsymbol{I}$ is the identity matrix. We refer to each iteration of Langevin dynamics as a negative sampling step. We note that this form of sampling allows us to use the gradient of the combined distribution to generate samples from distributions composed of p_θ and the other distributions. We use this ability to generate images from multiple different compositions of distributions. To enable high-resolution image generation, we further apply techniques in [27] to improve EBM training on CelebA images.

2.3.2 Composition of energy-based models

We next present different ways to compose EBMs using the proposed compositional operators. We consider a set of independently trained EBMs, $E(\boldsymbol{x}|\boldsymbol{c}_1), E(\boldsymbol{x}|\boldsymbol{c}_2), \ldots, E(\boldsymbol{x}|\boldsymbol{c}_n)$, which are learned conditional distributions on underlying concept codes $\boldsymbol{c}_i$. Latent codes we consider include position, size, color, gender, hairstyle, and age, which we also refer to as concepts.

Concept conjunction. In concept conjunction, given a set of independent concepts (such as a particular hairstyle, facial expression, or eye color), we aim to construct an output with the specified combination of the concepts. Since the likelihood of an output given a set of specific concepts is equal to the product of the likelihood of each individual concept, we have Equation (2.4), which is also known as the product of experts [49]:

$$p(\boldsymbol{x}|\boldsymbol{c}_1 \text{ AND } \boldsymbol{c}_2 \dots \text{ AND } \boldsymbol{c}_n) \propto \prod_{i=1}^{n} p(\boldsymbol{x}|\boldsymbol{c}_i) \propto e^{-\sum_{i=1}^{n} E(\boldsymbol{x}|\boldsymbol{c}_i)}. \tag{2.4}$$

We can thus apply Equation (2.3) to the distribution that is the sum of the energies of different concepts. We sample from this joint concept space:

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \frac{\lambda}{2}\nabla_{\boldsymbol{x}} \left(\sum_{i=1}^{n} E_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{c}_i) \right) + \omega_t,\ \omega_t \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}), \tag{2.5}$$

where ω_t is the Gaussian noise.

Concept disjunction. In concept disjunction, we aim to construct an output that represents either of the concepts, such as the colors red or blue. This involves creating a distribution where there is a significant probability that either of the chosen concepts is true. A natural choice of such a distribution is the sum of the likelihood of each concept:

$$p(\boldsymbol{x}|\boldsymbol{c}_1 \text{ OR } \boldsymbol{c}_2 \dots \text{ OR } \boldsymbol{c}_n) \propto \sum_{i=1}^{n} p(\boldsymbol{x}|\boldsymbol{c}_i)/Z(\boldsymbol{c}_i), \tag{2.6}$$

where $Z(\boldsymbol{c}_i)$ denotes the partition function of each concept. A tractable simplification becomes available if we assume all partition functions $Z(\boldsymbol{c}_i)$ to be equal:

$$\sum_{i=1}^{n} p(\boldsymbol{x}|\boldsymbol{c}_i) \propto \sum_{i=1}^{n} e^{-E(\boldsymbol{x}|\boldsymbol{c}_i)} = e^{\text{logsumexp}\left(-E(\boldsymbol{x}|\boldsymbol{c}_1), -E(\boldsymbol{x}|\boldsymbol{c}_2), \dots, -E(\boldsymbol{x}|\boldsymbol{c}_n)\right)}, \tag{2.7}$$

where $\text{logsumexp}(f_1, \dots, f_n) = \log \sum_{i=1}^{n} \exp(f_i)$. We can thus apply Equation (2.3) again to the distribution that is a negative smooth minimum of the energies of each concept. We then sample from the disjunction concept space:

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \frac{\lambda}{2}\nabla_{\boldsymbol{x}}\text{logsumexp}\big(-E(\boldsymbol{x}_{t-1}|\boldsymbol{c}_1), -E(\boldsymbol{x}_{t-1}|\boldsymbol{c}_2), \ldots, -E(\boldsymbol{x}_{t-1}|\boldsymbol{c}_n)\big) + \omega_t, \tag{2.8}$$

where $\omega_t \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$. While the assumption that leads to Equation 2.7 is not guaranteed to hold in general, in our experiments, we empirically find that the partition function $Z(\boldsymbol{c}_i)$ estimates to be similar across different partition functions and also analyze cases in which partitions functions are different (see Appendix A.1).

Concept negation. In concept negation, we aim to generate an output that does not contain certain concepts. Given the color red, we want the model output to have a different color, such as blue. Thus, we need to construct a distribution that places a high likelihood of data that is different from a given concept. One choice is a distribution inversely proportional to the concept. Importantly, negation must be defined with respect to another concept to be useful. For example, in image generation, the opposite of a person wearing eyeglasses is a person without wearing eyeglasses rather than a random noise image. Negation without a data distribution is not integrable and leads to a generation of chaotic textures which, while satisfying the absence of a concept, is not desirable. Thus in our experiments with the negation operation, we combine it with another concept to ground the negation and obtain an integrable distribution:

$$p(\boldsymbol{x}|\text{NOT}(\boldsymbol{c}_1), \boldsymbol{c}_2) \propto \frac{p(\boldsymbol{x}|\boldsymbol{c}_2)}{p(\boldsymbol{x}|\boldsymbol{c}_1)^{\alpha}} \propto e^{\alpha E(\boldsymbol{x}|\boldsymbol{c}_1) - E(\boldsymbol{x}|\boldsymbol{c}_2)}. \tag{2.9}$$

We found the smoothing parameter α to be a useful regularize and we use $\alpha = 0.01$ in our experiments. The above equation allows us to apply Langevin dynamics to sample from the negation concept space:

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \frac{\lambda}{2}\nabla_{\boldsymbol{x}}\big(-\alpha E(\boldsymbol{x}_{t-1}|\boldsymbol{c}_1) + E(\boldsymbol{x}_{t-1}|\boldsymbol{c}_2)\big) + \omega_t, \tag{2.10}$$

where $\omega_t \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. The equation can be further extended to the negation of more concepts.

Recursive concept combinations. We have defined the three classical symbolic operators for concept combinations. These symbolic operators can further be recursively chained on top of each other to specify more complex logical operations.

2.4 Composing Visual Concepts

In this section, we evaluate the proposed method for composing visual concepts, such as facial attributes and objects. We perform empirical studies to answer the following questions: (1) Can EBMs exhibit concept compositionality (such as concept negation, conjunction, and disjunction) in image generation? (2) Can we take advantage of concept combinations to learn new concepts in a continual manner? (3) Does explicit factor decomposition enable generalization to novel combinations of factors? (4) Can we perform concept inference using EBMs?

2.4.1 Experiment setup

Datasets. We perform experiments on the CelebA dataset [89] and the object scenes dataset rendered by MuJoCo [153]. The MuJoCo Scene images contain a central object of varying size and color at a varying position. The object shape includes "sphere", "cylinder", and "cube". The images are generated under different lighting conditions.

Implementation details. To train the energy-based models, we use the ImageNet32×32 model architecture and ImageNet128×128 model architecture from [30]. More model architecture details can be found in Appendix A.2.2.

Models trained on the Mujoco Scenes and CelebA datasets use the Adam optimizer with a learning rate of 3e-4. The first-order moment is 0.0, and the second-order moment is 0.999. The batch size is 128. The replay buffer size is $50,000$ with a 5% replacement rate. Spectral normalization is applied to models with a step size of 100

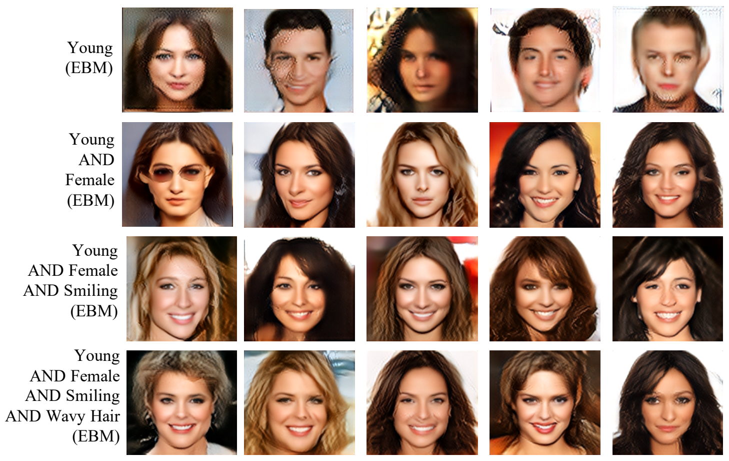

Young (EBM)
Young AND Female (EBM)
Young AND Female AND Smiling (EBM)
Young AND Female AND Smiling AND Wavy Hair (EBM)

Figure 2-2: **Concept conjunction of different concepts on CelebA.** Each row adds an additional energy function. Images on the first row are conditioned on young, while images on the last row are conditioned on young, female, smiling, and wavy hair.

for each Langevin dynamics step. We use 60 steps of Langevin sampling per training iteration for the CelebA dataset and 80 steps of Langevin sampling per training iteration for the Mujoco Scenes dataset. We use the Swish activation [124] to train the models and find that it greatly stabilizes and speeds up the training procedure.

2.4.2 Compositional generation

Qualitative evaluation. We first provide qualitative results of conjunction, disjunction, and negation operations on the CelebA and MuJoCo Scenes datasets.

Concept conjunction: In Fig. 2-2, we show that the conjunction of EBMs is able to combine multiple independent concepts, such as age, gender, smiling, and wavy hair, and get more precise generations after composing more concepts. Similarly, EBMs can combine independent concepts of shape, position, size, and color to get more precise generations in Fig. 2-3. We also show results of recursive concept combinations in Fig. 2-4. The conjunction operator can be used with other logical operators for more complex compositional visual generation.

Concept disjunction: The last row of Fig. 2-4 shows EBMs can combine concepts additively (generate images that contain at least one of the concepts). By constructing the Langevin sampling using the disjunction operator, EBMs can generate an image

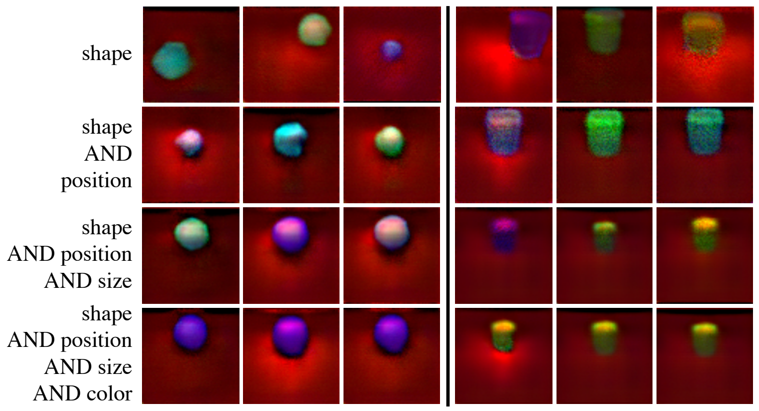

shape
shape AND position
shape AND position AND size
shape AND position AND size AND color

Figure 2-3: **Concept conjunction of different concepts on MuJoCo Scenes.** Each row adds an additional energy function. Images on the first row are only conditioned on shape, while images on the last row are conditioned on shape, position, size, and color. The left part is the generation of a sphere shape and the right is a cylinder.

that is "not smiling male" or "smiling female", where both "not smiling male" and "smiling female" are specified through the conjunction of energy-based models.

Concept negation: In Fig. 2-4, the fourth row shows images that are opposite to the trained concept using the negation operation. Since concept negation operation should be used with another concept as described in Section 2.3.2, we use "smiling" as the second concept.

Quantitative evaluation. We first evaluate the image generation quality on the MuJoCo Scenes dataset. We train a classifier to predict the object position and color on the MuJoCo Scenes dataset. The classifier is well trained with an accuracy of 99.3% accuracy for position inference and an accuracy of 99.9% for color inference. We use this pre-trained classifier to evaluate the generated images. For a given positional generation, we first use the pre-trained classifier to predict the object location in the generated image. If the distance between the predicted position and the given position is smaller than a threshold, the generation is considered correct. Similarly, a color generation is considered correct if the predicted color is the same as the given color.

In Table 2.1, we evaluate the quality of generated images given combinations of conjunction, disjunction, and negation on the color and position concepts. We use the pre-trained classifiers described above to evaluate the position accuracy

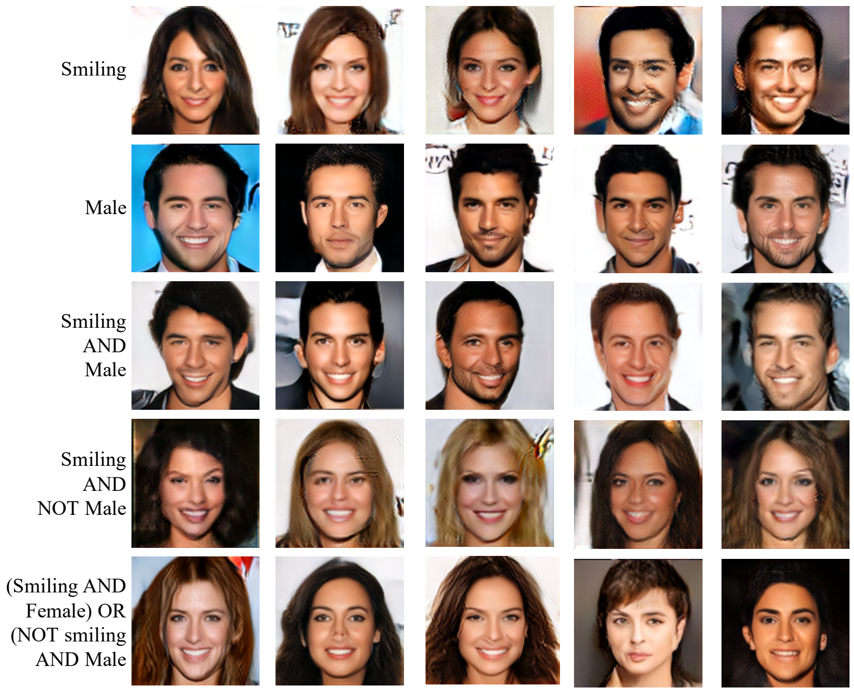


Figure 2-4: **Recursive concept compositions.** Examples of recursive compositions of disjunction, conjunction, and negation on the CelebA dataset.

Model	**Pos Accuracy (%) ↑**	**Color Accuracy (%) ↑**
ColorEBM	12.8	99.7
PosEBM	98.4	20.1
PosEBM AND ColorEBM	80.1	81.3
PosEBM AND (NOT ColorEBM)	87.2	9.6
(NOT PosEBM) AND ColorEBM	3.3	97.1
Color [157]	13.2	33.3
Pos [157]	14.6	20.2
Pos AND Color [157]	15.1	34.2

Table 2.1: **Quantitative results of EBMs and baselines**. Image generation results on the Mujoco Scenes dataset using the conjunction (AND), disjunction (OR), and negation (NOT) operators. “ColorEBM” means the energy-based model trained on the color information. “PosEBM” means the energy-based model trained on the position information.

and color accuracy of the generated images. When using a single Color EBM for image generation, the color accuracy is higher while the position accuracy is lower. Similarly, the positional accuracy is higher if we use a single Position EBM (PosEBM in Table 2.1). "PosEBM AND ColorEBM" has high accuracy on both position and color, demonstrating the conjunction operator can combine different concepts. In "PosEBM AND (NOT ColorEBM)", the color accuracy drops. This means negating a concept reduces the likelihood of that concept. "(NOT PosEBM) AND ColorEBM" shows similar results. We compare energy-based models with the approach proposed in [157] and find that EBMs have better performance.

2.4.3 Continual learning

Compositionality enables models to continually learn from the world by composing new knowledge with previously learned knowledge. However, it is still extremely challenging for existing AI models to have such continual learning ability. Our compositional energy-based models provide a possible solution for continual learning. In this part, we evaluate to what extent compositionality in EBMs enables continual learning of new concepts and their combination with previously learned concepts. If we create an EBM for a novel concept, can it be combined with previous EBMs that have never observed this concept in their training data? And can we continually repeat this process?

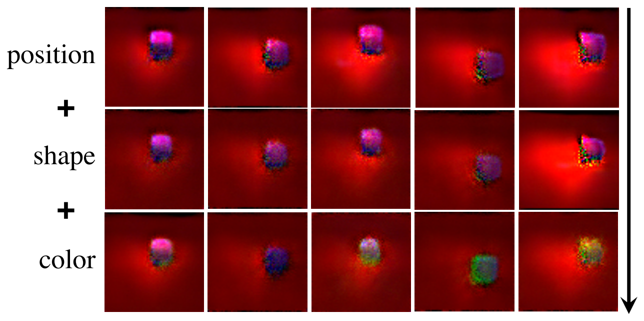


Figure 2-5: **Continual learning of concepts.** A position EBM is first trained on one shape (cube) of one color (purple) at different positions (first row). A shape EBM is then trained on different shapes of one fixed color (purple) (second row). Finally, a color EBM is trained on shapes of many colors (third row).

To answer these questions, we use the following methodology on the MuJoCo Scenes dataset: 1) We first train a position EBM on a dataset of objects in different positions, but with a fixed color and a fixed shape. In this experiment, we use the shape "cube" and the color "purple". The position EBM allows us to generate a purple

Table 2.2: **Quantitative evaluation of continual learning.** A position EBM is first trained on "purple" "cubes" at different positions. A shape EBM is then trained on different "purple" shapes. Finally, a color EBM is trained on shapes of many colors with earlier EBMs fixed. We compare EBMs with a GAN model [120] which is trained on the same position, shape, and color dataset. EBMs are better at continually learning new concepts and remembering old concepts (Acc means accuracy).

Model	**Position Acc (%) ↑**	**Shape Acc (%) ↑**	**Color Acc (%) ↑**
EBM (Position)	90.1	-	-
EBM (Position + Shape)	81.3	74.3	-
EBM (Position + Shape + Color)	78.1	70.3	52.1
GAN (Position)	94.1	-	-
GAN (Position + Shape)	11.1	97.7	-
GAN (Position + Shape + Color)	11.7	47.6	98.4

cube at various positions. (first row in Fig. 2-5). 2) Next, we train a shape EBM by training the model in combination with the position EBM to generate images of different shapes at different positions, but without training the position EBM. As shown in the second row of Fig. 2-5, after combining the position and shape EBMs, the "spheres" are placed in the same position as the "cubes" in the first row, even though these "spheres" never appear in these positions during training. 3) Finally, we train a color EBM in combination with both position and shape EBMs to generate images of different shapes and colors at different positions. Again, we fix both position and shape EBMs, and only train the color model. In the third row, the objects with different colors have the same position as the objects in the first row and the same shape as the objects in the second row. Continual learning is still a challenging research problem. Our results indicate that EBMs have the potential to enable continual learning of new concepts and compose them with previously learned concepts to generate new images.

In Table 2.2, we compare the continual learning ability of our EBMs and GAN [120]. Similar to the evaluation method used in Section 2.4.2, we train classifiers for position, shape, and color, respectively, to evaluate the quality of generated images. For a fair comparison, the GAN model is also trained sequentially on the position, shape, and color datasets.

The position accuracy of EBM does not drop significantly when continually learning new concepts (shape and color), indicating EBMs are able to extrapolate earlier learned

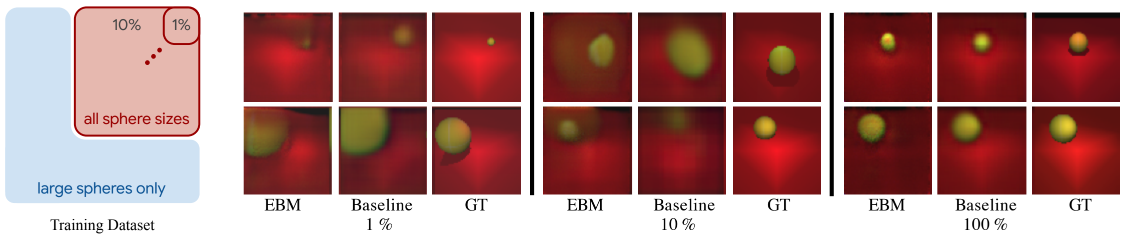


Figure 2-6: **Cross product extrapolation.** Left: the spheres of all sizes only appear in the top right corner (1%, 10%, . . .) of the scene and the remaining positions only have large size spheres. Right: generated images of novel size and position combinations using EBM and the baseline model.

concepts by combining them with newly learned concepts. In contrast, while the GAN models are able to learn different concepts given the corresponding data, their accuracy on position and shape drops significantly after learning the color concept. The bad performance shows that GAN models are bad at combining the newly learned attributes with the previous attributes.

2.4.4 Cross product extrapolation

Humans are endowed with the ability to extrapolate novel concept combinations when only a limited number of combinations were originally observed. For example, despite never having seen a “purple cube”, a human can imagine what it looks like based on the previous observation of “red cube” and “purple sphere”.

To evaluate the extrapolation ability of EBMs, we construct a new dataset of MuJoCo scene images with spheres of all possible sizes appearing only in the top right corner of the scene, and spheres of only large sizes appearing in the remaining positions as shown in the left part of Fig. 2-6. For spheres only in the top right corner of the scene, we design different settings. For example, 1% means only 1% of positions (starting from the top right corner) that contain all sphere sizes are used for training. At test time, we evaluate the generation of spheres of all sizes at positions that are not seen during training. Similar to 1%, 10% and 100% mean the spheres of all sizes appear only in the top right 10% and 100% of the scene, respectively. The task is to test the quality of generated objects with unseen size and position combinations.

This requires the model to extrapolate the learned position and size concepts in novel combinations.

We train two EBMs on this dataset. One is conditioned on the position information and trained only on large sizes. The second one is conditioned on the size information and trained at the aforementioned percentage of positions. The conjunction of the two EBMs is fine-tuned for generation through gradient descent. We compare this composed model with a baseline holistic model that is trained on both position and size information jointly. The baseline is trained on the same position and size combinations and optimized directly from the Mean Squared Error between the generated images and the ground truth image. For fair comparisons, the baseline model and our EBMs use a similar number of model parameters. The model details are described in Appendix A.2.2.

Qualitative evaluation. We qualitatively compare the EBM and baseline in Fig. 2-6. When spheres of all sizes are only distributed in the 1% of possible locations, both the EBM and baseline have bad performance. This is because the few combinations of sizes and positions make both models fail in extrapolation. For the 10% setting, our EBM is better than the baseline. EBM is able to combine concepts to form images from a few combination examples by learning an independent model for each concept factor. Both EBM and baseline models generate accurate images when given examples of all combinations (100% setting), but our EBM is closer to the ground truth than the baseline.

Quantitative evaluation. In Fig. 2-7, we quantitatively evaluate the extrapolation ability of EBM and the baseline. We train a regression model that outputs both the position and size of a generated image. We compute the error between the predicted size and ground truth size and report it in the left part of Fig. 2-7. Similarly, we report the position error in the right part. EBMs are able to extrapolate both position and size better than the baseline model with smaller errors. The size errors go down given more examples of all sphere sizes. For position error, both EBM and the baseline model have smaller errors at 1% data than 5% or 10% data. This result is due to the

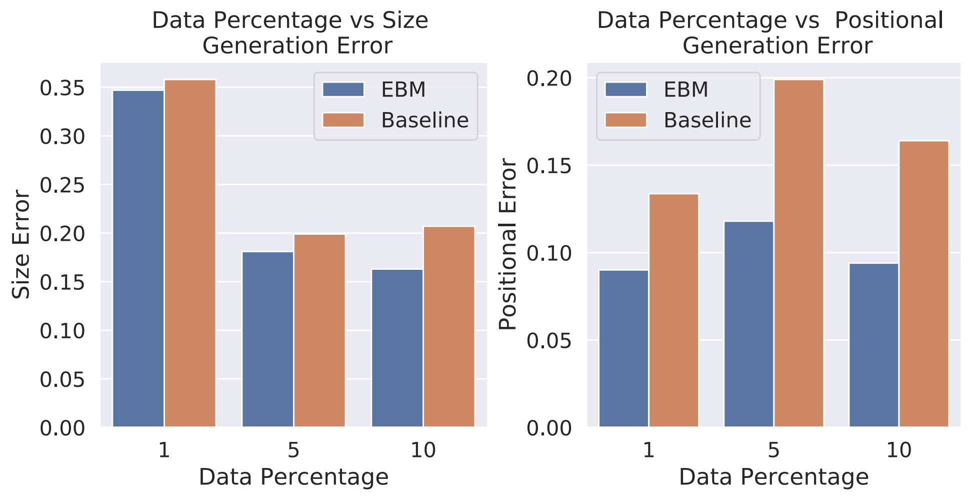


Figure 2-7: **Quantitative evaluation of cross product extrapolation.** Cross product extrapolation results with respect to the percentages of areas in the top right corner. EBM has smaller size and position errors, which means EBM is able to extrapolate better than the baseline model.

make-up of the data – with 1% data, only 1% of the rightmost sphere positions have different size annotations, so the models generate large spheres at the conditioned position, which are closer to the ground truth position since most positions (99%) are large spheres.

2.4.5 Concept inference

The EBM formulation also allows us to infer concepts from the generated images. For example, a positional EBM takes an image and an object's 2D position (x,y) as input and outputs an energy value. During inference, we iterate through all the possible positions (20 by 20 grid of positions) and select the position with minimal energy as the inference result. We evaluate this result by computing the Mean Absolute Error between the predicted position and the ground truth object position.

To evaluate the inference ability of EBMs, we generate a new MuJoCo Scene dataset for training and testing. In the training set, each scene has varying lighting conditions with one object, either sphere or cube, at all possible positions and some sizes. We build several test sets to evaluate the generalization ability of different models. The easiest one is *"Test"*, which has the same data distribution as the training

Table 2.3: **Position errors on different test datasets.** "Test" has the same data distribution as the training set. Other datasets change one environmental parameter, e.g., color, size, type, and light, which are unseen in the training set. "Avg" is the average error of "Color", "Light", "Size", and "Type". "Steps" indicates the number of sampling steps used to train EBMs. EBMs are able to generalize better on unseen datasets. The larger number of sampling steps significantly decreases overall errors.

Model	**Steps**	**Color** $\downarrow$	**Light** $\downarrow$	**Size** $\downarrow$	**Type** $\downarrow$	**Avg** $\downarrow$	**Test** $\downarrow$
Resnet [43]	-	20.002	5.881	10.378	6.310	10.643	3.635
PixelCNN [108]	-	60.607	58.589	33.889	48.138	50.306	43.460
EBM	200	10.899	6.307	8.431	6.304	7.985	3.903
EBM	400	**4.084**	**4.033**	**6.853**	**3.694**	**4.666**	**2.917**

dataset. The *"Size"* test set contains objects twice larger than objects in the training set. The *"Color"* set has object colors that have never been seen during training. *"Light"* is a test set with different lighting sources. The *"Type"* test set consists of cylinder images while the training images only contain spheres or cubes.

We compare EBMs with two baseline models, ResNet [43] (with the same model architecture as EBM) and PixelCNN [108]. Table 2.3 shows the comparison results using a different number of Langevin sampling steps. We find that inference in EBMs is able to generalize well to different test sets that are outside the training distribution, such as Color, Light, Size, and Type. A larger number of Langevin sampling steps further improves the generalization ability.

2.5 Composing Visual Relations

We have shown that the proposed compositional operators enable EBMs to compose concepts and objects and generate images that are outside the training distribution in Section 2.4. However, the visual world around us contains not only concepts and objects but also their associated relations. To better understand the visual world around us, we study composing visual relations between objects in this section.

While there has been significant work on designing deep neural networks that may compose individual objects together, less work has been done on composing the individual relations between objects. A principal difficulty is that while the placement of objects is mutually independent, their relations are entangled and dependent on each other. To circumvent this issue, existing works primarily compose relations by utilizing a holistic encoder, in the form of text or graphs. In this section, we instead propose to represent each relation as an unnormalized density (an energy-based model), enabling us to compose separate relations in a factorized manner. We show that such a factorized decomposition allows the model to both generate and edit scenes that have multiple sets of relations more faithfully. We further show that decomposition enables our model to effectively understand the underlying relational scene structures (Figure 2-8).

2.5.1 Learning relational energy functions

Given a training dataset $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{s}_i\}_{i=1}^N$ with N distinct images and associated relational descriptions, we aim at learning the underlying probability distribution $p_\theta(\boldsymbol{x}|\boldsymbol{s})$ — the probability distributions of an image $\boldsymbol{x}$ given the corresponding relational descriptions $\boldsymbol{s}$. The relational descriptions $\boldsymbol{s}$ contain a set of object relations $\{\boldsymbol{r}_1 \cdots, \boldsymbol{r}_n\}$. To represent $p_\theta(\boldsymbol{x}|\boldsymbol{s})$, we model each component relation separately using a probability distribution $p_\theta(\boldsymbol{x}|\boldsymbol{r}_i)$ which is represented as an energy-based model. The overall scene probability distribution is then modeled by a composition of individual probability distributions of the object relations $p_\theta(\boldsymbol{x}|\boldsymbol{s}) \propto \prod_{i=1}^n p_\theta(\boldsymbol{x}|\boldsymbol{r}_i)$.

In Section 2.3.2, we show that EBMs enable us to naturally compose separate probability distributions. Given a scene relational description $\boldsymbol{r}_i$, we seek to learn a conditional EBM to model the underlying probability distribution $p_\theta(\boldsymbol{x}|\boldsymbol{r}_i)$:

$$p_\theta(\boldsymbol{x}|\boldsymbol{r}_i) \propto e^{-E_\theta(\boldsymbol{x}|\boldsymbol{r}_i)}, \tag{2.11}$$

where $p_\theta(\boldsymbol{x}|\boldsymbol{r}_i)$ represents the probability distribution over images given relation $\boldsymbol{r}_i$.

The most straightforward manner of encoding relational scene descriptions is to encode the entire sentence using an existing text encoder, such as CLIP [119]. However, we find that such an approach cannot capture scene relations accurately. An issue

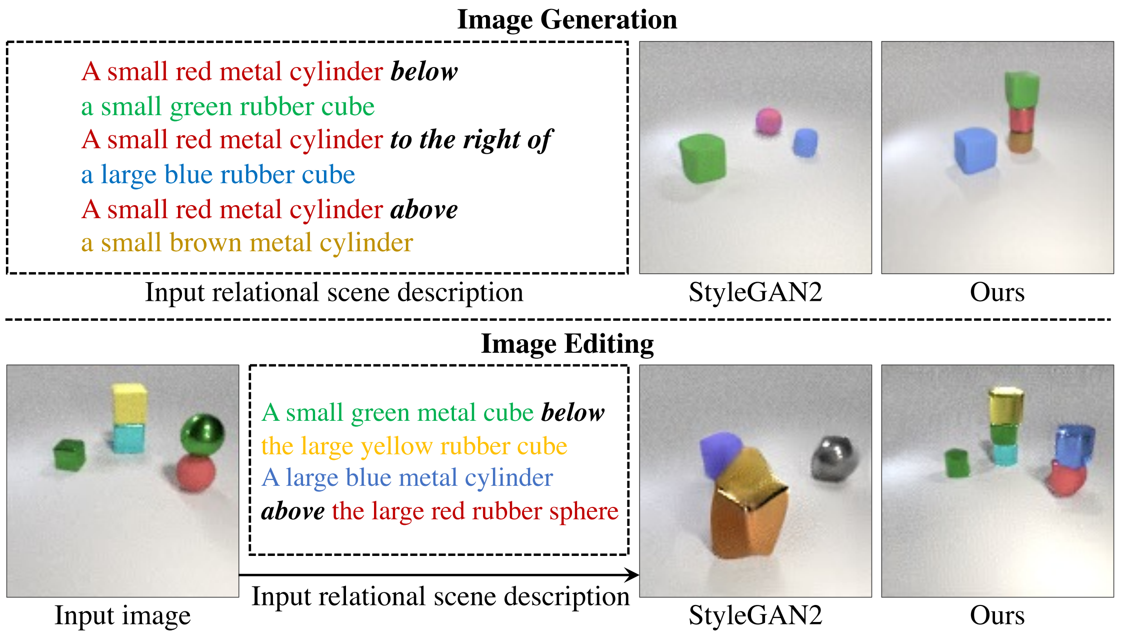


Image Generation
A small red metal cylinder ***below***
a small green rubber cube
A small red metal cylinder ***to the right of***
a large blue rubber cube
A small red metal cylinder ***above***
a small brown metal cylinder
Input relational scene description
StyleGAN2
Ours
Image Editing
A small green metal cube ***below***
the large yellow rubber cube
A large blue metal cylinder
above the large red rubber sphere
Input image
Input relational scene description
StyleGAN2
Ours

Figure 2-8: **Composing visual relations.** The proposed method can generate and edit images with multiple composed relations. **Top**: Image generation results based on relational scene descriptions. **Bottom**: Image editing results based on relational scene descriptions.

with such an approach is that the sentence encoder loses or masks the information captured by the relation tokens in $\boldsymbol{r}_i$.

To enforce that the underlying relation tokens in $\boldsymbol{r}_i$ are effectively encoded, we instead propose to decompose the relation $\boldsymbol{r}_i$ into a relation triplet (r_i', o_i^1, o_i^2), where r_i' is the relation token, *e.g.*, "below" and "to the right of", o_i^1 is the description of the first object, and o_i^2 is the description of the second object appeared in $\boldsymbol{r}_i$. Each entry in the relation triplet is then separately encoded.

Such an encoding scheme encourages the models to encode underlying objects and relations in a scene, enabling us to effectively model the relational distributions. We explored two separate approaches to encode the underlying object descriptions.

CLIP embedding. We use the pre-trained CLIP model to encode objects and a learned embedding layer to encode their relations. Taking the scene description of "a large blue rubber cube to the left of a small red metal cube" as an example, we use the pre-trained CLIP model to encode the two objects separately, *i.e.*, o_i^1 for "a large blue rubber cube" and o_i^2 for "a small red metal cube". We then use an embedding

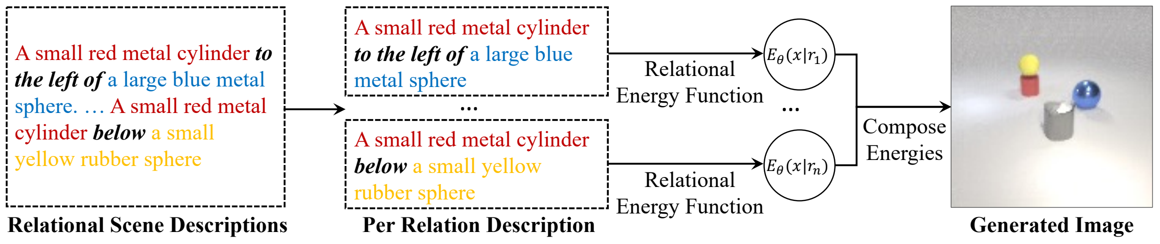

A small red metal cylinder *to the left of* a large blue metal sphere. … A small red metal cylinder *below* a small yellow rubber sphere
Relational Scene Descriptions
A small red metal cylinder *to the left of* a large blue metal sphere
…
A small red metal cylinder *below* a small yellow rubber sphere
Per Relation Description
Relational Energy Function
$E_\theta(x|r_1)$
…
Relational Energy Function
$E_\theta(x|r_n)$
Compose Energies
Generated Image

Figure 2-9: **Overview of our pipeline for understanding a relational scene description.** A relational scene description is split into a set of underlying object relation descriptions. Individual relation descriptions are represented as EBMs which are subsequently composed together to generate images.

layer to encode their relation, *i.e.*, r_i' for "to the left". The features of the first and second objects and their relations are concatenated and used as the feature of the relational scene description, which is further sent to the relational energy functions for image generation or image editing.

Learned embedding. We use the learned embedding layers for both objects and their relations. To encode an object, we use six different embedding layers to encode its color, size, material, shape, relation, and position separately. The embedded features of objects and their relations are concatenated and used as the feature of the relational scene description, which is further sent to the relational energy functions.

2.5.2 Composing relational energy functions

Given an underlying scene description $\boldsymbol{s}$, we represent the underlying probability distribution $p(\boldsymbol{x}|\boldsymbol{s})$ by factorizing it as a product of probabilities over the object relations $\boldsymbol{r}_i$ described in $\boldsymbol{s}$. Given the separate relational energy functions learned in Section 2.5.1, this probability $p(\boldsymbol{x}|\boldsymbol{s})$ can be written as:

$$p(\boldsymbol{x}|\boldsymbol{s}) = e^{-E_\theta(\boldsymbol{x}|\boldsymbol{s})} \propto \prod_{i=1}^{n} p(\boldsymbol{x}|\boldsymbol{r}_i) = e^{-\sum_{i=1}^{n} E_\theta(\boldsymbol{x}|\boldsymbol{r}_i)}. \tag{2.12}$$

The overview of the proposed method is shown in Figure 2-9.

2.5.3 Experiment setup

We conduct empirical studies to answer the following questions: (1) Can we learn relational models that generate and edit complex multi-object scenes given relational scene descriptions? (2) Can we use our model to generalize to scenes that are never seen in training? (3) Can we understand the set of relations in a scene and infer semantically equivalent descriptions?

To answer these questions, we evaluate the proposed method and baselines on image generation, image editing, and image classification on two main datasets, *i.e.*, Relational CLEVR [63] and iGibson [140]. We also test the image generation performance of the proposed model and baselines on a real-world dataset *i.e.*, Blocks [81].

Datasets. *Relational CLEVR.* We use $50,000$ pairs of images and relational scene descriptions for training. Each image contains $1 \sim 5$ objects. Each object consists of five different attributes, including color, shape, material, size, and its spatial relation to another object in the same image. There are 9 types of colors, 4 types of shapes, 3 types of materials, 3 types of sizes, and 6 types of relations.

iGibson. On the iGibson dataset, we use $30,000$ pairs of images and relational scene descriptions for training. Each image contains $1 \sim 3$ objects. Each object consists of the same five different types of attributes as the Relational CLEVR dataset. There are 6 types of colors, 5 types of shapes, 4 types of materials, 2 types of sizes, and 4 types of relations. The objects are randomly placed in the scenes.

Blocks. On the real-world Blocks dataset, a number of $3,000$ pairs of images and relational scene descriptions are used for training. Each image contains $1 \sim 4$ objects with different colors. We only consider the "above" and "below" relations as objects are placed vertically in the form of towers.

In the training set, each image's relational scene description only contains one scene relation, and objects are randomly placed in the scene. We generated three test subsets that contain relational scene descriptions with a different number of scene relations to test the generation ability of the proposed methods and baselines. The

Table 2.4: **Quantitative evaluation of image generation and image editing.** The accuracy of object relations in the generated images or edited images on the Relational CLEVR and iGibson datasets. We compare our method with baselines on three test sets, *i.e.* *1R*, *2R*, and *3R* (see text).

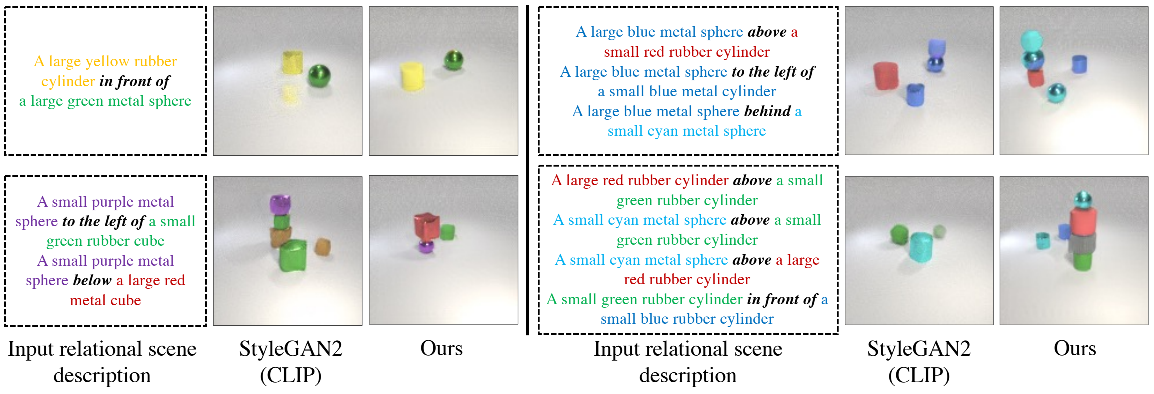
Dataset	**Model**	**Image Generation (%)**			**Image Editing (%)**		
		1R Acc ↑	2R Acc ↑	3R Acc ↑	1R Acc ↑	2R Acc ↑	3R Acc ↑
Relational CLEVR	StyleGAN2	10.68	2.46	0.54	10.04	2.10	0.46
	StyleGAN2 (CLIP)	65.98	9.56	1.78	-	-	-
	Ours (CLIP)	94.79	48.42	18.00	95.56	52.78	16.32
	Ours (Learned Embed)	**97.79**	**69.55**	**37.60**	**97.52**	**65.88**	**32.38**
iGibson	StyleGAN2	12.46	2.24	0.60	11.04	2.18	0.84
	StyleGAN2 (CLIP)	49.20	17.06	5.10	-	-	-
	Ours (CLIP)	74.02	43.03	**19.59**	78.12	32.84	12.66
	Ours (Learned Embed)	**78.27**	**45.03**	19.39	**84.16**	**44.10**	**20.76**

1R test subset is similar to the training set where each relational scene description contains one scene relation. The *2R* and *3R* test subsets have two and three scene relations in each relational scene description, respectively. Each test set has $5,000$ images with corresponding relational scene descriptions.

Baselines. We compare our method with two baseline approaches. The first baseline is StyleGAN2 [65], one of the state-of-the-art methods for unconditional image generation. To enable StyleGAN2 to generate images and edit images based on relational scene descriptions, we train a ResNet-18 classifier on top of it to predict the object attributes and their relations. Recently, CLIP [119] has achieved a substantial improvement in the text-image retrieval task by learning text-image feature embeddings on large-scale datasets. We thus design another baseline, StyleGAN2+CLIP, that combines the capabilities of both approaches. To do this, we encode relational scene descriptions into text embeddings using CLIP and condition StyleGAN2 on the embeddings to generate images.

2.5.4 Image generation results

Given a relational scene description, *e.g.*, "a blue cube on top of a red sphere", we aim to generate images that contain corresponding objects and their relations as described in the given descriptions.



A large yellow rubber cylinder *in front of* a large green metal sphere
A small purple metal sphere *to the left of* a small green rubber cube
A small purple metal sphere *below* a large red metal cube
Input relational scene description
StyleGAN2 (CLIP)
Ours
A large blue metal sphere *above* a small red rubber cylinder
A large blue metal sphere *to the left of* a small blue metal cylinder
A large blue metal sphere *behind* a small cyan metal sphere
A large red rubber cylinder *above* a small green rubber cylinder
A small cyan metal sphere *above* a small green rubber cylinder
A small cyan metal sphere *above* a large red rubber cylinder
A small green rubber cylinder *in front of* a small blue rubber cylinder
Input relational scene description
StyleGAN2 (CLIP)
Ours

Figure 2-10: **Image generation results on the Relational CLEVR dataset.** Images are generated based on $1 \sim 4$ relational descriptions. Note that the models are trained on a single relational description and the composed scene relations (2, 3, and 4 relational descriptions) are outside the training distribution.

Quantitative comparisons. To evaluate the quality of generated images, we train a binary classifier to predict whether the generated image contains objects and their relations described in the given relational scene description.

Given an image and a relational scene description, we first feed the image to convolutional layers to generate an image feature, and then send the relational scene description to an embedding layer, followed by fully connected layers to generate a relational scene feature. The image feature and relational scene feature are combined and then passed through fully connected layers and a sigmoid function to predict whether the given image matches the relational scene description. The binary classifier is trained on real images from the training dataset. We train a classifier on each dataset. The classification accuracy on real images is close to 100%, indicating that the classifier is effective. During testing, we generate an image based on a relational scene description and send the generated image and the relational scene description to the classifier for prediction. For a fair comparison, we use the same classifier to evaluate images generated by all the approaches on each dataset.

The "Image Generation" column in Table 2.4 shows the classification results of different approaches on the Relational CLEVR and iGibson datasets. On each dataset, we test each method on three test subsets, *i.e.*, *1R*, *2R*, and *3R*, and report their binary classification accuracies. Both variants of our proposed approach outperform

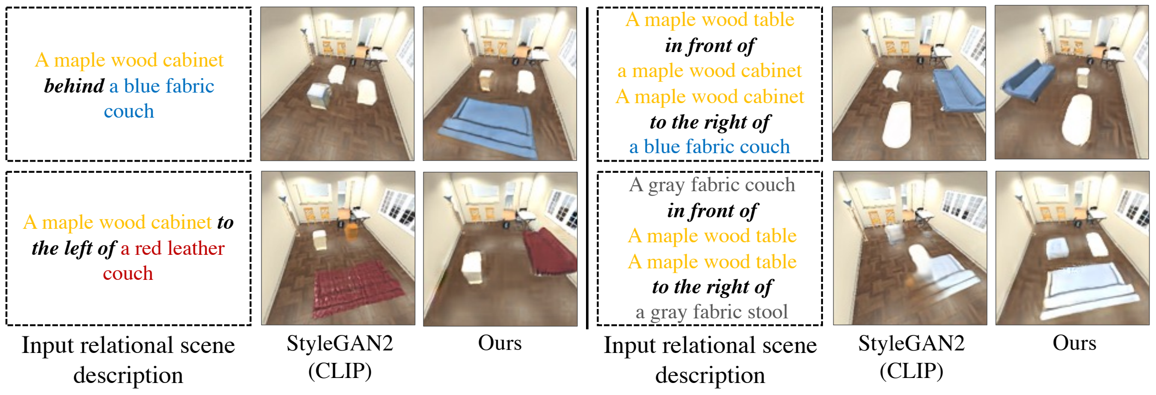

A maple wood cabinet *behind* a blue fabric couch
A maple wood cabinet *to the left of* a red leather couch
Input relational scene description
StyleGAN2 (CLIP)
Ours
A maple wood table *in front of* a maple wood cabinet
A maple wood cabinet *to the right of* a blue fabric couch
A gray fabric couch *in front of* A maple wood table
A maple wood table *to the right of* a gray fabric stool
Input relational scene description
StyleGAN2 (CLIP)
Ours

Figure 2-11: **Image generation results on the iGibson dataset.** Images are generated based on $1 \sim 2$ relational descriptions. Note that the two composed scene relations are outside the training distribution.

StyleGAN2 and StyleGAN2 (CLIP), indicating that our method can generate images that contain the objects and their relations described in the relational scene descriptions. We find that our approach using the learned embedding, *i.e.*, Ours (Learned Embed), achieves better performances on the Relational CLEVR and iGibson datasets than the other variant using the CLIP embedding, *i.e.*, Ours (CLIP).

StyleGAN2 and StyleGAN2 (CLIP) can perform well on the *1R* test subset. This is an easier test subset because the models are trained on images with a single scene relation, and the models generate images based on a single relational scene description during testing as well. The *2R* and *3R* are more challenging test subsets because the models need to generate images conditioned on relational scene descriptions of multiple scene relations. Our models outperform the baselines by a large margin, indicating the proposed approach has a better generalization ability and can compose multiple relations that are never seen during training.

Human evaluation results. To further evaluate the performance of the proposed method on image generation, we conduct a user study to ask humans to evaluate whether the generated images match the given input scene description. We compare the correctness of the object relations in the generated images and the input language. Given a language description, we generate an image using “Ours (Learned Embed)” and “StyleGAN2 (CLIP)”. We shuffle these two generated images and ask the workers

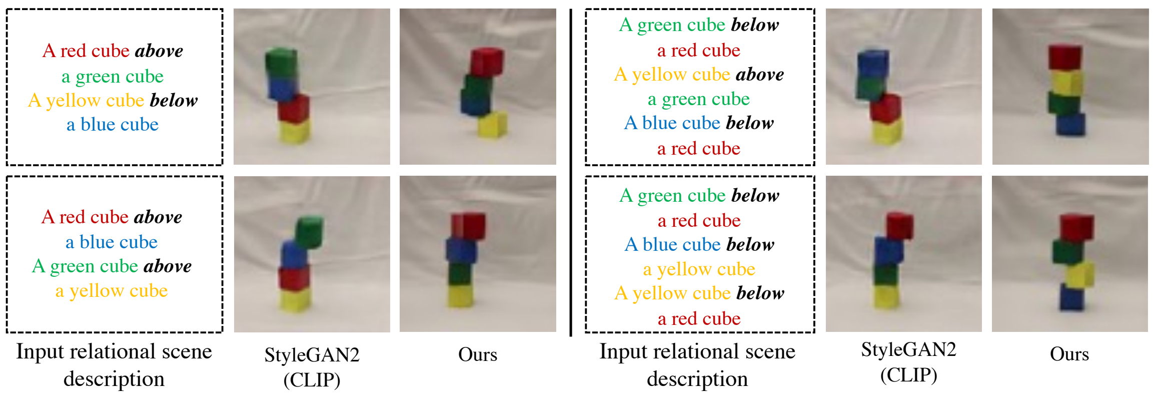

A red cube *above* a green cube
A yellow cube *below* a blue cube
A red cube *above* a blue cube
A green cube *above* a yellow cube
A green cube *below* a red cube
A yellow cube *above* a green cube
A blue cube *below* a red cube
A green cube *below* a red cube
A blue cube *below* a yellow cube
A yellow cube *below* a red cube
Input relational scene description
StyleGAN2 (CLIP)
Ours
Input relational scene description
StyleGAN2 (CLIP)
Ours

Figure 2-12: **Image generation results on the Blocks dataset.** Images are generated based on the relational scene description. Note that the models are trained on a single relational scene description and the composed scene relations are outside the training distribution.

to tell which image has better quality and the object relations match the input language description. We tested 300 examples in total, including 100 examples with a single-sentence relational description (1R), 100 examples with two-sentence relational descriptions (2R), and 100 examples with three-sentence relational descriptions (3R). There are 32 workers involved in this human experiment.

The workers think that there are 87%, 86%, and 91% of generated examples that "Ours (Learned Embed)" is better than "StyleGAN2 (CLIP)" for *1R*, *2R*, and *3R*, respectively. The human experiment shows that our proposed method is better than "StyleGAN2 (CLIP)". The conclusion is coherent with our binary classification evaluation results.

Qualitative comparisons. The image generation results on Relational CLEVR, iGibson, and Block scenes are shown in Figure 2-10, 2-11, and 2-12, respectively. We show examples of generated images conditioned on relational scene descriptions of different numbers of scene relations. Our method generates images that are consistent with the relational scene descriptions. Note that both the proposed method and the baselines are trained on images that only contain a single scene relation describing the visual relationship between two objects. Our approach can generalize well when composing more visual relations. Taking the upper right figure in Figure 2-10 as an example, a relational scene description of multiple scene relations, *i.e.*, “A large blue metal sphere above a small red rubber cylinder. A large blue metal sphere to the left of a small blue metal cylinder, $\cdots$”, is never seen during training. “StyleGAN2 (CLIP)” generates wrong objects and scene relations. In contrast, our method has the ability to generalize to novel combinations and generate correct images.

2.5.5 Image editing results

Given an input image, we aim to edit this image based on relational scene descriptions, such as replacing “put a red cube in front of the blue cylinder” with “put a red cube behind the blue cylinder”.

Quantitative comparisons. Similar to image generation, we use a classifier to predict whether the image after editing contains the objects and their relations described in the relational scene descriptions. For the evaluation on each dataset, we use the same classifier for both image generation and image editing.

The “Image Editing” column in Table 2.4 shows the classification results of different approaches on the Relational CLEVR and iGibson datasets. Both variants of our proposed approach, *i.e.*, “Ours (CLIP)” and “Ours (Learned Embed)” outperform the baselines, *i.e.*, “StyleGAN2” and “StyleGAN2 (CLIP)”, substantially. The high performance of our approach on the *2R* and *3R* test subsets shows that the proposed method is able to generalize to relational scene descriptions that are outside the training distribution.

Qualitative comparisons. We show image editing examples in Figure 2-13. The left part is image editing results conditioned on a single scene relation, while the right part is conditioned on two scene relations. We show examples that edit images by inverting individual spatial relations between two objects. Taking the first image in Figure 2-13 as an example, “the small purple metal sphere” is behind “the large yellow rubber sphere”, after editing, our model can successfully put “the small purple metal sphere” in front of “the large yellow rubber sphere”. Even for relational scene descriptions of two object relations that are never seen during training, our model can

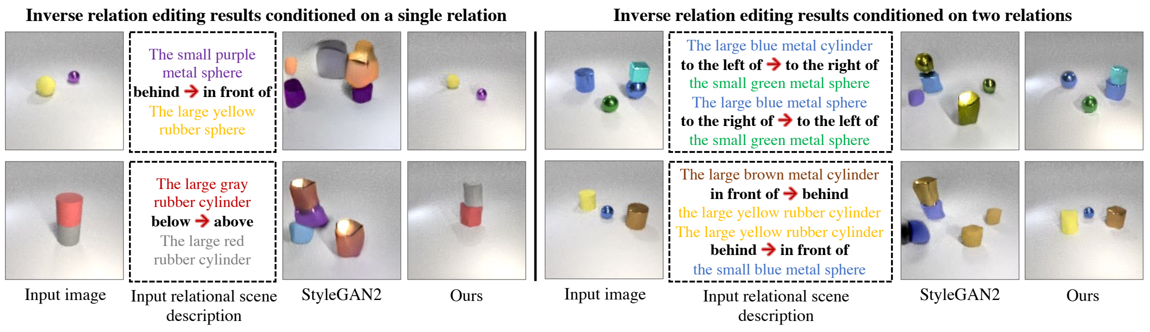

Inverse relation editing results conditioned on a single relation
Inverse relation editing results conditioned on two relations
The small purple metal sphere
behind → in front of
The large yellow rubber sphere
The large gray rubber cylinder
below → above
The large red rubber cylinder
The large blue metal cylinder
to the left of → to the right of
the small green metal sphere
The large blue metal sphere
to the right of → to the left of
the small green metal sphere
The large brown metal cylinder
in front of → behind
the large yellow rubber cylinder
The large yellow rubber cylinder
behind → in front of
the small blue metal sphere
Input image
Input relational scene description
StyleGAN2
Ours

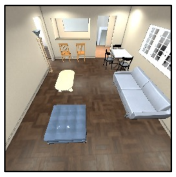
Figure 2-13: **Image editing results on the Relational CLEVR dataset. Left**: image editing results based on a single relational scene description. **Right**: image editing results based on two composed relational scene descriptions. Note that the composed scene relations in the right part are outside the training distribution. Our approach can still edit the images accurately.

edit images so that the selected objects are placed correctly.

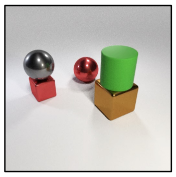
2.5.6 Relational understanding

We hypothesize the good generation performance of our proposed approach is due to our system's understanding of relations and ability to distinguish between different relational scene descriptions. In this section, we evaluate the relational understanding ability of our proposed method and baselines by comparing their image-to-text retrieval and semantic equivalence results.

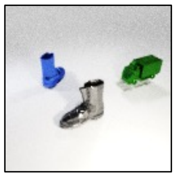
Image-to-text retrieval. In Figure 2-14, we evaluate whether our proposed model can understand different relational scene descriptions by image-to-text retrieval. We create a test set that contains 240 pairs of images and relational scene descriptions. Given a query image, we compute the similarity of this image and each relational scene description in the gallery set. The top 1 retrieved relational scene description is shown in Figure 2-14. We compare our method with two baselines. We use the pre-trained CLIP model and test it on our dataset directly. "Fine-tuned CLIP" means the CLIP model is fine-tuned on our dataset. Even though CLIP has shown good performance on the general image-text retrieval task, we find that it cannot understand spatial relations well, while EBMs can retrieve all the ground truth descriptions.

Query image	CLIP	Fine-tuned CLIP	Ours
	•A maple wood coffee table ***on the right of*** a gray fabric couch ✗ •A gray fabric couch ***on the left of*** a maple wood coffee table ✗ •A maple wood coffee table ***in front of*** a blue fabric stool ✗	•A maple wood coffee table ***on the left*** a gray fabric couch ✓ •A gray fabric couch ***behind*** a blue fabric stool ✗ •A blue fabric stool ***in front of*** a maple wood coffee table ✓	•A maple wood coffee table ***on the left of*** a gray fabric couch ✓ •A gray fabric couch ***on the right of*** a blue fabric stool ✓ •A blue fabric stool ***in front of*** a maple wood coffee table ✓

(a) Top 1 image-text retrieval result on iGibson scenes.

Query image	CLIP	Fine-tuned CLIP	Ours
	•A large gray metal sphere ***on the left of*** a small red metal cube ✗ •A small red metal cube ***on the right of*** a large brown metal cube ✗ •A large brown metal cube ***below*** a large green rubber cylinder ✓	•A large gray metal sphere ***above*** a small red metal cube ✓ •A small red metal cube ***behind*** a large brown metal cube ✓ •A large brown metal cube ***below*** a large green rubber cylinder ✓	•A large gray metal sphere ***above*** a small red metal cube ✓ •A small red metal cube ***on the left of*** a large brown metal cube ✓ •A large brown metal cube ***below*** a large green rubber cylinder ✓

(b) Top 1 image-text retrieval result on CLEVR scenes.

Query image	CLIP	Fine-tuned CLIP	Ours
	•A blue object ***in front of*** a gray object ✗ •A gray object ***on the left of*** a green object ✓ •A green object ***behind*** a blue object ✗	•A blue object ***in front of*** a gray object ✗ •A gray object ***behind*** a green object ✗ •A green object ***on the left of*** a blue object ✗	•A blue object ***behind*** a gray object ✓ •A gray object ***on the left of*** a green object ✓ •A green object ***on the right of*** a gray object ✓

(c) Top 1 image-text retrieval result on Blender scenes (outside the training distribution).

Figure 2-14: **Image-to-text retrieval results.** We compare the proposed approach with the pretrained CLIP and fine-tuned CLIP and show their top-1 retrieved relation description based on the given image query.

We also find that our approach generalizes across datasets. In the bottom row of Figure 2-14, we conduct an additional image-to-text retrieval experiment on the Blender [18] scenes that are never seen during training. Our approach can still find the correct relational scene description for the query image.

Can we understand semantically equivalent relational scene descriptions? Given two relational scene descriptions describing the same image but in different ways, can our approach understand that the descriptions are semantically similar or equivalent? To evaluate this, we create a test subset that contains 5,000 images. Each image has 3 different relational scene descriptions. There are two relational scene descriptions that match the image but describe the image in different ways, such as "a cabinet in front of a couch" and "a couch behind a cabinet". There is one further description that does not match the image. The relative score difference between the two ground truth relational scene descriptions should be smaller than the difference between one ground truth relational scene description and one wrong relational scene

Table 2.5: **Quantitative evaluation of semantic equivalence on the Relational CLEVR dataset.**

Model	Semantic Equivalence (%)		
	1R Acc ↑	2R Acc ↑	3R Acc ↑
Classifier	52.82	27.76	14.92
CLIP	37.02	14.40	5.52
CLIP (Fine-tuned)	60.02	35.38	20.9
Ours (CLIP)	70.68	50.48	38.06
Ours (Learned Emb)	**74.76**	**57.76**	**44.86**

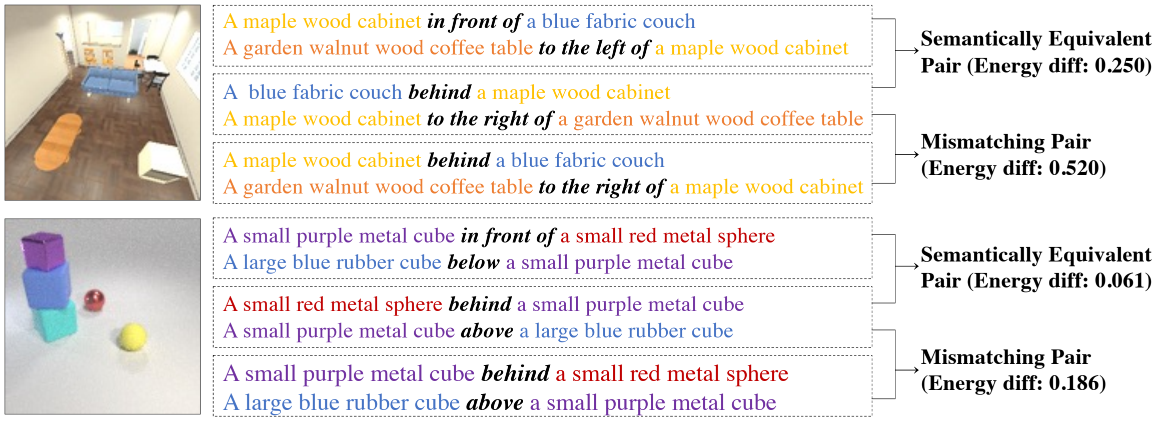

A maple wood cabinet *in front of* a blue fabric couch
A garden walnut wood coffee table *to the left of* a maple wood cabinet
A blue fabric couch *behind* a maple wood cabinet
A maple wood cabinet *to the right of* a garden walnut wood coffee table
A maple wood cabinet *behind* a blue fabric couch
A garden walnut wood coffee table *to the right of* a maple wood cabinet
Semantically Equivalent Pair (Energy diff: 0.250)
Mismatching Pair (Energy diff: 0.520)
A small purple metal cube *in front of* a small red metal sphere
A large blue rubber cube *below* a small purple metal cube
A small red metal sphere *behind* a small purple metal cube
A small purple metal cube *above* a large blue rubber cube
A small purple metal cube *behind* a small red metal sphere
A large blue rubber cube *above* a small purple metal cube
Semantically Equivalent Pair (Energy diff: 0.061)
Mismatching Pair (Energy diff: 0.186)

Figure 2-15: **Examples of semantic equivalence on Relational CLEVR and iGibson scenes.** Given an input image, our approach is able to recognize whether the relational scene descriptions are semantically equivalent or not.

description.

We compare our approach with three baselines. For each model, given an image, if the difference between two semantically equivalent relational scene descriptions is smaller than the difference between the semantically different ones, we will classify it as correct. We compute the percentage of correct predictions and show the results in Table 2.5. Our proposed method outperforms the baselines substantially, indicating that our EBMs can distinguish semantically equivalent relational scene descriptions and semantically nonequivalent relational scene descriptions. In Figure 2-15, we further show two examples generated by our approach on the iGibson and Relational CLEVR datasets. The energy difference between the semantically equivalent relational scene descriptions is smaller than the mismatching pairs.

2.5.7 Zero-shot generalization across datasets

We find that our method can generalize across datasets as shown in the third example in Figure 2-14. To quantitatively evaluate the generalization ability across datasets of the proposed method, we test the image-to-text retrieval accuracy on the Blender dataset. We render a new Blender dataset using three different objects, including boots, toys, and trucks. Note that our model and baselines are trained on Relational CLEVR. They had never seen the Blender scenes during training.

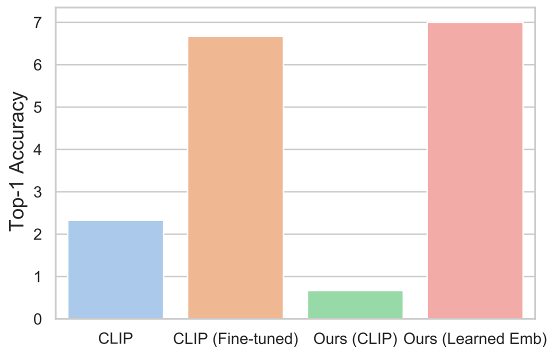


Figure 2-16: **Zero-shot generalization on Blender scenes.** Our approach with learned embedding outperforms other methods on image-to-text retrieval.

We generate a Blender test set that contains 300 pairs of images and relational scene descriptions. For each image, we do text retrieval on the 300 relational scene descriptions. The top 1 accuracy is shown in Figure 2-16. We compare our approaches with two baselines, *i.e.*, CLIP and CLIP fine-tuned on the Relational CLEVR dataset. We find the CLIP model and our approach using the CLIP embedding perform badly on the Blender dataset. This is because CLIP is not good at modeling relational scene description, as we have shown in Section 2.5.6. Our approach using the learned embedding outperforms other methods, indicating that our EBMs with a good embedding feature can generalize well even on unseen datasets, such as Blender.

2.6 Conclusion

In this chapter, we demonstrate the potential usage of energy-based models for compositional image generation, editing, and concept inference. We illustrate that EBMs support composition on both concept and object relation levels, unifying different perspectives of compositionality and can recursively combine them together. We further show how this composition can be applied to continual learning and cross product extrapolation.

Our proposed approach represents a significant stride towards enhancing the composability of deep learning models. A truly compositional system stands to offer remarkable societal advantages, potentially ushering in an era of intelligent robots with the ability to seamlessly acquire diverse skills over time. Furthermore, it could facilitate a superhuman synthesis of scientific knowledge, leading to groundbreaking scientific discoveries. We hope that our findings will serve as a wellspring of inspiration for future research endeavors in these domains.

One limitation of the current approach is that the evaluated datasets are simpler compared to the complex descriptions used in the real world. In the next chapter, we will study how to scale these models to more complex datasets.

Chapter 3

Compositional Visual Generation with Diffusion Models

Nan Liu*, Shuang Li*, Yilun Du*, Antonio Torralba, Joshua B. Tenenbaum; ECCV 2022.

In this chapter, we focus on compositional visual generation using diffusion models [145, 147, 51]. Similar to energy-based models, diffusion models are another type of generative model that has been broadly used for visual generation recently. Text-guided diffusion models such as DALL-E 2 [125] are able to generate stunning photorealistic images given natural language descriptions. While such models are highly flexible, they struggle to understand the composition of certain concepts, such as confusing the attributes of different objects or relations between objects. We introduce an alternative structured approach for compositional generation using diffusion models. An image is generated by composing a set of diffusion models, with each of them modeling a certain component of the image. To do this, we build the connection between diffusion models and energy-based models and apply the compositional operators to diffusion models for composing different concepts.

The proposed method can generate scenes at test time that are substantially more complex than those seen in training, composing sentence descriptions, object

*Equal Contribution

relations, human facial attributes, and even generalizing to new combinations that are rarely seen in the real world. We further illustrate how our approach may be used to compose pre-trained text-guided diffusion models and generate photorealistic images containing all the details described in the input descriptions, including the binding of certain object attributes that have been shown difficult for DALL-E 2, which is one of the-start-of-the-art image generation models. These results point to the effectiveness of the proposed method in promoting structured generalization for visual generation.

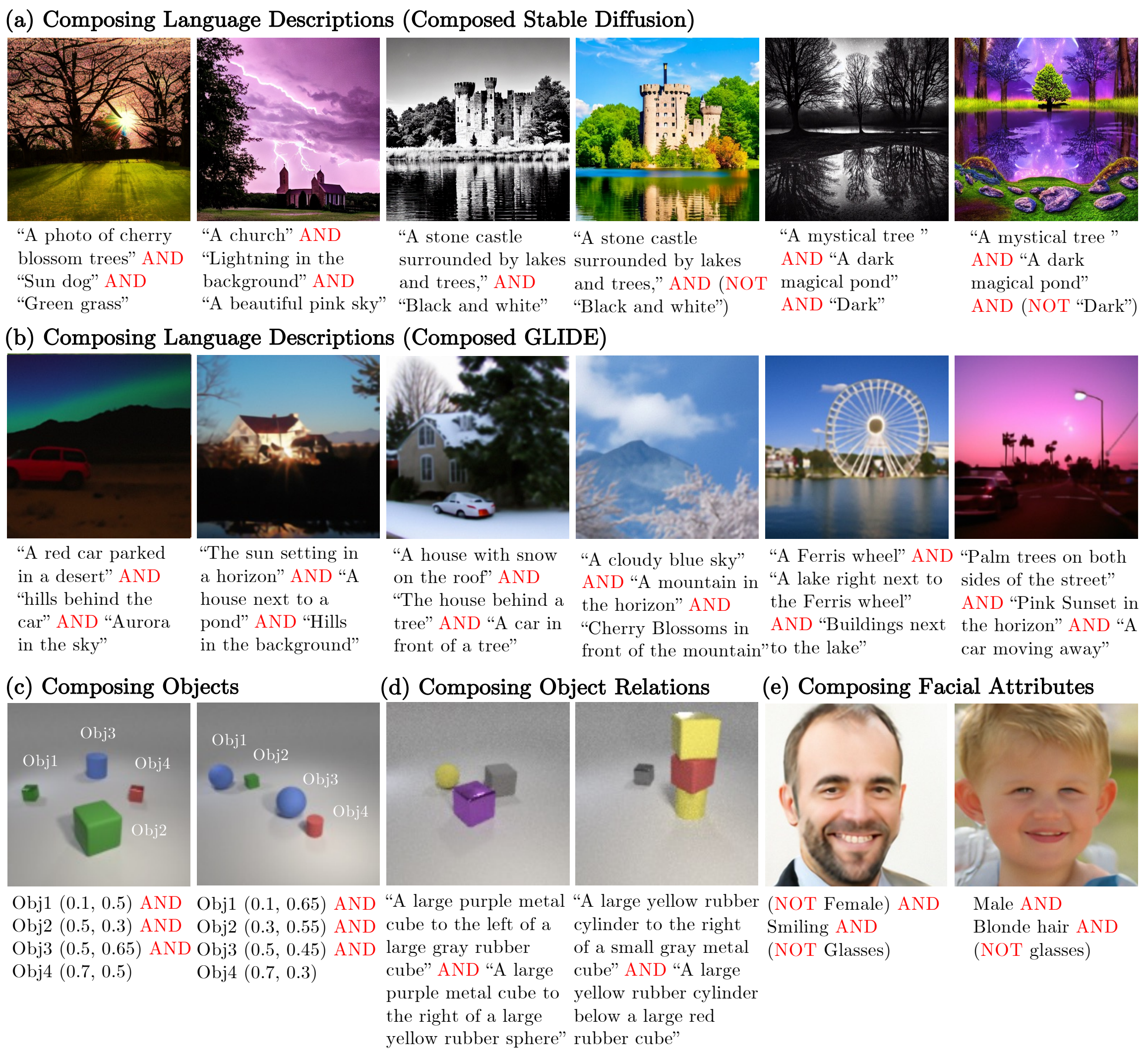
3.1 Introduction

Existing text-conditioned diffusion models such as DALL-E 2 [125] have recently made remarkable strides towards compositional generation, and are capable of generating photorealistic images given textual descriptions. However, such systems are not fully compositional and generate incorrect images when given more complex descriptions [95, 150]. An underlying difficulty may be that such models encode text descriptions as fixed-size latent vectors. However, as textual descriptions become more complex, more information needs to be squeezed into the fixed-size vector. Thus it is impossible to encode arbitrarily complex textual descriptions.

We propose to factorize the compositional generation problem, using different diffusion models to capture different subsets of a compositional specification. These diffusion models are then explicitly composed together to generate an image. By explicitly factorizing the compositional generative modeling problem, our method can generalize to significantly more complex combinations that are unseen during training.

Such an explicit form of compositionality has been explored before under the context of energy-based models [25, 26, 87]. However, directly training EBMs is unstable and hard to scale. We show that diffusion models can be interpreted as implicitly parameterized EBMs, which can be further composed for image generation, significantly improving training stability and image quality.

Our proposed method enables zero-shot compositional generation across different domains as shown in Fig. 3-1. First, we illustrate how our approach may be applied



(a) Composing Language Descriptions (Composed Stable Diffusion)

"A photo of cherry blossom trees" AND "Sun dog" AND "Green grass"

"A church" AND "Lightning in the background" AND "A beautiful pink sky"

"A stone castle surrounded by lakes and trees," AND "Black and white"

"A stone castle surrounded by lakes and trees," AND (NOT "Black and white")

"A mystical tree " AND "A dark magical pond" AND "Dark"

"A mystical tree " AND "A dark magical pond" AND (NOT "Dark")

(b) Composing Language Descriptions (Composed GLIDE)

"A red car parked in a desert" AND "hills behind the car" AND "Aurora in the sky"

"The sun setting in a horizon" AND "A house next to a pond" AND "Hills in the background"

"A house with snow on the roof" AND "The house behind a tree" AND "A car in front of a tree"

"A cloudy blue sky" AND "A mountain in the horizon" AND "Cherry Blossoms in front of the mountain"

"A Ferris wheel" AND "A lake right next to the Ferris wheel" AND "Buildings next to the lake"

"Palm trees on both sides of the street" AND "Pink Sunset in the horizon" AND "A car moving away"

(c) Composing Objects

Obj1 (0.1, 0.5) AND Obj2 (0.5, 0.3) AND Obj3 (0.5, 0.65) AND Obj4 (0.7, 0.5)

Obj1 (0.1, 0.65) AND Obj2 (0.3, 0.55) AND Obj3 (0.5, 0.45) AND Obj4 (0.7, 0.3)

(d) Composing Object Relations

"A large purple metal cube to the left of a large gray rubber cube" AND "A large purple metal cube to the right of a large yellow rubber sphere"

"A large yellow rubber cylinder to the right of a small gray metal cube" AND "A large yellow rubber cylinder below a large red rubber cube"

(e) Composing Facial Attributes

(NOT Female) AND Smiling AND (NOT Glasses)

Male AND Blonde hair AND (NOT glasses)

Figure 3-1: Our method allows compositional visual generation across a variety of domains, such as language descriptions, objects, object relations, and human attributes.

to large pre-trained diffusion models, such as Stable Diffusion [131], GLIDE [103], and Point-E [104] to compose multiple text descriptions to generate 2D images or 3D assets. Next, we illustrate how our approach can be applied to compose objects, enabling zero-shot generalization to a larger number of objects. Finally, we illustrate how our framework can compose different facial attributes to generate human faces.

We introduce an approach for compositional visual generation using diffusion models. In summary, our contributions are:

- First, we show that diffusion models can be composed by interpreting them as energy-based models, and drawing on this connection, we demonstrate how to compose diffusion models together.

- Second, we propose two compositional operators, *Conjunction* and *Negation*, on top of diffusion models that allow us to compose concepts in different domains during inference without any additional training. We show that the proposed method enables effective zero-shot combinatorial generalization, *i.e.*, generating images with more complicated compositions of concepts.

- Finally, we evaluate our method on composing language descriptions, objects, and human facial attributes. Our method can generate high-quality images containing all the concepts and outperforms baselines by a large margin. For example, the accuracy of our method is 24.02% higher than the best baseline for composing three objects in specified positions on the CLEVR dataset.

3.2 Related Work

Controllable image generation. Our work is related to existing work on controllable image generation. One type of approach towards controllable image generation specifies the underlying content of an image utilizing text through GANs [167, 175, 9], VQ-VAEs [126], or diffusion models [103]. An alternative type of approach towards controllable image generation manipulates the underlying attributes in an image [142, 165, 177]. In contrast, we are interested in *compositionally controlling* the underlying content of an image at test time, generating images that exhibit compositions of different components. Recent work has utilized EBMs to compose different factors for image generation [25, 106, 26, 87]. We illustrate how to implement such composition on diffusion models, achieving better performance.

Diffusion models. Diffusion models have emerged as a promising class of generative models that formulate the data-generating process as an iterative denoising procedure [145, 51]. The denoising procedure can be seen as parameterizing the gradients of the data distribution [148], which is similar to EBMs [79, 30, 107, 38, 36]. Diffusion models have recently shown great promise in image generation tasks [23], enabling effective image editing [96, 70], text conditioning [103, 131, 41], and image inpainting [133]. The iterative, gradient-based sampling of diffusion models enables us to compose multiple factors during inference. While diffusion models have been developed for image generation [146], they have further proven successful in the generation of waveforms [15], 3D shapes [176], decision making [61], and text [8], suggesting that our proposed composition operators may further be applied to such domains.

3.3 Background

3.3.1 Denoising diffusion models

Denoising Diffusion Probabilistic Models (DDPMs) [51] are a class of generative models where generation is modeled as a denoising process. Starting from a sampled noise, the diffusion model performs T denoising steps until a sharp image is formed. In particular, the denoising process produces a series of intermediate images with decreasing levels of noise, denoted as $\boldsymbol{x}_T, \boldsymbol{x}_{T-1}, ..., \boldsymbol{x}_0$, where $\boldsymbol{x}_T$ is sampled from a Gaussian prior and $\boldsymbol{x}_0$ is the final output image.

DDPMs construct a forward diffusion process by gradually adding Gaussian noise to the ground truth image. A diffusion model then learns to revert this noise corruption process. Both the *forward processes* $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ and the *reverse process* $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ are modeled as the products of Markov transition probabilities:

$$q(\boldsymbol{x}_{0:T}) = q(\boldsymbol{x}_0)\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}), \qquad p_\theta(\boldsymbol{x}_{T:0}) = p(\boldsymbol{x}_T)\prod_{t=T}^{1} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \tag{3.1}$$

where $q(\boldsymbol{x}_0)$ is the real data distribution and $p(\boldsymbol{x}_T)$ is a standard Gaussian prior.

A *generative process* $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is trained to generate realistic images by approximating the reverse process through variational inference. Each step of the *generative process* is a Gaussian distribution $\mathcal{N}$ with a learned mean $\mu_\theta(\boldsymbol{x}_t, t)$ and covariance matrix $\sigma_t^2\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix.

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}\big(\mu_\theta(\boldsymbol{x}_t, t), \sigma_t^2\boldsymbol{I}\big) = \mathcal{N}\big(\boldsymbol{x}_t - \epsilon_\theta(\boldsymbol{x}_t, t), \sigma_t^2\boldsymbol{I}\big). \tag{3.2}$$

The mean $\mu_\theta(\boldsymbol{x}_t, t)$ is represented by a perturbation $\epsilon_\theta(\boldsymbol{x}_t, t)$ to a noisy image $\boldsymbol{x}_t$. The goal is to remove the noise gradually by predicting a less noisy image at timestep $\boldsymbol{x}_{t-1}$ given a noisy image $\boldsymbol{x}_t$. To generate real images, we sample $\boldsymbol{x}_{t-1}$ from $t = T$ to $t = 1$ using the parameterized marginal distribution $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, with an individual step corresponding to:

$$\boldsymbol{x}_{t-1} = \boldsymbol{x}_t - \epsilon_\theta(\boldsymbol{x}_t, t) + \mathcal{N}(0, \sigma_t^2\boldsymbol{I}). \tag{3.3}$$

The generated images become more realistic over multiple iterations.

The sampling procedure used by diffusion models in Equation (3.3) is functionally similar to the sampling procedure used in energy-based models (see Section 2.3.1):

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \frac{\lambda}{2}\nabla_{\boldsymbol{x}} E_\theta(\boldsymbol{x}_{t-1}) + \mathcal{N}(0, \sigma_t^2\boldsymbol{I}). \tag{3.4}$$

In both settings, images are iteratively refined starting from Gaussian noise, with a small amount of additional noise added at each iterative step.

3.4 Method

In this section, we first introduce how we interpret diffusion models as energy-based models in Section 3.4.1 and then present how to compose diffusion models for visual generation in Section 3.4.2.

3.4.1 Diffusion models as energy-based models

The sampling procedure of diffusion models in Equation (3.3) and EBMs in Equation (3.4) are functionally similar. At a timestep t, in diffusion models, images are updated using a learned denoising network $\epsilon_\theta(\boldsymbol{x}_t, t)$ while in EBMs, images are updated using the gradient of the energy function $\nabla_{\boldsymbol{x}} E_\theta(\boldsymbol{x}_t) \propto \nabla_{\boldsymbol{x}} \log p_\theta(\boldsymbol{x}_t)$. The denoising

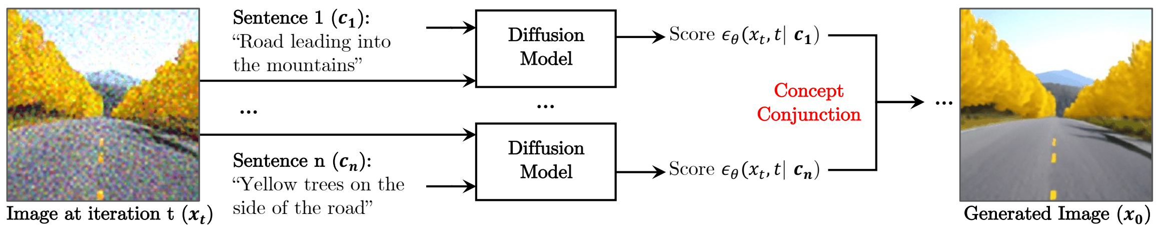

Sentence 1 (c_1):
"Road leading into the mountains"
...
Sentence n (c_n):
"Yellow trees on the side of the road"
Diffusion Model
...
Diffusion Model
Score $\epsilon_\theta(x_t, t| c_1)$
Score $\epsilon_\theta(x_t, t| c_n)$
Concept Conjunction
...
Image at iteration t (x_t)
Generated Image (x_0)

Figure 3-2: **Compositional generation.** Our method can compose multiple concepts during inference and generate images containing all the concepts without further training. We first send an image from iteration t and each of the concepts to the diffusion model to generate a set of scores $\{\epsilon_\theta(\boldsymbol{x}_t, t|\boldsymbol{c}_1), \ldots, \epsilon_\theta(\boldsymbol{x}_t, t|\boldsymbol{c}_n)\}$. We then compose different concepts using the proposed compositional operators, such as conjunction, to denoise the generated image. The final image is obtained after T iterations.

network $\epsilon_\theta(\boldsymbol{x}_t, t)$ is trained to predict the underlying score of the data distribution [159, 146] when the number of diffusion steps increases to infinity. Similarly, an EBM is trained so that $\nabla_{\boldsymbol{x}} E_\theta(\boldsymbol{x}_t)$ corresponds to the score of the data distribution as well. In this sense, $\epsilon_\theta(\boldsymbol{x}_t, t)$ and $\nabla_{\boldsymbol{x}} E_\theta(\boldsymbol{x}_t)$ are functionally the same, and the underlying sampling procedure in Equation (3.3) and Equation (3.4) are equivalent. We may view a trained diffusion model $\epsilon_\theta(\boldsymbol{x}_t, t)$ as an implicitly parameterized EBM. Such parameterization enables us to apply previous techniques for composing EBMs to diffusion models.

Composing EBMs. Previous EBMs [49, 25, 87] have shown good performance on compositional visual generation. Given n independent EBMs, $E_\theta^1(\boldsymbol{x}), \cdots, E_\theta^n(\boldsymbol{x})$, the functional form of EBMs in Equation (2.1) enables us to compose multiple separate EBMs together to obtain a new EBM. The composed distribution can be represented as:

$$p_{\text{compose}}(\boldsymbol{x}) \propto \prod_{i=1}^{n} p_\theta^i(\boldsymbol{x}) \propto e^{-\sum_{i=1}^{n} E_\theta^i(\boldsymbol{x})}, \tag{3.5}$$

where $p_\theta^i \propto e^{-E_\theta^i(\boldsymbol{x})}$ is the probability density of image $\boldsymbol{x}$ (Equation (2.1)). Langevin dynamics is then used to iteratively refine the generated image $\boldsymbol{x}$ [25, 87]:

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \frac{\lambda}{2} \nabla_{\boldsymbol{x}} \left(\sum_{i=1}^{n} E_\theta^i(\boldsymbol{x}_{t-1}) \right) + \mathcal{N}(0, \sigma_t^2 \boldsymbol{I}). \tag{3.6}$$

Composing diffusion models. By leveraging the interpretation that diffusion models are functionally similar to EBMs, we may compose diffusion models in a similar way. The *generative process* and the score function of a diffusion model can be represented as $p_\theta^i(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ and $\epsilon_\theta^i(\boldsymbol{x}_t, t)$, respectively. If we treat the score functions in diffusion models as the learned gradient of energy functions in EBMs, the score function of the composed diffusion model can be written as $\sum_{i=1}^n \epsilon_\theta^i(\boldsymbol{x}_t, t)$. Thus the *generative process* of composing multiple diffusion models becomes:

$$p_{\text{compose}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}\left(\boldsymbol{x}_t - \sum_{i=1}^{n} \epsilon_\theta^i(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}\right). \tag{3.7}$$

A complication of parameterizing a gradient field of EBM $\nabla_{\boldsymbol{x}} E_\theta(\boldsymbol{x}_t)$ with a learned score function $\epsilon_\theta(\boldsymbol{x}_t, t)$ is that the gradient field may not be conservative, and thus does not correspond to a valid probability density. However, as discussed in [136], explicitly parameterizing the learned function $\epsilon_\theta(\boldsymbol{x}_t, t)$ as the gradient of EBM achieves similar performance as the non-conservative parameterization of diffusion models, suggesting this is not problematic.

3.4.2 Compositional generation through diffusion models

Next, we discuss how we compose diffusion models for image generation. We aim to generate images conditioned on a set of concepts $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n\}$. To do this, we represent each concept $\boldsymbol{c}_i$ using a diffusion model, which can be composed to generate images. Inspired by EBMs [25, 87], we define two compositional operators, **conjunction (AND)** and **negation (NOT)**, to compose diffusion models. We learn a set of diffusion models representing the conditional probability distribution $p(\boldsymbol{x}|\boldsymbol{c}_i)$ given concept $\boldsymbol{c}_i$ and an unconditional probability distribution $p(\boldsymbol{x})$.

Concept conjunction (AND). We aim to generate images containing certain attributes. Following [25], the conditional probability can be factorized as:

$$p(\boldsymbol{x}|\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n) \propto p(\boldsymbol{x}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_n) = p(\boldsymbol{x}) \prod_{i=1}^{n} p(\boldsymbol{c}_i|\boldsymbol{x}). \tag{3.8}$$

Here we assume the concepts are conditionally independent given $\boldsymbol{x}$. We can represent $p(\boldsymbol{c}_i|\boldsymbol{x})$ using the combination of a conditional distribution $p(\boldsymbol{x}|\boldsymbol{c}_i)$ and an unconditional distribution $p(\boldsymbol{x})$, with both of them are parameterized as diffusion models $p(\boldsymbol{c}_i|\boldsymbol{x}) \propto \frac{p(\boldsymbol{x}|\boldsymbol{c}_i)}{p(\boldsymbol{x})}$. The expression of $p(\boldsymbol{c}_i|\boldsymbol{x})$ corresponds to the implicit classifier that represents the likelihood of $\boldsymbol{x}$ exhibiting concept $\boldsymbol{c}_i$. Substituting $p(\boldsymbol{c}_i|\boldsymbol{x})$ in Equation 3.8, we can rewrite Equation 3.8 as:

$$p(\boldsymbol{x}|\boldsymbol{c}_1, \dots, \boldsymbol{c}_n) \propto p(\boldsymbol{x}) \prod_{i=1}^{n} \frac{p(\boldsymbol{x}|\boldsymbol{c}_i)}{p(\boldsymbol{x})}. \tag{3.9}$$

We sample from this resultant distribution using Equation (3.7) with the composed score function $\hat{\epsilon}(\boldsymbol{x}_t, t)$:

$$\hat{\epsilon}(\boldsymbol{x}_t, t) = \epsilon_\theta(\boldsymbol{x}_t, t) + \sum_{i=1}^{n} w_i \big(\epsilon_\theta(\boldsymbol{x}_t, t|\boldsymbol{c}_i) - \epsilon_\theta(\boldsymbol{x}_t, t)\big), \tag{3.10}$$

where w_i is a hyperparameter corresponding to the temperature scaling on concept $\boldsymbol{c}_i$. We can generate images with the composed concepts using the following *generative process*:

$$p_{compose}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}\big(\boldsymbol{x}_t - \hat{\epsilon}(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}\big). \tag{3.11}$$

In the setting in which image generation is conditioned on a single concept, the above sampling procedure reduces to the classifier-free guidance [52].

Concept negation (NOT). In concept negation, we aim to generate realistic images with the absence of a certain factor $\tilde{\boldsymbol{c}}_j$. However, the negation of a concept can be ill-defined. For example, the negation of "dark" can be "bright" or random noises. Thus we generate images conditioned on other concepts as well to make the generated images look real. Following [25], concept negation can be represented as the composed probability distribution $p(\boldsymbol{x}|\text{NOT } \tilde{\boldsymbol{c}}_j, \boldsymbol{c}_i)$. Similarly, we refactorize the joint probability distribution as follows:

$$p(\boldsymbol{x}|\text{NOT } \tilde{\boldsymbol{c}}_j, \boldsymbol{c}_i) \propto p(\boldsymbol{x}, \text{NOT } \tilde{\boldsymbol{c}}_j, \boldsymbol{c}_i) \propto p(\boldsymbol{x}) \frac{p(\boldsymbol{c}_i|\boldsymbol{x})}{p(\tilde{\boldsymbol{c}}_j|\boldsymbol{x})}. \tag{3.12}$$

Algorithm 1: Code for Composing Diffusion Models

```
1: Require Diffusion model $\epsilon_\theta(\boldsymbol{x}_t, t|\boldsymbol{c})$, scales $w_i$ and $w$, covariance matrix $\sigma_t^2\boldsymbol{I}$
2: // Code for conjunction
3: Initialize sample $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
4: for $t = T, \ldots, 1$ do
5:     $\epsilon_i \leftarrow \epsilon_\theta(\boldsymbol{x}_t, t|\boldsymbol{c}_i)$    // compute conditional scores for each concept $\boldsymbol{c}_i$
6:     $\epsilon \leftarrow \epsilon_\theta(\boldsymbol{x}_t, t)$    // compute unconditional score
7:     $\boldsymbol{x}_{t-1} \sim \mathcal{N}\Big(\boldsymbol{x}_t - \big(\epsilon + \sum_{i=1}^n w_i(\epsilon_i - \epsilon)\big), \sigma_t^2\boldsymbol{I}\Big)$    // sampling
8: end for
9:
10: // Code for negation
11: Initialize sample $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
12: for $t = T, \ldots, 1$ do
13:     $\tilde{\epsilon}_j \leftarrow \epsilon_\theta(\boldsymbol{x}_t, t|\tilde{\boldsymbol{c}}_j)$    // compute conditional score for the negated concept $\tilde{\boldsymbol{c}}_j$
14:     $\epsilon_i \leftarrow \epsilon_\theta(\boldsymbol{x}_t, t|\boldsymbol{c}_i)$    // compute conditional score for concept $\boldsymbol{c}_i$
15:     $\epsilon \leftarrow \epsilon_\theta(\boldsymbol{x}_t, t)$    // compute unconditional score
16:     $\boldsymbol{x}_{t-1} \sim \mathcal{N}\Big(\boldsymbol{x}_t - \big(\epsilon + w(\epsilon_i - \tilde{\epsilon}_j)\big), \sigma_t^2\boldsymbol{I}\Big)$    // sampling
17: end for
```

Using the factorization $p(\boldsymbol{c}_i|\boldsymbol{x}) \propto \frac{p(\boldsymbol{x}|\boldsymbol{c}_i)}{p(\boldsymbol{x})}$, we can rewrite Equation (3.12) as:

$$p(\boldsymbol{x}|\text{NOT } \tilde{\boldsymbol{c}}_j, \boldsymbol{c}_i) \propto p(\boldsymbol{x})\frac{p(\boldsymbol{x}|\boldsymbol{c}_i)}{p(\boldsymbol{x}|\tilde{\boldsymbol{c}}_j)} \tag{3.13}$$

We may construct the composed score function $\hat{\epsilon}(\boldsymbol{x}, t)$ as:

$$\hat{\epsilon}(\boldsymbol{x}_t, t) = \epsilon_\theta(\boldsymbol{x}_t, t) + w\big(\epsilon_\theta(\boldsymbol{x}_t, t|\boldsymbol{c}_i) - \epsilon_\theta(\boldsymbol{x}_t, t|\tilde{\boldsymbol{c}}_j)\big), \tag{3.14}$$

where w is the hyperparameter that controls the strength of the negation. We can generate samples using this composed score function and Equation 3.11.

Algorithm 1 provides the pseudo-code for composing diffusion models using concept conjunction and negation. Our method can compose pre-trained diffusion models during inference without any additional training.

3.5 Experiment Setup

3.5.1 Datasets

CLEVR. CLEVR [63] is a synthetic dataset containing objects with different shapes, colors, and sizes. The training set consists of $30,000$ images at 128×128 resolution. Each image contains $1 \sim 5$ objects and a 2D coordinate (x, y) label indicating that the image contains an object at (x, y). In our experiments, the 2D coordinate label is the coordinate of one object in the image.

Relational CLEVR. Relational CLEVR [87] contains relational descriptions between objects in the image, such as "a red cube to the left of a blue cylinder". The training dataset contains $50,000$ images at 128×128 resolution. Each training image contains $1 \sim 5$ objects and one label describing a relation between two objects. If there is only one object in the image, the second object and their relation in the relational description are both nulls.

FFHQ. FFHQ [67] is a real-world human face dataset. The original FFHQ dataset consists of 70,000 human face images without labels. [19] annotates three binary attributes, including *smile*, *gender*, and *glasses*, for the images using pre-trained classifiers. There are 51,067 images labeled by the classifiers.

3.5.2 Evaluation metrics

Binary classification accuracy. During testing, we evaluate the performance of the proposed method and baselines on three different settings. The first test setting, **1 Component**, generates images conditioned on a single concept (matching the training distribution). The second and third test settings, **2 Components** and **3 Components**, generate images by composing two and three concepts, respectively, using the *conjunction* and *negation* operators. They are used to evaluate the model's generalization ability to new combinations.

For each task, we use the training data (real images) to train a binary classifier that takes an image and a concept, *e.g.*, 'smiling', as input and predicts whether the image contains or represents the concept. We then apply this classifier to a generated image, checking whether it faithfully captures each of the concepts. In each test setting, each method generates $5,000$ images for evaluation. The accuracy of the method is the percentage of generated images capturing all the concepts (See Appendix B.2 for more details about the classifiers).

Fréchet Inception Distance (FID) is a commonly used metric for evaluating the quality of generated images. It uses a pre-trained inception model [151] to extract features for the generated images and real images, and measures their feature similarity. Specifically, we use Clean-FID [111] to evaluate the generated images. FID is usually computed on $50,000$ generated images, but we use $5,000$ images in our experiments.

3.6 Experiments

We compare the proposed method and baselines (Section 3.6.1) on compositional generation in different domains. We show results of composing natural language descriptions (Section 3.6.2), objects (Section 3.6.3), and human facial attributes (Section 3.6.4). Results analysis are shown in Section 3.6.5.

3.6.1 Baselines

We compare our method with baselines for compositional visual generation.

StyleGAN2-ADA [66] is the state-of-the-art GAN method for both unconditional and conditional image generation.

StyleGAN2 [68] is one of the state-of-the-art GAN methods for unconditional image generation. To enable compositional image generation, we optimize the latent code z by decreasing the binary classification loss of the generated image and the given label. We use the resultant latent code to generate images.

LACE [106] uses pre-trained classifiers to generate energy scores in the latent space of the pre-trained StyleGAN2 model. LACE uses compositional operators [25] for compositional image synthesis.

GLIDE [103] is a recently released text-conditioned diffusion model for image generation. For composing language descriptions, we use the pre-trained GLIDE released by OpenAI. For the rest of the tasks, we use the GLIDE code and train a model on each task.

Energy-based models (EBM) [25] is the first paper using EBMs for compositional visual generation. We use the three compositional operators for composing different concepts.

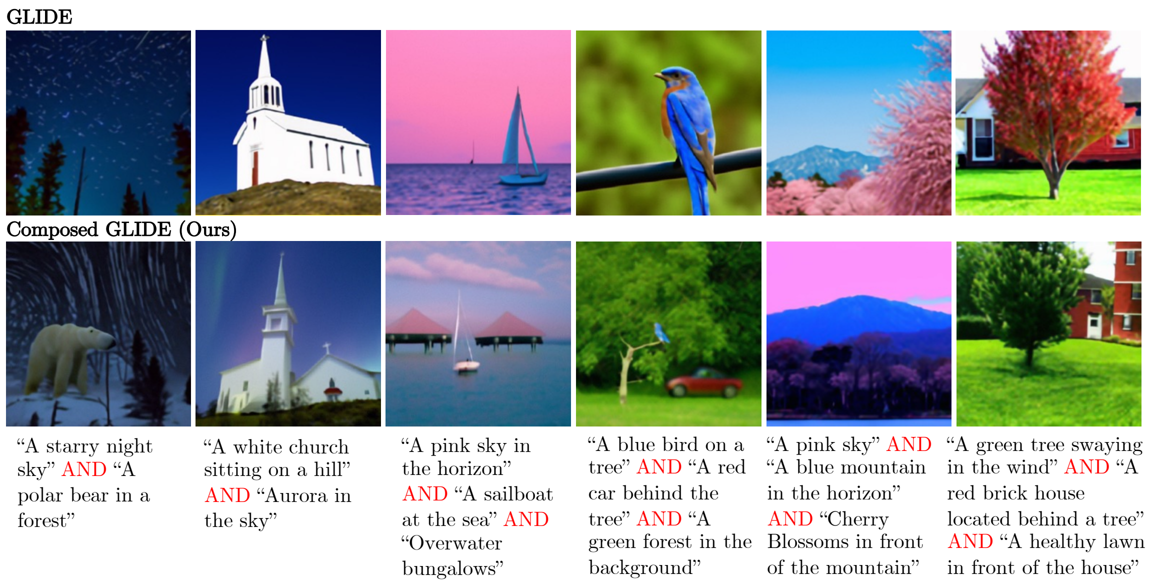

GLIDE
Composed GLIDE (Ours)
"A starry night sky" AND "A polar bear in a forest"
"A white church sitting on a hill" AND "Aurora in the sky"
"A pink sky in the horizon" AND "A sailboat at the sea" AND "Overwater bungalows"
"A blue bird on a tree" AND "A red car behind the tree" AND "A green forest in the background"
"A pink sky" AND "A blue mountain in the horizon" AND "Cherry Blossoms in front of the mountain"
"A green tree swaying in the wind" AND "A red brick house located behind a tree" AND "A healthy lawn in front of the house"

Figure 3-3: **Composing language descriptions.** We develop Composed GLIDE (Ours), a version of GLIDE [103] that utilizes our compositional operators to combine textual descriptions, without further training. We compare it with the original GLIDE, which directly encodes the descriptions as a single long sentence. Our approach can capture text details more accurately, such as the "overwater bungalows" in the third example.

3.6.2 Composing language descriptions

Composing language for 2D image generation. Our approach can effectively compose natural language descriptions. We first show the image generation results of the pre-trained diffusion model, GLIDE [103], in Fig. 3-3. We develop Composed GLIDE, a version of GLIDE that utilizes our compositional operators to combine textual descriptions, without further training. We compare this model with the original GLIDE model.

In Fig. 3-3, GLIDE takes a single long sentence as input, for example, "A pink sky in the horizon, a sailboat at the sea, and overwater bungalows". In contrast, Composed GLIDE composes several short sentences using the concept conjunction operator, *e.g.*, "A pink sky in the horizon" AND "A sailboat at the sea" AND "Overwater bungalows". While both GLIDE and Composed GLIDE can generate reasonable images containing objects described in the text prompt, our approach with the compositional operators can more accurately capture text details, such as the presence of "a polar bear" in the first example and the "overwater bungalows" in the third example.

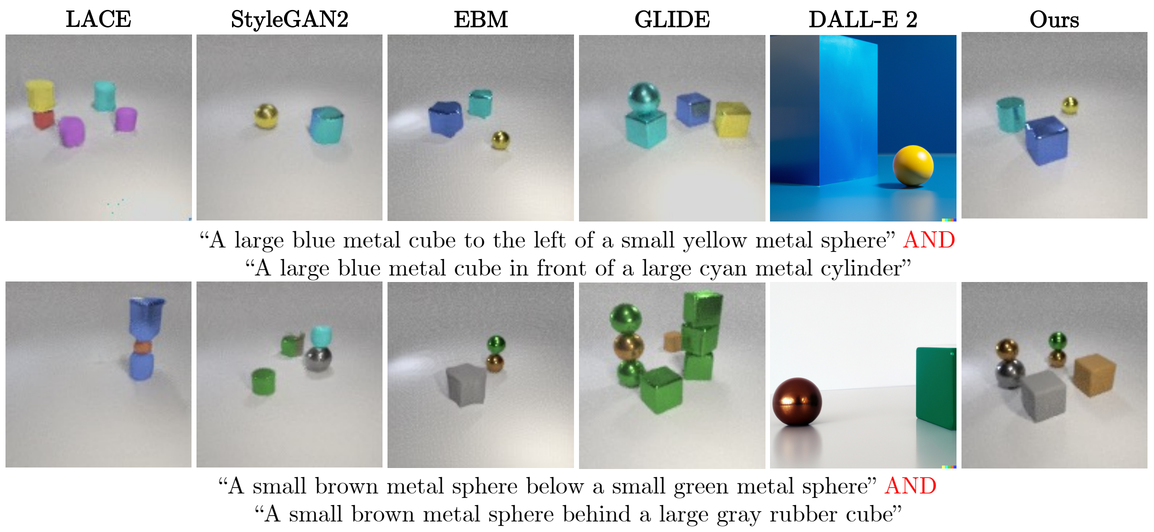

LACE
StyleGAN2
EBM
GLIDE
DALL-E 2
Ours
"A large blue metal cube to the left of a small yellow metal sphere" AND
"A large blue metal cube in front of a large cyan metal cylinder"
"A small brown metal sphere below a small green metal sphere" AND
"A small brown metal sphere behind a large gray rubber cube"

Figure 3-4: **Composing object relational descriptions.** Image generation results on the Relational CLEVR dataset. Our model is trained to generate images conditioned on a single object relation, but during inference, our model can compose multiple object relations, generating better results than baselines.

Composing object relational descriptions. We further compare the proposed approach and baselines on composing object relational descriptions in Figure 3-4. Our model is trained to generate images conditioned on a single object relation, but it can compose multiple object relations during inference without additional training. Both LACE and StyleGAN2 fail to capture object relations in the input sentences, but EBM and diffusion models can correctly compose multiple object relations. Diffusion models generate higher-quality images compared with EBM, *e.g.*, the object boundaries are sharper in our results than EBM. Surprisingly, DALL-E 2 and GLIDE can generate high-quality images, but they fail to understand object relations.

Composing language for 3D asset generation. We demonstrate that the proposed method can compose language descriptions for 3D point cloud generation, which can be further used to generate 3D meshes. We first use Point-E [104], the pre-trained 3D point cloud generation model, to generate the point clouds of an object based on the text description. We then convert the 3D point clouds into 3D meshes using marching cubes [90]. The results are shown in Fig. 3-5.

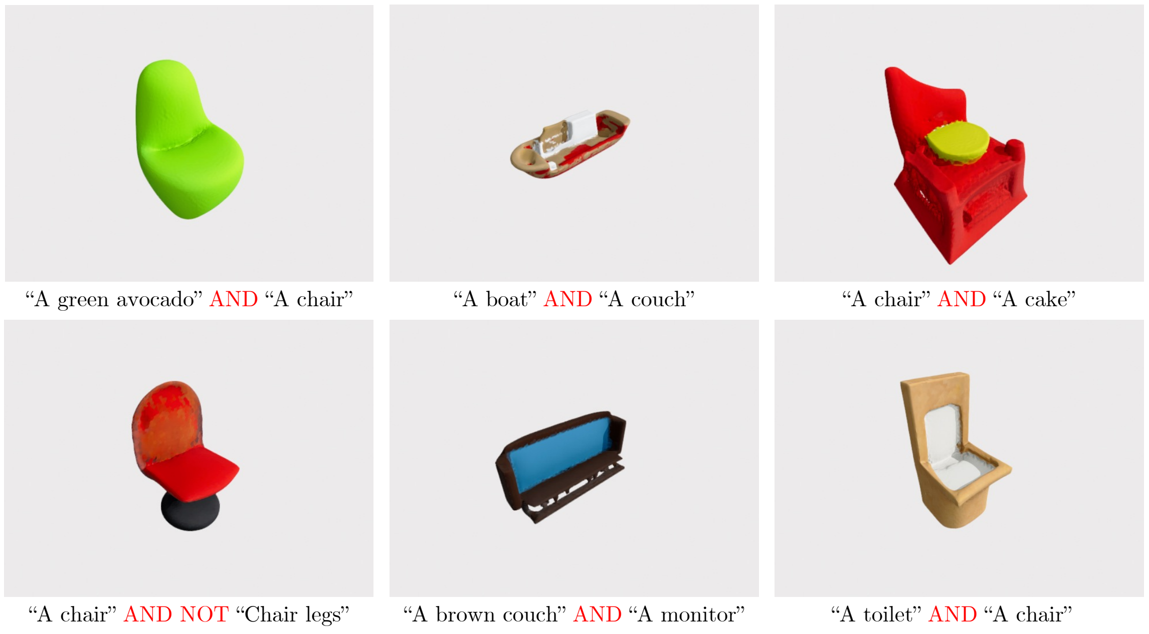

"A green avocado" AND "A chair"
"A boat" AND "A couch"
"A chair" AND "A cake"
"A chair" AND NOT "Chair legs"
"A brown couch" AND "A monitor"
"A toilet" AND "A chair"

Figure 3-5: **Composing language descriptions for 3D asset generation.** We provide qualitative results of composing the pre-trained text-to-3D diffusion model, Point-E [104], to generate 3D objects.

3.6.3 Composing objects

Given a set of 2D object positions, we aim to generate images containing objects at these positions.

Qualitative results. We compare the proposed method with baselines on composing objects in Fig. 3-6. We only show the concept conjunction here because the object positions are not binary values, and thus the negation of object positions is not interpretable. Given a set of object position labels, we compose them to generate images. Our model can generate images of objects at certain locations, while the baseline methods either miss objects or generate incorrect objects.

Quantitative results. As shown in Table 3.1, our method outperforms baselines by a large margin. The binary classification accuracy of our method is 15.88% higher than EBM in the *1 component* test setting and is 24.02% higher than EBM in the more challenging *3 Components* setting. Our method is more effective in zero-shot compositional generalization. In addition, our method can generate images with lower

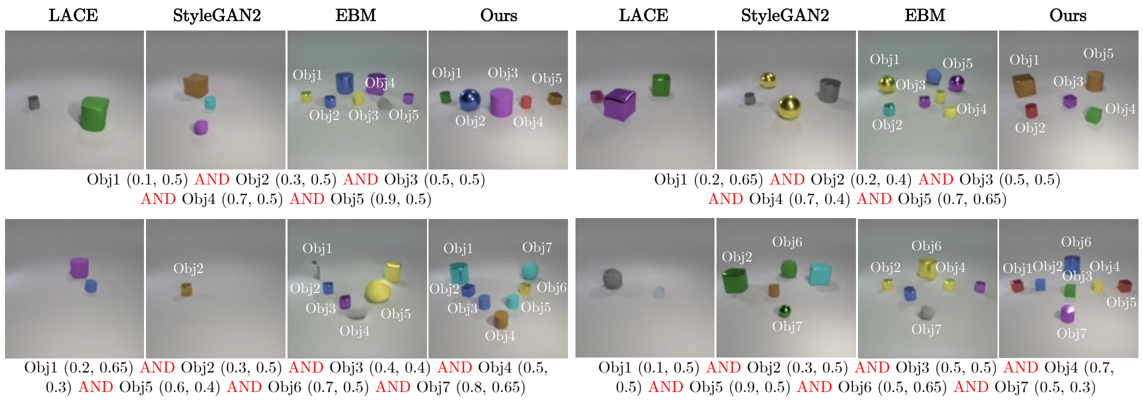

LACE
StyleGAN2
EBM
Ours
LACE
StyleGAN2
EBM
Ours
Obj1 (0.1, 0.5) AND Obj2 (0.3, 0.5) AND Obj3 (0.5, 0.5) AND Obj4 (0.7, 0.5) AND Obj5 (0.9, 0.5)
Obj1 (0.2, 0.65) AND Obj2 (0.2, 0.4) AND Obj3 (0.5, 0.5) AND Obj4 (0.7, 0.4) AND Obj5 (0.7, 0.65)
Obj1 (0.2, 0.65) AND Obj2 (0.3, 0.5) AND Obj3 (0.4, 0.4) AND Obj4 (0.5, 0.3) AND Obj5 (0.6, 0.4) AND Obj6 (0.7, 0.5) AND Obj7 (0.8, 0.65)
Obj1 (0.1, 0.5) AND Obj2 (0.3, 0.5) AND Obj3 (0.5, 0.5) AND Obj4 (0.7, 0.5) AND Obj5 (0.9, 0.5) AND Obj6 (0.5, 0.65) AND Obj7 (0.5, 0.3)

Figure 3-6: **Composing objects.** Our method can compose multiple objects while baseline methods either miss objects or generate objects at wrong positions.

Table 3.1: **Quantitative evaluation of image generation results on CLEVR.** The binary classification accuracy (Acc) and FID scores are reported. Our method outperforms baselines on all three test settings.

Models	1 Component		2 Components		3 Components	
	Acc (%) ↑	FID ↓	Acc (%) ↑	FID ↓	Acc (%) ↑	FID ↓
StyleGAN2-ADA [66]	37.28	57.41	-	-	-	-
StyleGAN2 [68]	1.04	51.37	0.04	23.29	0.00	19.01
LACE [106]	0.70	50.92	0.00	22.83	0.00	19.62
GLIDE [103]	0.86	61.68	0.06	38.26	0.00	37.18
EBM [25]	70.54	78.63	28.22	65.45	7.34	58.33
Ours	**86.42**	**29.29**	**59.20**	**15.94**	**31.36**	**10.51**

FID scores, indicating the generated images are more similar to real images.

3.6.4 Composing human facial attributes

Qualitative results. We compare the proposed method with baselines on composing facial attributes in Fig. 3-7. We find that LACE and StyleGAN2 can generate high-fidelity images, but the generated images do not match the given labels. For example, StyleGAN2 generates humans without wearing glasses when the input label contains "Glasses", while LACE generates males sometimes when the input is "Not Male". Our method can generate high-fidelity images, containing all the attributes in the input label.

Quantitative results. The results of our method and baselines on three test settings are shown in Table 3.2. Our method is comparable with the best baseline on each

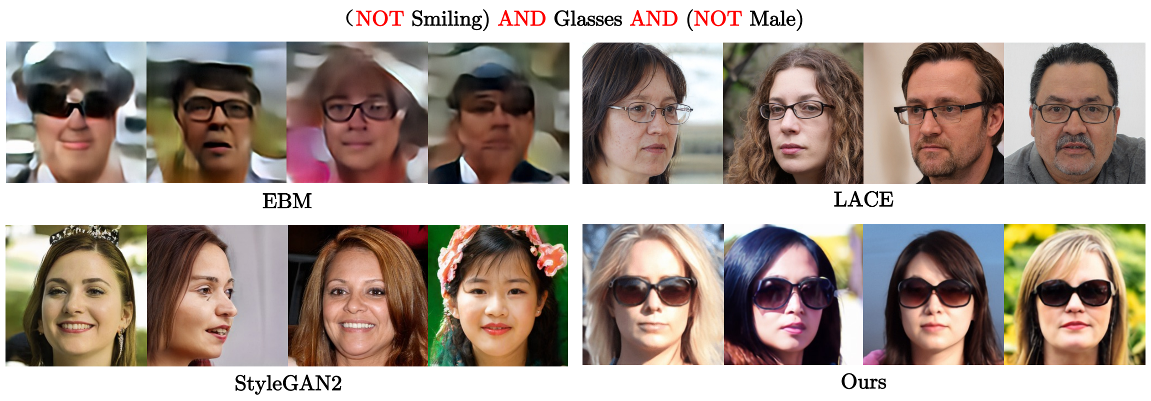


Figure 3-7: **Composing facial attributes.** Image generation results on the FFHQ dataset. Our model is trained to generate images conditioned on a single human facial attribute, but during inference, our model can recursively compose multiple facial attributes using the proposed compositional operators.

Table 3.2: **Image generation results on FFHQ.** The binary classification accuracy (Acc) and FID are reported. Our method achieves comparable results with the best baseline on three test settings.

Models	**1 Component**		**2 Components**		**3 Components**	
	Acc (%) ↑	FID ↓	Acc (%) ↑	FID ↓	Acc (%) ↑	FID ↓
StyleGAN2-ADA [66]	91.06	**10.75**	-	-	-	-
StyleGAN2 [68]	58.90	18.04	30.68	18.06	16.96	18.06
LACE [106]	97.60	28.21	**95.66**	36.23	**80.88**	34.64
GLIDE [103]	98.66	20.30	48.68	22.69	27.24	21.98
EBM [25]	98.74	89.95	93.10	99.64	30.01	335.70
Ours	**99.26**	18.72	92.68	**17.22**	68.86	**16.95**

evaluation metric.

3.6.5 Results analysis

We show our composed results on image generation and the results generated conditioned on each individual sentence description in Fig. 3-8. We provide four successfully composed examples, where the generated images contain all the concepts mentioned in the input sentences.

Failure cases. We observed three main failure cases of the proposed method. The first one is that the pre-trained diffusion models do not understand certain concepts, such as "person" in (b). This is because the pre-trained diffusion model, GLIDE [103],

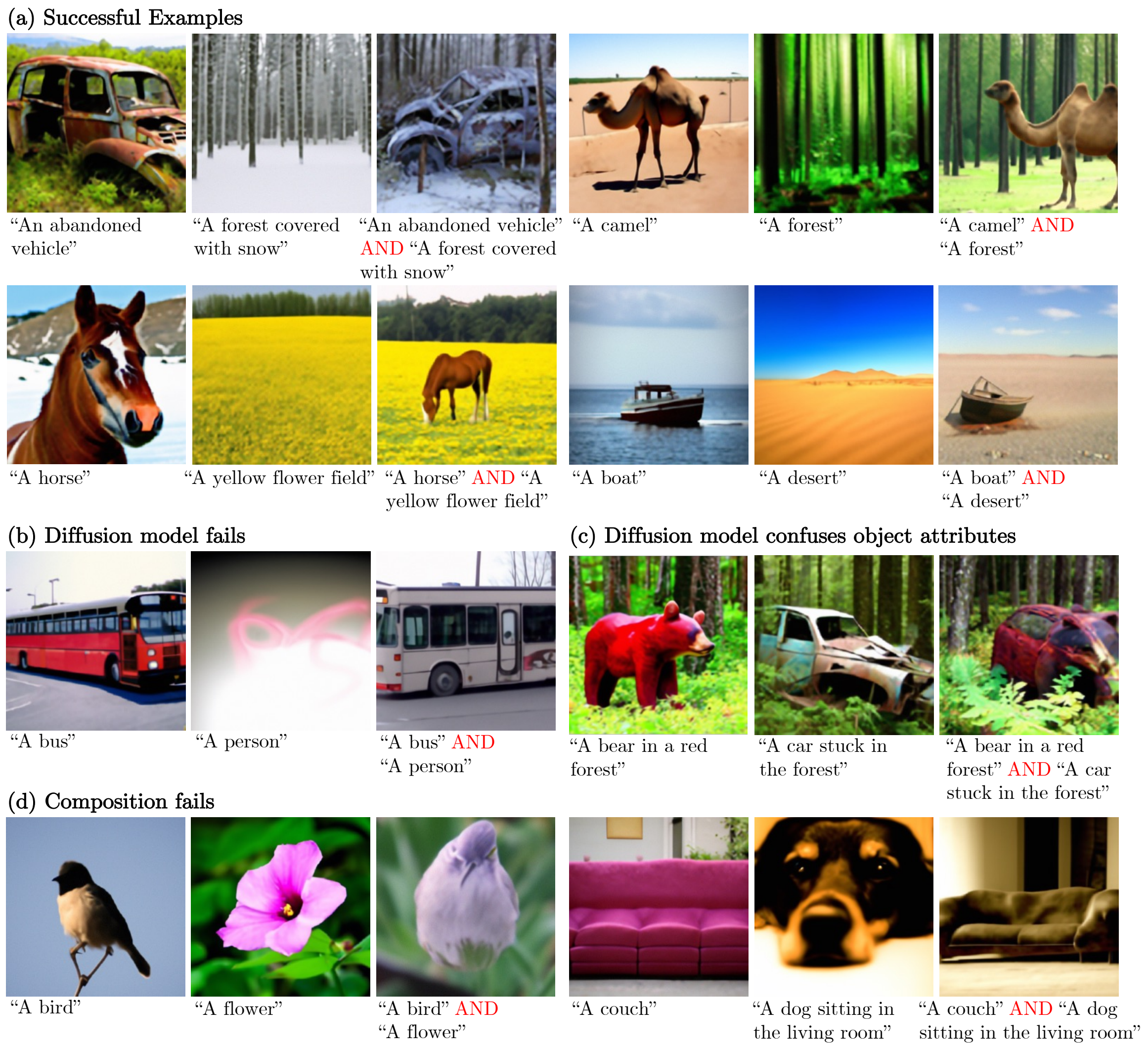

(a) Successful Examples
"An abandoned vehicle"
"A forest covered with snow"
"An abandoned vehicle" AND "A forest covered with snow"
"A camel"
"A forest"
"A camel" AND "A forest"
"A horse"
"A yellow flower field"
"A horse" AND "A yellow flower field"
"A boat"
"A desert"
"A boat" AND "A desert"
(b) Diffusion model fails
"A bus"
"A person"
"A bus" AND "A person"
(c) Diffusion model confuses object attributes
"A bear in a red forest"
"A car stuck in the forest"
"A bear in a red forest" AND "A car stuck in the forest"
(d) Composition fails
"A bird"
"A flower"
"A bird" AND "A flower"
"A couch"
"A dog sitting in the living room"
"A couch" AND "A dog sitting in the living room"

Figure 3-8: **Results analysis.** Successful examples (a) and failure examples (b-d) generated by the proposed method. There are three main types of failures: (b) The pre-trained diffusion model does not understand certain concepts, such as "person". (c) The pre-trained diffusion model confuses objects' attributes. (d) The composition fails. This usually happens when the objects are in the center of the images.

is trained to avoid generating human images. The second type of failure is because the diffusion models confuse the objects' attributes. In (c), the generated image contains "a red bear" while the input is "a bear in a red forest". The third type of failure is because the composition does not work, *e.g.*, the "bird-shape and flower-color object" and the "dog-fur and sofa-shape object" in (d). Such failures usually happen when the objects are in the center of the images.

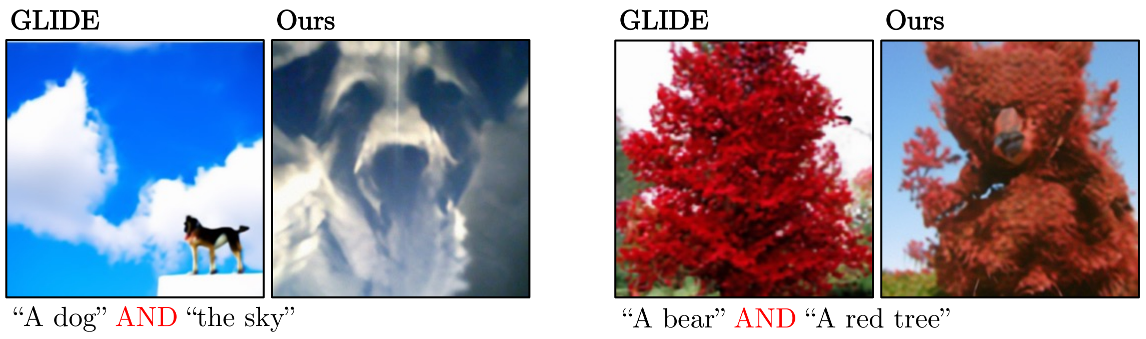


Figure 3-9: **Interesting cases.** Our method (composing multiple sentences) generates different styles of images compared to GLIDE (directly encoding the descriptions as a single long sentence).

Interesting cases. As shown in Fig. 3-9, we find that our method, which combines multiple textual descriptions, can generate different styles of images compared to GLIDE, which directly encodes the descriptions as a single long sentence. Taking "a dog" and "the sky" as inputs, our method generates a dog-shaped cloud, whereas GLIDE generates a dog under the sky using the prompt "a dog and the sky".

3.7 Conclusion

In this chapter, we compose diffusion models for image generation. By building the connection between energy-based models and diffusion models, we may explicitly compose diffusion models and generate images with significantly more complex combinations that are never seen during training. The proposed compositional operators allow us to compose diffusion models during inference time without any additional training. The proposed composable diffusion models can generate images conditioned on sentence descriptions, objects, and human facial attributes, and can generalize to new combinations that are rarely seen in the real world. These results demonstrate the effectiveness of the proposed method for compositional visual generation.

In Chapter 2 and Chapter 3, we define the compositional operators for composing concepts and goals. These concepts and goals are usually from the same domain. However, it is important to note that our reality exhibits a rich tapestry of modalities, encompassing various modes of existence and perception. In the next chapter, we will introduce how to compose pre-trained models from different modalities, such as vision models and language models, and build a more powerful multi-modal framework that can solve a wide range of tasks.

Part II

Composing Models

Chapter 4

Composing Ensembles of Pre-trained Models via Iterative Consensus

Shuang Li, Yilun Du, Joshua B. Tenenbaum, Antonio Torralba, Igor Mordatch; ICLR 2023.

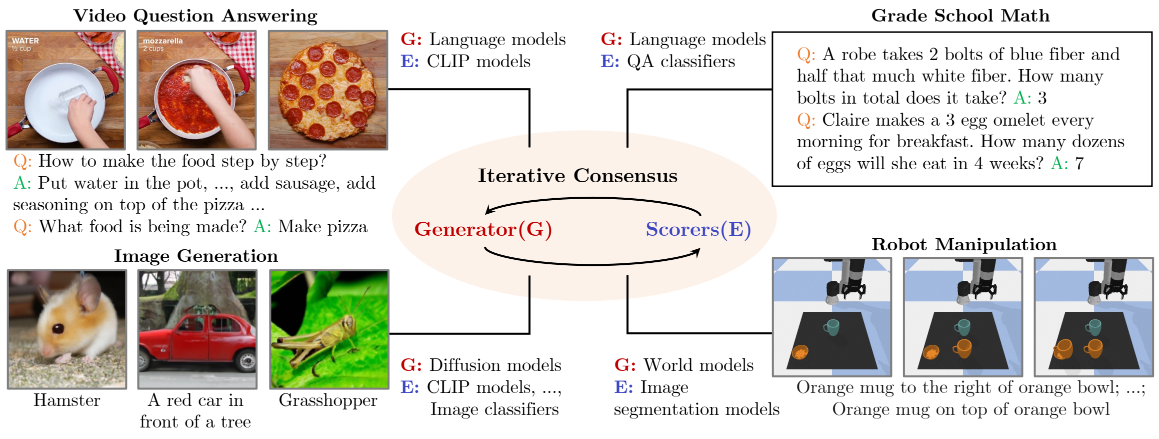
In Part I, we developed compositional operators to compose concepts and goals, predominantly from a single domain like natural language descriptions. Building on that foundation, this section delves into composing models from diverse domains. Our goal is to establish a robust multimodal framework capable of addressing different tasks.

Large pre-trained models exhibit distinct and complementary capabilities dependent on the data they are trained on. Language models such as GPT-3 [13] are capable of textual reasoning but cannot understand visual information, while vision models such as DALL-E 2 [125] can generate photorealistic photos but fail to understand complex language descriptions. In this part, we introduce a unified framework for composing ensembles of different pre-trained models – combining the strengths of each individual model to solve various multimodal problems in a zero-shot manner. We use pre-trained models as "generators" or "scorers" and compose them via closed-loop iterative consensus optimization. The generator constructs proposals and the scorers iteratively provide feedback to refine the generated result. Such closed-loop communication enables models to correct errors caused by other models, significantly boosting performance on downstream tasks. We demonstrate that consensus achieved by an ensemble of scorers outperforms the feedback of a single scorer, by leveraging the strengths of each expert model. Results show that the proposed method can be used as a general purpose framework for a wide range of zero-shot multimodal tasks, such as image generation, video question answering, mathematical reasoning, and robotic manipulation.

4.1 Introduction

Large pre-trained models have shown remarkable zero-shot generalization abilities, ranging from zero-shot image generation and natural language processing to machine reasoning and action planning. Such models are trained on large datasets scoured from the internet, often consisting of billions of datapoints. Individual pre-trained models capture different aspects of knowledge on the internet, with language models (LMs) capturing textual information in news, articles, and Wikipedia pages, and visual-language models (VLMs) modeling the alignments between visual and textual information. While it is desirable to have a single sizable pre-trained model capturing all possible modalities of data on the internet, such a comprehensive model is challenging to obtain and maintain, requiring intensive memory, an enormous amount of energy, months of training time, and millions of dollars. A more scalable alternative approach is to compose different pre-trained models together, leveraging the knowledge from different expert models to solve complex multimodal tasks.

Building a unified framework for composing multiple models is challenging. Prior works [2, 174] have explored composing pre-trained models in two main ways: (jointly) finetuning models on large datasets, or using common interfaces such as language to combine different models. However, these works have several key limitations: First, simply combining models does not fully utilize each pre-trained model as there is no closed-loop feedback between models. Cascading models, such as Socratic models [174], allows one-way communication but prevents information processed by



Video Question Answering
WATER
mozzarella
Q: How to make the food step by step?
A: Put water in the pot, ..., add sausage, add seasoning on top of the pizza ...
Q: What food is being made? A: Make pizza
G: Language models
E: CLIP models
G: Language models
E: QA classifiers
Grade School Math
Q: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? A: 3
Q: Claire makes a 3 egg omelet every morning for breakfast. How many dozens of eggs will she eat in 4 weeks? A: 7
Iterative Consensus
Generator(G)
Scorers(E)
Image Generation
Hamster
A red car in front of a tree
Grasshopper
G: Diffusion models
E: CLIP models, ..., Image classifiers
G: World models
E: Image segmentation models
Robot Manipulation
Orange mug to the right of orange bowl; ...;
Orange mug on top of orange bowl

Figure 4-1: **Composing models.** The proposed framework that composes a "generator" and an ensemble of "scorers" through iterative consensus enables zero-shot generalization across a variety of multimodal tasks.

later models from propagating back to earlier models to correct errors. Secondly, common interfaces are limited to particular types of models. Language is used as the intermediate connection in Socratic models [174], but a language interface is insufficient to solve many real-world tasks, such as continuous robot control, which requires continuous representations. In addition, Socratic models require pre-designed language templates for the communication between models, which limits scalability. Thirdly, jointly finetuning multiple models [2] requires careful optimization to ensure that the model behaviors remain stable. Such models also require intensive memory and large datasets and can only be used for solving specific tasks.

To resolve these difficulties, we propose a unified framework to compose models in a zero-shot manner* without any training/finetuning. Our framework employs a single model as a generator and an ensemble of scorers. The generator iteratively generates proposals, and each scorer provides a feedback score indicating their agreement. The generator refines its outputs until all the scorers achieve a final consensus. This iterative closed-loop communication between the generator and scorers enables models to correct the errors caused by other models, substantially boosting performance.

The ensemble of scorers is inspired by the idea of "wisdom of the crowds". Each scorer provides complementary feedback to the generator, compensating for the

*By zero-shot, we mean the composed models are never trained together on the evaluation task.

potential weaknesses of other scorers. A Vision-Language scorer, for example, may correct the biases of a language model. We notice that different pre-trained model instances from the same family have a diversity of outputs, which leads to more robust scorers. We demonstrate that guiding the generator with such an ensemble of scorers significantly outperforms a generator guided by a single scorer. In summary, our contributions are:

- First, we propose a unified framework for composing pre-trained models across a variety of tasks, such as image generation, video question answering, mathematical reasoning, and robot manipulation.
- Second, we illustrate how the proposed framework can effectively solve zero-shot multimodal tasks without any training/finetuning. The closed-loop communication between the generator and scorers allows the models to interact with each other to improve performance iteratively.
- Finally, we illustrate how our framework enables the use of ensembles of different pre-trained models as scorers, significantly improving the zero-shot results by leveraging the strengths of multiple expert models.

These observations point to the effectiveness of the proposed method as a general purpose framework for composing pre-trained models for solving various zero-shot multimodal tasks.

4.2 Related Work

Large pre-trained models have shown great success across a variety of domains, such as language generation/translation, image generation, and decision-making.

Language models. Large language models, such as ELMo [113], BERT [22], and GPT-2 [122], are able to achieve state-of-the-art performance on many standard NLP benchmarks. More recent works, such as PALM [16], Chinchilla [55], GPT-3 [13], and GPT-4 [109] further enable few-shot learning from textual prompts.

Vision-language models. Large pre-trained vision-language generative models, such as DALL-E 2 [125], Parti [171], and Imagen [134], can generate high-resolution images given natural language descriptions. Large pre-trained vision-language discriminative models, such as CLIP [119], convert images and languages into the same feature space, achieving remarkable zero-shot generalization ability on downstream tasks.

Decision-making models. Large pre-trained models have been widely applied to solve decision-making tasks, such as learning general purpose policies [128, 84, 143], making planners [58, 1], and learning world models [32]. However, due to the large variability in decision-making tasks, no existing pre-trained models can be readily applied across different tasks.

Composing pre-trained models. Composing large pre-trained models has been widely studied recently. The predominant way to compose pre-trained models is to (joint) finetune them on new tasks [82, 161, 2, 102], but such approaches are computationally expensive. Alternative approaches compose models through a common interface such as language [152, 174]. Other works compose pre-trained models by composing learned probability distributions of the data, such as energy-based models [88, 87, 25], which can be applied to image generation. In this chapter, we propose a framework to compose pre-trained models across a variety of domains without any training or finetuning.

4.3 Method

Given a set of large pre-trained models, we aim to utilize the expert knowledge from different models to solve zero-shot multimodal tasks. We separate pre-trained models into two categories – generators (G) such as GPT [13, 122] and Diffusion models [51] that can generate candidate solutions, and scorers (E) such as CLIP [119] and classifiers that output a scalar score to evaluate each generated solution. We propose **PIC** (composing ensembles of **P**re-trained models via **I**terative **C**onsensus), a framework which composes ensembles of pre-trained models for multimodal tasks. The core idea of PIC is to generate solutions through iterative optimization, where we leverage the knowledge from different models to jointly construct a consensus solution. In PIC, a generator G iteratively and sequentially generates candidate solutions, each of which is refined based on the feedback from a set of scorers. In particular, we seek to obtain a solution $\boldsymbol{x}^*$ such that

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\sim G} \sum_{i=1}^{n} E_i(\boldsymbol{x}), \tag{4.1}$$

where $\{E_1, \cdots, E_n\}$ is the set of scorers. At each iteration, we refine the solutions to have a lower score than the previous iterations. This procedure, described in Equation (4.1), converges to a solution that minimizes the energy across multiple pre-trained models, which maximizes the agreement between the generator and scorers. In contrast to Socratic Models where different pre-trained models are called sequentially, the closed-loop iterative refinement through which we obtain $\boldsymbol{x}^*$ enables the generator and scorers to communicate with each other to reach a consensus on the final solution.

Below, we illustrate how PIC can be broadly applied across tasks in image generation, video question answering, grade school math, and robot manipulation. To optimize Equation (4.1), we consider two different optimization procedures – either a continuous approach that leverages the gradients of each scorer $E_i(\boldsymbol{x})$ or a discrete approach that directly samples possible solutions.

4.3.1 Applications to zero-shot tasks

Image generation. We first apply the proposed framework to image generation to generate images conditioned on a text description or a class label. We use the reverse diffusion process of GLIDE [103], a text-guided diffusion model, as the generator to generate image proposals. At each step of the diffusion process (corresponding to a step of the iterative refinement), we use the gradient from an ensemble of scorers, such as CLIP [119], to guide and update the generated proposals. We iteratively repeat this procedure until the final step.

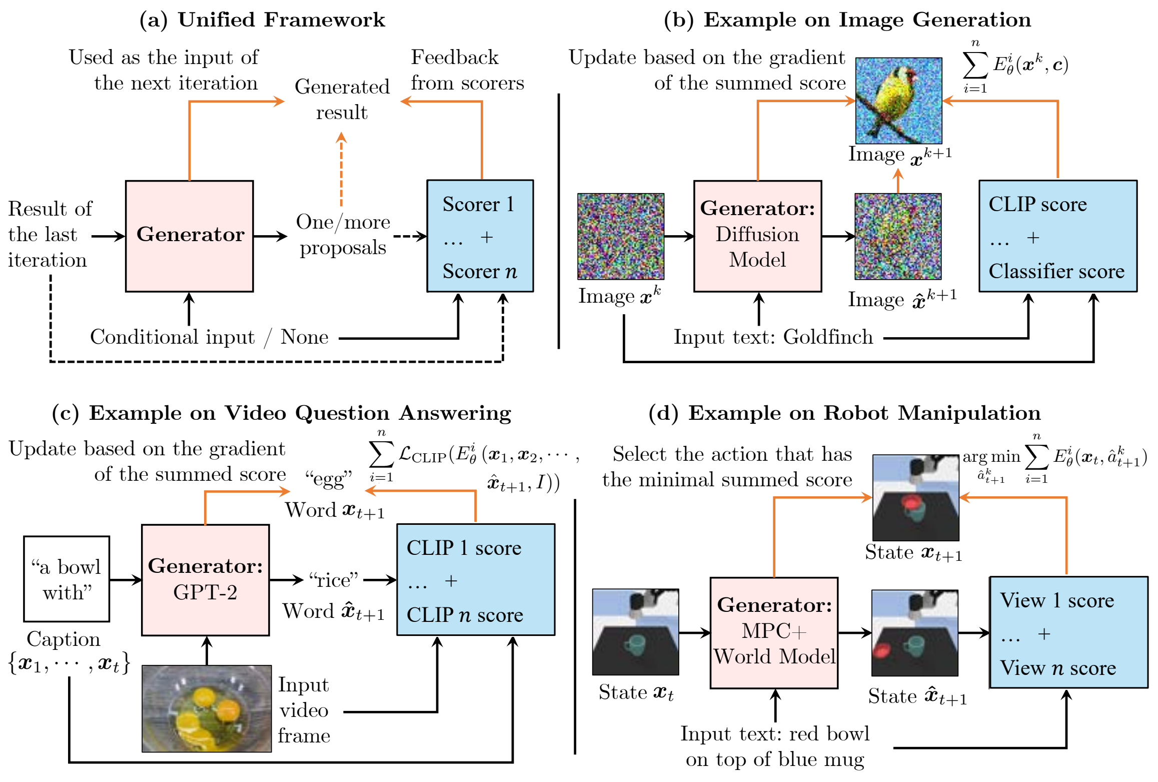

(a) Unified Framework
Used as the input of the next iteration
Generated result
Feedback from scorers
Result of the last iteration
Generator
One/more proposals
Scorer 1
... +
Scorer n
Conditional input / None
(b) Example on Image Generation
Update based on the gradient of the summed score
$\sum_{i=1}^{n} E_\theta^i(\boldsymbol{x}^k, \boldsymbol{c})$
Image $\boldsymbol{x}^{k+1}$
Image $\boldsymbol{x}^k$
Generator: Diffusion Model
Image $\hat{\boldsymbol{x}}^{k+1}$
CLIP score
... +
Classifier score
Input text: Goldfinch
(c) Example on Video Question Answering
Update based on the gradient of the summed score
$\sum_{i=1}^{n} \mathcal{L}_{\text{CLIP}}(E_\theta^i(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \hat{\boldsymbol{x}}_{t+1}, I))$
"egg"
Word $\boldsymbol{x}_{t+1}$
"a bowl with"
Caption $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t\}$
Generator: GPT-2
"rice"
Word $\hat{\boldsymbol{x}}_{t+1}$
CLIP 1 score
... +
CLIP n score
Input video frame
(d) Example on Robot Manipulation
Select the action that has the minimal summed score
$\arg\min_{\hat{a}_{t+1}^k} \sum_{i=1}^{n} E_\theta^i(\boldsymbol{x}_t, \hat{a}_{t+1}^k)$
State $\boldsymbol{x}_{t+1}$
State $\boldsymbol{x}_t$
Generator: MPC+ World Model
State $\hat{\boldsymbol{x}}_{t+1}$
View 1 score
... +
View n score
Input text: red bowl on top of blue mug

Figure 4-2: **The proposed framework and examples on three representative tasks.** (a) Overview of the proposed unified framework. Dashed lines are omitted for certain tasks. (b) Image generation. A pre-trained diffusion model is used as the generator, and multiple scorers, such as CLIP and image classifiers, are used to provide feedback to the generator. (c) Video question answering. GPT-2 is used as the generator, and a set of CLIP models are used as scorers. (d) Robot manipulation. MPC+World model is used as the generator, and a pre-trained image segmentation model is used to compute the scores from multiple camera views to select the best action. Orange lines represent the components used to refine the generated result.

As shown in Fig. 4-2 (b), the image $\boldsymbol{x}^k$ generated at iteration k is first sent to the diffusion model to generate an image proposal $\hat{\boldsymbol{x}}^{k+1}$. Each scorer outputs a score to evaluate whether the generated image matches the given text input. For example, CLIP computes the cosine similarity between the image and text features as the score. The scores generated by different scorers are summed, and their gradient with respect to $\boldsymbol{x}^k$ is used to compute the next reverse prediction $\boldsymbol{x}^{k+1}$:

$$\boldsymbol{x}^{k+1} \leftarrow \hat{\boldsymbol{x}}^{k+1} - \lambda \nabla_{\boldsymbol{x}^k} \sum_{i=1}^{n} E_\theta^i\left(\boldsymbol{x}^k, \boldsymbol{c}\right), \tag{4.2}$$

where n is the number of scorers and $\boldsymbol{c}$ is the input label. We denote the reverse process prediction as $\boldsymbol{x}^{k+1}$ instead of $\boldsymbol{x}^{k-1}$ (used by most diffusion models) to keep the consistent notation across tasks.

Video question answering (VQA). Caption generation for a single video frame is shown in Fig. 4-2 (c). We use GPT-2 as the generator and multiple different CLIP models, trained with different configurations, as the scorers. Given a video frame I, we generate a sequence of words to describe it. To integrate feedback from scorers to the generator, similar to [152], we define a context cache C_t (a set of embedding functions in GPT-2) that stores the context information generated so far, which is updated iteratively based on the feedback from scorers. The prediction of the next word from the generator G is given by $\boldsymbol{x}_{t+1} = G(\boldsymbol{x}_t, C_t)$. To update C_t, we first use G to generate a set of candidate words $\hat{\mathbf{X}}_{t+1} = \{\hat{\boldsymbol{x}}_{t+1}\}$, and then use the feature distance (after softmax) between each sentence (the concatenation of previous words and each new word $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \hat{\boldsymbol{x}}_{t+1}\}$, where $\hat{\boldsymbol{x}}_{t+1} \in \hat{\mathbf{X}}_{t+1}$) and the video frame as the probability of them matching. The CLIP score is the cross-entropy loss $\mathcal{L}_{\text{CLIP}}$ between this new probability distribution and the original distribution of the next word obtained from the generator G. The gradient of the summed score (multiple CLIP models) is then propagated to G to update C_t:

$$C_t^{k+1} \leftarrow C_t^k + \lambda \nabla_{C_t^k} \sum_{i=1}^{n} \mathcal{L}_{\text{CLIP}}(E_\theta^i(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \hat{\boldsymbol{x}}_{t+1}, I)), \tag{4.3}$$

where k is the step of iterative refinement. After several iterations, the updated C_t is used to generate the next token $\boldsymbol{x}_{t+1} = G(\boldsymbol{x}_t, C_t)$. We repeat this process until we generate the entire caption. We cascade the captions of multiple video frames and questions about this video to prompt GPT-3 for video question answering (See Appendix C.1.2 for more details).

Grade school math. We further apply PIC to solve grade school math problems. We use GPT-2 as the generator and treat the grade school math problem as a text-generation problem. The scorer, a pre-trained question-solution classifier, provides the generator feedback to guide the next token's generation $\boldsymbol{x}_{t+1}$. We follow the approach used in VQA to iteratively optimize the generations based on the feedback from scorers. Our generator G first generates a set of candidate words $\hat{\mathbf{X}}_{t+1} = \{\hat{\boldsymbol{x}}_{t+1}\}$, and then the classifier predicts the probability of each solution (the concatenation of previous words and each new word $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \hat{\boldsymbol{x}}_{t+1}\}$, where $\hat{\boldsymbol{x}}_{t+1} \in \hat{\mathbf{X}}_{t+1}$) matching the given question. The classifier score is the cross-entropy loss between this new probability distribution and the original distribution of the next word obtained from the generator G. The gradient of the classifier score is used to update C_t through iterative refinement, same as Eq. (4.3). The updated C_t is used to predict the next word $\boldsymbol{x}_{t+1} = G(\boldsymbol{x}_t, C_t)$. We repeat this process until we generate the complete solution.

Robot manipulation. Finally, we illustrate how PIC can be applied to manipulate objects in the robot environment to conform to a set of object relations such as "red bowl on top of blue mug" shown in Fig. 4-2 (d). We use the combination of Model Predictive Control (MPC) [163] and the World Model as the generator. At each time step, we first use MPC to sample a set of possible actions and then render the state images (after executing an action) from multiple camera views using the world model. For each action, the scorer computes a summed score across all camera views as its final score, which is used to select the best action to execute. Thus, in this domain, the ensemble consists of scorers based on different views of the scene.

For the generator, we assume that there is a pre-trained model, *i.e.* world model, that can accurately render and simulate the dynamic changes in the robot world. Since such a large pre-trained model does not directly exist, we approximate it using an environment simulator combined with MPC as the generator. For the scorer, we use the pre-trained ViLD [42] to generate segmentation maps for images captured by n different camera views, and the corresponding text label for each segment, which are used to obtain object relations. We compare the generated object relations and the relations specified by the text description to obtain the score, *i.e.* score equals 0 if they match; otherwise, 1 (here the score means the distance) (see Appendix C.1.4 for details). To obtain a final world state $\boldsymbol{x}_T$ that satisfies the specified relations, and the action sequence $\{a_1, \cdots, a_T\}$ that manipulates the objects into the final state $\boldsymbol{x}_T$, the generator iteratively samples possible actions $\hat{a}^k_{t+1}$ and gets feedback from scorers. The best action is selected as:

$$a_{t+1} = \underset{\hat{a}^k_{t+1}}{\arg\min} \sum_{i=1}^{n} E^i_\theta(\boldsymbol{x}_t, \hat{a}^k_{t+1}). \tag{4.4}$$

Each scorer, E^i_θ, outputs a score for the resultant state obtained when a candidate action $\hat{a}^k_{t+1}$ is applied to the current world state $\boldsymbol{x}_t$. We execute a_{t+1} in the environment and get a new state $\boldsymbol{x}_{t+1}$. We repeat this process until the task is accomplished or we are at the final step T.

4.4 Experiment Setup

We evaluate the proposed framework for composing pre-trained models on four representative tasks, including image generation, video question answering, grade school math, and robot manipulation.

Image generation. We first show that composing the pre-trained image generator and scorer models such as CLIP enables effective zero-shot image generation. We evaluate the image generation results on ImageNet [21] with the image resolution of 64×64. The class labels are used as the text input to guide image generation. Each method generates 50 images for each class. We evaluate the image generation quality using Inception Score (IS) [135], Fréchet Inception Distance (FID) [44], and Kernel Inception Distance (KID) [11]. IS measures the distribution of generated images. Higher values mean the models can generate more distinct images. FID considers the distributions of both generated images and real images. Lower scores represent that the generated images are closer to the real images. KID is similar to FID, measuring the similarity between two data distributions, but is in the kernel space.

Video question answering. We evaluate methods for solving VQA tasks on ActivityNet-QA [172]. Our method generates free-form language answers instead of selecting an answer from a pre-defined answer set [168, 80]. To evaluate such free-form VQA, we ask workers from Amazon Mechanical Turk to measure whether the generated answer matches the given question and video (See Appendix C.2 for IRB approval and experimental details). For fair comparisons, all the approaches answer the same 300 video questions, and each answer is evaluated by three different workers. The accuracy rate and vocabulary size are reported. An answer is correct if at least two workers believe it is correct. The accuracy rate is the percentage of correctly answered questions over all the questions. To evaluate the diversity of generated answers, we also report the vocabulary size (*i.e.*, the number of words) of answers generated by each method.

Grade school math. GSM8K [17] is a dataset for grade school math problems. Each problem consists of a question, intermediate analyses, and a final solution. We evaluate approaches to solving problems on the 1K test set. We use beam search to generate candidate solutions. The accuracy of beam size=1 and beam size=5 are reported. For beam size=1, we mark the result as correct if it matches the final solution. For beam size=5, we mark the result as correct if any of the five generated results matches the solution.

Robot manipulation. We evaluate how pre-trained models may be used to manipulate objects in Ravens [173]. In Ravens, the action space of the robot is to drop an object at a 2D location on the table. The goal is to obtain a scene configuration that satisfies the object relations specified by a textual description or a real-world image, such as "blue mug to the left of purple bowl". The task is successful if the object relations in the final state satisfy all the relations specified by the input text or image. We report the success rate of tasks with two and three specified object relations.

4.5 Experiments

We compare the proposed method with baselines on the above four zero-shot tasks.

Table 4.1: **Image generation results on ImageNet.** Our PIC can compose the pre-trained generator (G) and scorers (E) through iterative optimization. Composing multiple scorers further boosts performance.

Method Name	**Generator**	**Scorer**	**IS** $\uparrow$	**FID** $\downarrow$	**KID** $\downarrow$
PIC (G+E1)	GLIDE	CLIP	25.017	30.462	6.174
PIC (G+E2)	GLIDE	CLS	22.077	30.871	7.952
PIC (G+E3)	GLIDE	CLS-FREE	25.926	29.219	5.325
PIC (G+E1+E2+E3)	GLIDE	CLIP + CLS + CLS-FREE	**34.952**	**29.184**	**3.766**

4.5.1 Image generation

We evaluate the zero-shot conditional image generation on ImageNet in Table 4.1. We first show the results of composing a single generator (G) and a single scorer (E). We compose GLIDE [103] with three different types of scorers, respectively. E1 is CLIP [119] that computes the cosine similarity between the image and text features. We use the negative cosine similarity as the score. E2 is the image classifier (CLS) [23] that provides the probability of the image matching the text label. We use the negative probability as the score. E3 is the classifier-free guidance (CLS-FREE) [53] which can be treated as an implicit classifier that directly provides pixel-wise gradient feedback to the generated image (Appendix C.1.1). We then compose the generator with all scorers, *i.e.*, G+E1+E2+E3. Composing the generator and a single scorer allows zero-shot image generation. Composing multiple scorers significantly outperforms a single scorer. We note that the generator is not trained on ImageNet; thus the results in Table 4.1 cannot be directly compared with methods trained on ImageNet.

4.5.2 Video question answering

Quantitative results. We compare PIC with one of the state-of-the-art VQA approaches, *i.e.*, JustAsk [168], on ActivityNet-QA [172]. In Table 4.2, JustAsk (FT) is finetuned on ActivityNet-QA, thus achieving the best results. We then compare PIC with JustAsk (Pretrain) for zero-shot VQA. The generator of our method, GPT-2 (medium size), is trained on Webtext [122] using the Huggingface library [164]. Our scorers are CLIP models [119, 130] trained on different datasets or using different configurations. PIC (G+E1) outperforms JustAsk (Pretrain) by %7.72. Composing

Table 4.2: **Video question answering results on ActivityNet-QA.** JustAsk (FT) is finetuned on ActivityNet-QA, thus achieving the best results. For zero-shot VQA, our method (PIC) significantly outperforms JustAsk (Pretrain), one of the best VQA methods. Using multiple scorers further improves performance.

Method Name	**Zero-Shot**	**Generator**	**Scorer**	**Accuracy ↑**	**Vocab ↑**
JustAsk (FT)	No	-	-	**64.667**	160
JustAsk (Pretrain)	Yes	-	-	50.671	210
PIC (G+E1)	Yes	GPT-2	CLIP-32	58.389	267
PIC (G+E1+E2+E3)	Yes	GPT-2	CLIP-32 + CLIP-14 + CLIP-multi	**61.168**	**304**

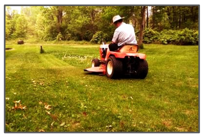
One video frame		JustAsk (Pretrain)	PIC (G+E1)	PIC (G+E1+E2+E3)	JustAsk (FT) (not zero-shot)
	Q: is the person in blue a man or a woman?	A: no	A: woman	A: woman	A: yes
	Q: what happened before the pole vault?	A: audience cheered	A: the person in the video is stretching	A: the athlete is running	A: magnesia powder
	Q: how many people are there in the video?	A: no	A: 1	A: 4	A: 2
	Q: what kind of trousers does the woman wearing yellow clothes look like?	A: tight trousers	A: short	A: short	A: short
	**Q:** what is the person with hat doing?	A: nursing bicycle	A: using a pickup	A: cutting grass	A: weed
	Q: what happened to the person in the hat before the engine?	A: someone passed through	A: put on the helmet	A: turn on the engine	A: speech

Figure 4-3: **Video question answering example results.** Our approach successfully identifies gender and clothing, but its failure to count objects is a reflection of GPT-2 and CLIP's inability to count.

more scorers further improves the accuracy by %2.78. In addition, the vocabulary size of answers generated by our method is larger than other approaches, indicating that our method can answer questions using richer language and more diverse phrasing. Note that our method solves a "more challenging" problem than JustAsk (Pretrain) and JustAsk (FT). Our method generates open-language answers while JustAsk (Pretrain) and JustAsk (FT) select an answer from a pre-defined answer set. Generating free-form responses requires both semantic and grammatical correctness. PIC performs well on both these dimensions while also using a richer vocabulary.

Qualitative results. In Fig. 4-3, we show answers generated by different approaches given a video (only showing a single video frame) and questions. Our approach successfully identifies gender and clothing, but none of the approaches know how to

Table 4.3: **Grade school math results on GSM8K.** Our method (PIC) that composes GPT-2 and a pre-trained question-solution classifier significantly outperforms the baselines, including GPT-FT that is finetuned on GSM8K.

Method Name	Generator	Scorer	BS=1 ↑	BS=5 ↑
GPT-Pretrain	GPT-2 (Pretrain)	-	1.744	12.206
GPT-FT	GPT-2 (FT)	-	3.487	18.271
PIC (G+E)	GPT-2 (Pretrain)	CLS	**16.831**	**20.773**

Grade school math questions	Ground Truth	GPT Pretrain	GPT FT	PIC (G+E)
Q: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?	60	A: 25%	A: 20	A: 60
Q: Melanie is a door-to-door saleswoman. She sold a third of her vacuum cleaners at the green house, 2 more to the red house, and half of what was left at the orange house. If Melanie has 5 vacuum cleaners left, how many did she start with?	18	A: 5	A: 15	A: 18
Q: A fog bank rolls in from the ocean to cover a city. It takes 10 minutes to cover every 3 miles of the city. If the city is 42 miles across from the oceanfront to the opposite inland edge, how many minutes will it take for the fog bank to cover the whole city?	140	A: 10	A: 10	A: 140

Figure 4-4: **Grade school math example results.** Our method can solve math problems involving addition, subtraction, multiplication, and division.

count numbers.

4.5.3 Grade school math

Quantitative results. In Table 4.3, we compare PIC with two baselines, *i.e.* GPT-Pretrain and GPT-FT, for solving math problems on GSM8K [17]. GPT-Pretrain uses the pre-trained GPT-2 (medium-size GPT-2 trained on Webtext using Huggingface) to generate numeric strings. GPT-FT is based on GPT-Pretrain and then finetuned on GSM8K. Our method uses the same GPT-2 (Pretrain) as the generator and a question-solution classifier (CLS) as the scorer. The classifier is trained on GSM8K to distinguish whether a solution is correct for a given question. We surprisingly find that PIC achieves significantly better performance than GPT-FT (%13.344 higher on beam size=1), even though the generator has never seen the math problems before. The classifier only provides feedback to the generator, but through iterative refinement, combining a generator and a scorer without joint training is more effective than directly finetuning GPT-2 on GSM8K (we find the overfitting problem when finetuning GPT-2 on GSM8K).

Table 4.4: **Robot manipulation results on Ravens.** PIC can manipulate objects to achieve object relations specified by textual descriptions (Text) or real-world images (Image). Using scorers of multiple camera views substantially improves the success rate.

Method Name	**2 Relations**		**3 Relations**	
	Text ↑	Image ↑	Text ↑	Image ↑
PIC (G+E1)	35.0	27.5	50.0	45.0
PIC (G+$\sum_{i=1}^{5}$ E$_i$)	**67.5**	**52.6**	**75.0**	**65.3**

Qualitative results. Example results of different methods are shown in Fig. 4-4. Our method can solve math problems involving addition, subtraction, multiplication, and division, even for solutions with three-digit numbers. In contrast, GPT-FT often fails to understand math problems.

4.5.4 Robot manipulation

Quantitative results. We evaluate the proposed method of manipulating objects to achieve object relations specified by the textual descriptions (Text) or real-world images (Image). The success rate of two object relations and three object relations are reported in Table 4.4. We find that using scorers of multiple camera views substantially improves the accuracy on both settings.

Qualitative results. Figure 4-5 shows the example results of the proposed method manipulating objects to accomplish the given task. Our method enables zero-shot robot manipulation on objects with different sizes, colors, and shapes given either the language goal or image goal.

4.6 Analysis

PIC exhibits effective zero-shot generalization ability on a variety of tasks. To further understand the source of such generalization, we investigate two key components in PIC, *i.e.*, the composition of multiple scorers (consensus optimization) (Section 4.6.1) and the iterative refinement (Section 4.6.2).

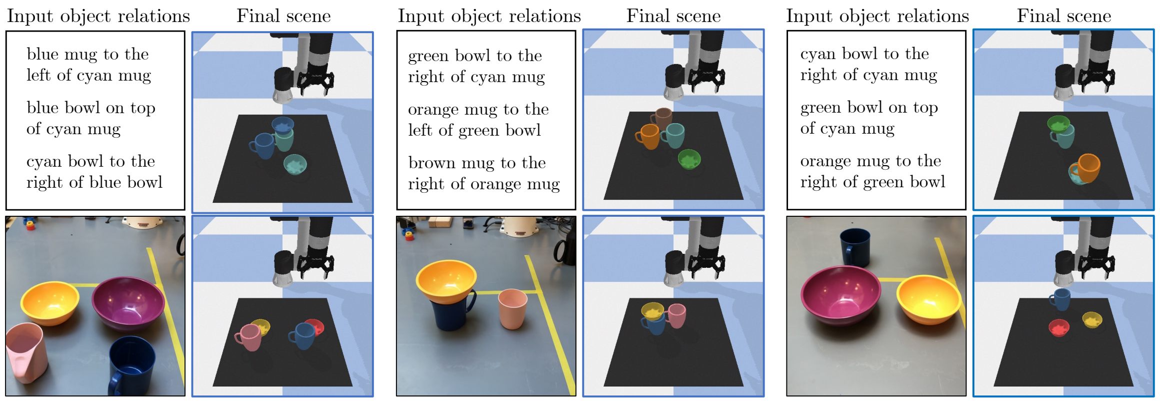

Input object relations
Final scene
blue mug to the left of cyan mug
blue bowl on top of cyan mug
cyan bowl to the right of blue bowl
Input object relations
Final scene
green bowl to the right of cyan mug
orange mug to the left of green bowl
brown mug to the right of orange mug
Input object relations
Final scene
cyan bowl to the right of cyan mug
green bowl on top of cyan mug
orange mug to the right of green bowl

Figure 4-5: **Robot manipulation example results.** The robot manipulates objects to achieve certain object relations that are specified by textual descriptions (first row) or real-world images (second row).

4.6.1 Effect of consensus optimization

We have shown that composing multiple scorers contributes to zero-shot generalization. We further explore the influence of gradually adding each new scorer on the zeros-shot performance.

Image generation. In Table 4.5, we first show the results of composing GLIDE and the CLIP scorer. We then gradually add a new scorer, the image classifier or classifier-free guidance, each time. Finally, we report the results of composing the generator and all scorers. The performance improves every time we add a new scorer, indicating that composing multiple scorers improves zero-shot performance.

Robot manipulation. In Table 4.7, we analyze the effect of composing multiple scores on robot manipulation. The goal is specified by textual descriptions. Composing scores from multiple views, PIC (G+$\sum_{i=1}^{3} E_i$) and PIC (G+$\sum_{i=1}^{5} E_i$), leads to higher accuracy.

4.6.2 Effect of iterative refinement

Next, we explore the influence of iterative refinement on zero-shot generalization, *i.e.*, the feedback loop between the generator and scorers. We compare PIC with baselines

Table 4.5: **Effect of composing multiple scorers.** Image generation results on ImageNet. Gradually adding new scorers keeps improving the performance, indicating that composing multiple scorers contributes to zero-shot image generation.

Method Name	Generator	Scorer	IS $\uparrow$	FID $\downarrow$	KID $\downarrow$
PIC (G+E1)	GLIDE	CLIP	25.017	30.462	6.174
PIC (G+E1+E2)	GLIDE	CLIP + CLS	30.438	29.543	5.435
PIC (G+E1+E3)	GLIDE	CLIP + CLS-FREE	30.500	29.726	4.304
PIC (G+E1+E2+E3)	GLIDE	CLIP + CLS + CLS-FREE	**34.952**	**29.184**	**3.766**

Table 4.6: **Effect of iterative refinement.** Grade school math results on GSM8K. PIC with iterative refinement outperforms baselines where the scorer only provides feedback to the generator at the end stage ($t = T$). BS is the beam search size.

Method Name	Generator	Scorer	Interaction	BS=1 $\uparrow$
GPT-Pretrain+E	GPT-2 (Medium) (Pretrain)	CLS	$t = T$	9.704
GPT-FT+E	GPT-2 (Medium) (FT)	CLS	$t = T$	14.481
PIC (G+E)	GPT-2 (Medium) (Pretrain)	CLS	$t = \{1, \cdots, T\}$	**17.210**

that compose the generator and scorers, but with the scorers only providing feedback to the generator at the end.

Grade school math. In Table 4.6, the baselines, GPT-Pretrain+E and GPT-FT+E, generate five proposal solutions for a given math problem. Then the scorer, *i.e.*, the same question-solution classifier used in PIC, selects the best solution based on its score. PIC iteratively refines the generated answer while the baselines refine the entirely generated solutions at the end. PIC and GPT-Pretrain+E use the same generator and scorer, but PIC outperforms GPT-Pretrain+E by %7.507. PIC also achieves better performance than GPT-FT+E, which uses a stronger generator (finetuned on the GSM8K dataset).

Robot manipulation. In Table 4.7, the baseline, No-IR (G+$\sum_{i=1}^{5} E_i$), first samples 100 trajectories without using the feedback from scorers. Then the scorers select the best trajectories based on the summed score. The generator and scorers of this baseline are the same as our method, *i.e.*, PIC (G+$\sum_{i=1}^{5} E_i$), but our method outperforms the baseline by %37.5 on the "2 Relations" setting, indicating the effectiveness of iterative refinement in the proposed framework.

Together, these results show that the composition of multiple scorers and iterative

Table 4.7: **Effect of composing multiple scorers and iterative refinement on robot manipulation.** Both components are important for zero-shot generalization.

Method Name	Interaction	2 Relations ↑	3 Relations ↑
PIC (G+E1)	$t = \{1, \cdots, T\}$	35.0	50.0
PIC $(\mathbf{G}+\sum_{\mathbf{i=1}}^{\mathbf{3}} \mathbf{E_i})$	$t = \{1, \cdots, T\}$	57.5	63.3
PIC $(\mathbf{G}+\sum_{\mathbf{i=1}}^{\mathbf{5}} \mathbf{E_i})$	$t = \{1, \cdots, T\}$	**67.5**	**75.0**
No-IR $(\mathbf{G}+\sum_{\mathbf{i=1}}^{\mathbf{5}} \mathbf{E_i})$	$t = T$	30.0	46.6

refinement are both important for zero-shot generalization. These results point to the potential broader applicability of the proposed method as a general purpose framework for zero-shot multimodal tasks.

4.7 Conclusion

In this chapter, we propose a unified framework for composing ensembles of pre-trained models through iterative consensus without any training or finetuning. Our framework consists of a generator and an ensemble of scorers. The scorers provide feedback to the generator to iteratively improve its generated results. We show that the proposed method allows effective zero-shot generalization on four representative tasks, *i.e.*, image generation, video question answering, grade school math, and robot manipulation, and even outperforms methods that directly fine-tune models on certain tasks. We further analyze the source of such zero-shot generalization by exploring the effect of the composition of multiple scorers and the iterative refinement, and find that both are important for zero-shot generalization.

Our framework enables the composition of separately trained models and boosts performance by leveraging the knowledge from multiple expert models. The scorers can be learned at different times on different data in an incremental learning manner, enabling the combination of incrementally learned knowledge. Our framework thus paves the way for many potential applications in lifelong learning and continual learning settings.

As our method does not need any training or finetuning, one drawback is that its performance depends on the pre-trained models. Thus effective zero-shot generalization also requires powerful pre-trained models. Training large models is complementary to the framework and methods we proposed and may be directly applied. In the next chapter, we will introduce how to obtain pre-trained models efficiently.

Part III

Transferring Compositionality

Chapter 5

Pre-Trained Language Models for Interactive Decision-Making

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, Jacob Andreas, Igor Mordatch, Antonio Torralba, Yuke Zhu; NeurIPS 2022.

In Part I and Part II, we introduce constructing compositional structures upon pre-trained models, enhancing their compositionality. In this part, we focus on the methodology of acquiring pre-trained models enriched with substantial prior knowledge of fundamental concepts as well as their combinations.

Recent progress illustrates that sufficiently large-scale models such as GPT-4 [109] and DALL-E 2 [125] demonstrate the emergence of compositionality. It is important to note that the training of such models often requires extensive datasets, a process that can be both financially expensive and challenging to scale for a wide range of real-world applications. In fact, for many practical scenarios, achieving the same level of data availability and quality might be unfeasible or even impossible. One question to ask is whether we can transfer the compositionality in pre-trained models to solve new tasks without using extensive training data.

Large pre-trained language models (LMs) have shown remarkable composition ability in language tasks. We propose an approach for using LMs to scaffold learning and generalization in general sequential decision-making problems. This approach leverages pre-trained language models as a starting point, facilitating efficient knowledge transfer from well-established models to new tasks. We demonstrate that this framework reduces the need for extensive training on large datasets, and enables effective combinatorial generalization across different environments and supervisory modalities. The ability to leverage existing knowledge could accelerate the development of AI systems for real-world problems.

5.1 Introduction

Language models (LMs) play a key role in machine learning approaches to natural language processing tasks [22]. This includes tasks that are not purely linguistic, and require nontrivial planning and reasoning capabilities [94, 47]: for example, instruction following, vision-language navigation, and visual question answering. Indeed, some of these tasks are so distant from language modeling that one can ask whether pre-trained LMs can be used as a general framework even for tasks that involve no language at all. If so, how might these capabilities be accessed in a model trained only to process and generate natural language strings?

We study these questions through the lens of embodied decision-making, investigating the effectiveness of LM pre-training as a general framework for learning policies across a variety of environments. We propose **PIC**, a framework that uses Pre-Trained **L**anguage Models for **I**nteractive **D**ecision-Making. As shown in Figure 5-1 (right), we encode the inputs to a policy—including observations, goals, and history—as a sequence of embeddings. These embeddings are passed to a policy network initialized with the parameters of a pre-trained LM, which is fine-tuned to predict actions. This framework is broadly applicable, accommodating goals and environment states represented as natural language strings, image patches, or scene graphs.

We find that imitation learning using pre-trained LMs as policy initializers improves in-domain performance and enables strong generalization over novel tasks. For i.i.d. training and evaluation tasks, this approach yields 20% more successful policies than

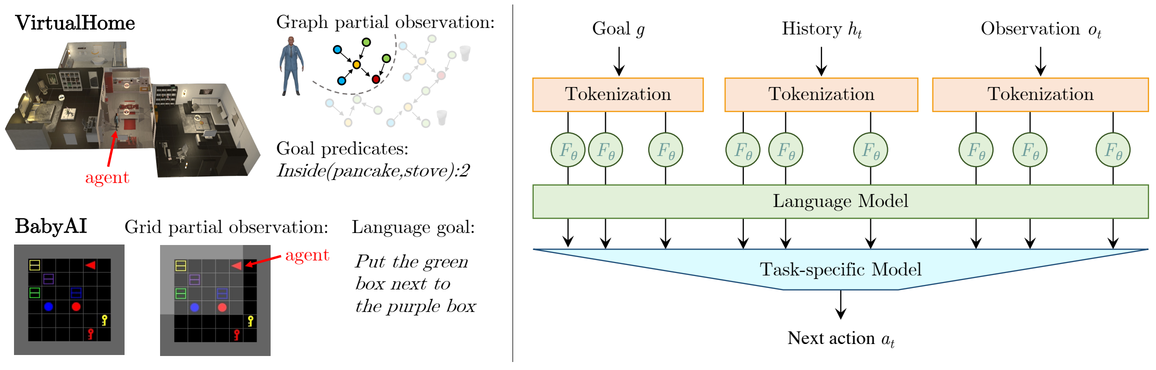


Figure 5-1: **Environments (left):** Different environments have different types of observations and goals. **Our approach (right):** We use pre-trained LMs as a general framework for interactive decision-making by converting policy inputs into sequential data. Such a method enables effective combinatorial generalization to novel tasks.

other baseline methods in VirtualHome [117]. For combinatorial generalization to out-of-distribution tasks, *i.e.*, tasks involving new combinations of goals, states or objects, LM pre-training confers even more benefits: it improves task completion rates by 43.6% for novel tasks (see Fig. 5-3). These results hold for a variety of environment representations: encoding states as natural language strings, when possible, improves the data efficiency of training, but even LMs fine-tuned on random environment encodings generalize combinatorially to new goals and states when trained on large enough datasets.

We further examine how our method may be used in environments where expert data is not available, and agents must instead actively gather data. To do this, we integrate an **A**ctive **D**ata **G**athering (**ADG**) procedure into pre-trained LMs. Our proposed approach to ADG consists of three parts. First, exploration collects trajectories using a mix of random actions and actions generated by the current policy. Exploration is insufficient in this high dimensional problem and most of the trajectories will likely fail to achieve the end goal. A key insight is that even the failed trajectories contain useful sub-trajectories that solve certain sub-goals, and we relabel these goals in a hindsight relabeling stage. The relabeled goal describes what was achieved in the extracted sub-trajectory. The policy update stage samples relabeled trajectories to update the policy. The active data gathering procedure allows us to train the LM-policy without pre-collected expert data. It also outperforms reinforcement learning (RL) methods on embodied decision-making tasks and enables more effective compositional generalization to novel tasks.

Finally, we investigate *why* LID contributes to compositional generalization. We hypothesize three possible causes for the effectiveness of LM-based policy initialization: (1) the use of *language-based input encodings*, and more generally LMs' ability to reason about natural language strings; (2) the *sequential structure* of transformer inputs, in contrast to the fixed-sized observations used by most policy architectures, and (3) *task-general inductive bias* conferred by weight initialization with LM pretraining. We investigate (1) by encoding the policy inputs as different types of sequences. Different input encoding schemes have only a negligible impact on the performance: the effectiveness of language modeling is not limited to utilizing natural strings, but in fact extends to arbitrary sequential encodings. We study (2) by encoding observations with a single vector embedding, thereby removing its sequential structure. This operation significantly degrades the model's performance on novel tasks. Finally, we investigate (3) by learning the parameters of the policy from scratch. The success rate after removing the pre-trained LM weights drops by 11.2%, indicating that LM pretraining provides useful inductive bias for sequence processing even when sequences are not natural language strings. To summarize, our work has four main contributions:

- First, we propose to use pre-trained LMs as a general scaffold for interactive decision-making across a variety of environments by converting all policy inputs into sequential data.

- Second, we demonstrate that language modeling improves combinatorial generalization in policy learning: initializing a policy with a pre-trained LM substantially improves out-of-distribution performance on novel tasks.

- Third, we integrate an active data gathering procedure into the proposed approach to further enable policy learning on environments without using pre-collected expert data.

- Finally, we perform several analyses to explain the generalization capabilities of pre-trained LMs, finding that natural strings are not needed to benefit from LM pre-training, but the sequential input encoding and weight pre-training are important.

Our approach starts with pre-trained models, which lets us transfer knowledge effectively from these well-established models to solve new tasks. This means we don't have to spend as much time training on huge amounts of data. These results point to the effectiveness of the proposed framework with pre-trained LMs as a general-purpose framework to promote compositional generalization.

5.2 Related Work

In recent years, word and sentence representations from pre-trained LMs [114, 22, 121] have become ubiquitous in natural language processing [178, 115]. Some of the most successful applications of pre-training lie at the boundary of natural language processing and other domains, as in instruction following [47] and language-guided image retrieval [92].

Learning representations of language. From nearly the earliest days of the field, natural language processing researchers observed that representations of words derived from distributional statistics in large text corpora serve as useful features for downstream tasks [20, 31]. The earliest versions of these representation learning schemes focused on isolated word forms [97, 112]. However, recent years have seen a number of techniques for training (masked or autoregressive) language models to produce contextualized word representations (which incorporate information neighboring words in sentences and paragraphs) via a variety of masked-word prediction objectives [22, 170].

Applications of pre-trained LMs. LMs can be fine-tuned to perform language processing tasks other than language modeling by casting those tasks as word-prediction problems. Successful uses of representations from pre-trained models include syntactic parsing [74] and language-to-code translation [160]; successful adaptations of LM prediction heads include machine translation [178], sentiment classification [13] and style transfer [69]. A number of tasks integrate language and other modalities, including visual question answering and image captioning [169]. Recent works find that image representations can be injected directly into LMs' embedding layers [154].

Policy learning and LM. Traditional policy learning methods, such as PPO [137], DQN [100], DDPG [86], A3C [99], perform well on playing tasks on Atari, OpenAI gym [12], and MuJoCo [153]. Some of them might fail to solve more challenging tasks on embodied environments [117, 141]. Several recent papers [129, 60, 57] propose to use LM for policy learning. Frozen Pretrained Transformer (FPT) [93] demonstrates that pre-trained LMs require very little fine-tuning to *match* the performance of task-specific models on several image classification and numerical sequence processing tasks. Semi-Supervised Skill Learning with Latent Language $(\mathrm{SL})^3$ [138] shows that LMs can serve as an effective backbone for hierarchical policies that express plans as natural language strings [5, 7]. In this chapter, we focus on building a general framework for decision-making tasks using pre-trained LMs, even when language is not provided as an input or output.

5.3 Decision-Making and Language Modeling

5.3.1 POMDPs and policy learning

We explore the application of LMs to general sequential decision-making tasks in partially observed environments. These tasks may be formalized as partially observable Markov decision processes (POMDPs). A POMDP is defined by a set of states, a set of observations, a set of actions, and a transition model $\mathcal{T}(s_{t+1}|s_t, a_t)$ that maps the current state and action to the next state. Importantly, in a POMDP setting, the observation o_t only captures a portion of the underlying state s_t, and an optimal decision-making strategy (a **policy**) must incorporate both the current observation and the history of previous observations and actions. In our experiments, policies are parametric models $\pi_\phi(a_t|g, h_t, o_t)$ that output the probability of an action given the goals g, history information $h_t = \{o_1, a_1, \cdots, o_{t-1}, a_{t-1}\}$, and partial observations o_t of the current state s_t.

In Fig. 5-1 (right), we show a high-level overview of the proposed method. We first convert all policy inputs into a sequence and provide them as input to a transformer encoder. Representations from this encoder model are then passed to a task-specific decoder that predicts actions. We collect a dataset of N training trajectories $\mathcal{D} = \{d^i\}_{i=1}^N$, where each trajectory consists of a goal and a sequence of observations and actions: $d^i = \{g^i, o_1^i, a_1^i, \cdots, o_{T_i}^i, a_{T_i}^i\}$, where T_i is the length of the trajectory. We then train the policy to maximize the probability of actions we want to achieve $\boldsymbol{a}^i = \{a_1^i, \ldots, a_{T_i}^i\}$ across trajectories using the cross-entropy loss:

$$\phi^* = \arg\min_{\phi} \left(-\sum_{i=1}^{N} \sum_{t=1}^{T_i} \ln \pi_\phi(a_t^i | g^i, h_t^i, o_t^i) \right). \tag{5.1}$$

5.3.2 Language models as policy initializers

Our experiments focus on **autoregressive**, **transformer-based LMs** [156]. These models are trained to fit a distribution over a text sequence $\boldsymbol{y} = \{y_i\}_{i=1}^n$ via the chain rule $p(\boldsymbol{y}) = p(y_1) \prod_{i=2}^n p(y_i \mid y_1, \ldots, y_{i-1})$. Each term on the right hand side is parameterized by a transformer network, which accepts the conditioned tokens as input. Each token passes through a learned embedding layer F_θ, then the full conditioned sequence is fed into the LM. In our work, we use a standard LM, GPT-2, to process the input sequence rather than to predict future tokens.

Both POMDP decision-making and language modeling are naturally framed as sequence prediction tasks, where successive words or actions/observations are predicted based on a sequence of previous words or actions/observations. This suggests that pre-trained LMs can be used to initialize POMDP policies by fine-tuning them to model high-reward or expert trajectories, as described below.

5.4 Method

We evaluate the effectiveness of pre-trained LMs in solving decision-making tasks across environments. We use **BabyAI** [59] and **VirtualHome** [117] to evaluate the proposed method. While both environments feature complex goals, the nature of these goals, as well as the state and action sequences that accomplish them, differ substantially across environments (Fig. 5-1 (left)).

5.4.1 Policy network

We first examine whether pre-trained LMs provide effective initializers when states and action histories are represented as natural language strings. We encode the inputs to the policy—including observations, goals, and action histories—as sequences of words. These word sequences are passed to the LM (using its pre-trained word embedding layer F_θ) and used to obtain contextualized token representations. Token representations are averaged and used to predict actions. We design a policy network following the general policy framework proposed in Fig. 5-1.

Environment encodings in VirtualHome. In VirtualHome, each goal consists of a sequence of predicates and multiplicities, and is translated into a templated English sentence (*e.g.*, "`Inside(apple, fridge):2`" becomes "put two apples inside the fridge"). To encode the agent's partial observation, we extract a list of currently visible objects, their states (*e.g.*, "open, clean"), and 3D world coordinates. We use a fully-connected layer to encode the 3D information and generate a feature representation of each object in the observation. To encode history, we store information about all previous actions and convert them into templated English sentences (*e.g.*, "I have put the plate on the kitchen table and the apple inside the fridge").

Environment encodings in BabyAI. The observation by default is a 7×7 grid. We convert the observation into 7×7 text descriptions, *e.g.*, "purple ball", "grey wall", "open door", and combine them into a long sentence. We then convert the history actions into text descriptions, *e.g.*, "turn left" and "go forward". We combine the language instruction (without modification) with the observation and history text descriptions, and feed them to the pre-trained LM.

We note that the policy network described above does not strictly require that these encodings take the form of natural language strings—other encodings of the environment as a sequence also work (see Section 5.7). This framework could also be generalized to support pixel-based observations using discretization schemes like the one employed in the Vision Transformer [24].

Action prediction. We pool LM outputs into a "context representation" that is used to predict the next action. In training, we maximize the probabilities of demonstrated actions. In inference, we select the valid action with the highest probability. See Appendix D.3.1 for details.

VirtualHome and BabyAI have quite different observation spaces, action spaces, and goal spaces; however, we show that embedding policy inputs as sequences and utilizing the pre-trained LM as a policy initializer, enables effective generalization to novel tasks in both environments. We note that PIC is not limited to VirtualHome and BabyAI, but is straightforwardly applicable to other embodied environments, such as ALFRED [144] and iGibson [141].

5.4.2 Training

We first examine PIC through imitation learning on data collected by experts. We then show that integrating an active data gathering procedure into PIC enables policy learning without using expert data. We use VirtualHome as an example to explain the data gathering.

Policy learning with expert data. The policy model is first initialized from a pre-trained LM and then fine-tuned on data collected by experts. We build on the VirtualHome environment to collect a set of expert trajectories using regression planning [75] and create a **VirtualHome-Imitation Learning dataset**. Given a task described by goal predicates, the planner generates an action sequence to accomplish this task (See Appendix D.5.1). The planner has access to privileged

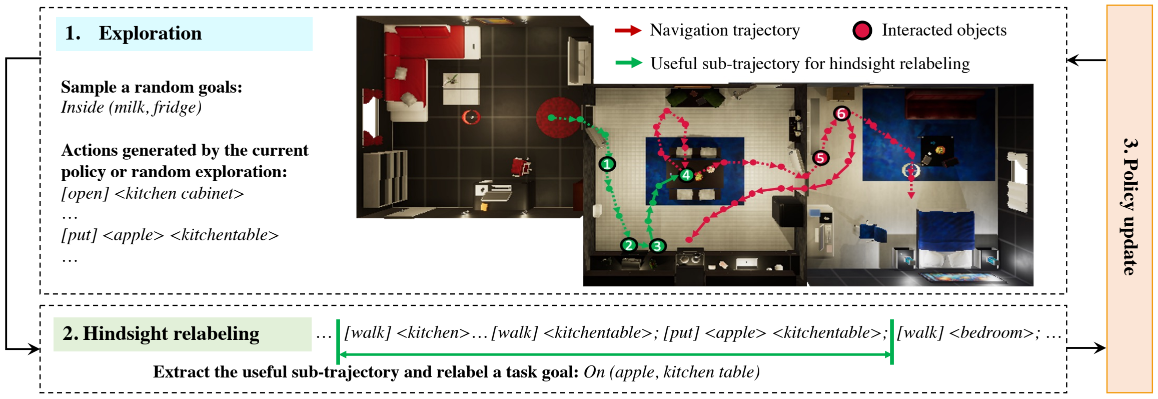

1. Exploration
Sample a random goals:
Inside (milk, fridge)
Actions generated by the current policy or random exploration:
[open] <kitchen cabinet>
…
[put] <apple> <kitchentable>
…
Navigation trajectory
Interacted objects
Useful sub-trajectory for hindsight relabeling
1
2
3
4
5
6
2. Hindsight relabeling
… [walk] <kitchen>… [walk] <kitchentable>; [put] <apple> <kitchentable>; [walk] <bedroom>; …
Extract the useful sub-trajectory and relabel a task goal: On (apple, kitchen table)
3. Policy update

Figure 5-2: **PIC with the active data gathering procedure.** By iteratively repeating the exploration, hindsight relabeling, and policy update, PIC with active data gathering can learn an effective policy without using pre-collected expert data.

information, such as information about the pre-conditions and effects of each action, allowing an agent to robustly perform tasks in partially observable environments and generate expert trajectories for training and evaluation.

Policy learning with active data gathering. Collecting expert data is sometimes challenging. It may require privileged information of the environment or human annotations, which can be time-consuming and difficult to scale. A promising way to scale up supervision is Hindsight Experience Replay (HER) [6], which allows agents to learn from orders of magnitude more data without supervision. However, existing HER methods [37] focus on simple tasks with small state/action space and full observability. They cannot tackle more complicated embodied decision-making tasks, requiring nontrivial planning and reasoning or natural language understanding. PIC with the active data gathering (**PIC-ADG**) can be used in solving tasks in such environments.

As shown in Fig. 5-2, PIC-ADG consists of three stages, *i.e.*, **exploration**, **hindsight relabeling**, and **policy update**. The key idea is to gradually improve the task success rate by asking the agent to iteratively explore the environment, relabel failure samples, and update its policy using imitation learning. In the **exploration** stage, we first randomly sample a goal and an initial state. We then use a mix of random actions and actions generated by the current policy $\pi_\phi(a_t|g, h_t, o_t)$ to obtain the next action. We repeat this process until this episode ends. We collect M trajectories and store them in the replay buffers. The generated actions in the early stages rarely complete the given task. However, even the failed trajectories contain useful sub-trajectories that solve certain sub-goals. In the **hindsight relabeling** stage, we extract useful sub-trajectories and relabel a goal g' for each of them. We design a goal relabel function f_l that generates a goal based on the sequence of observations and actions using hand-designed templates. In practice, we implement the goal relabel function as a program (see Appendix D.5.2). The *hindsight relabeling* stage allows sample-efficient learning by reusing the failure cases. During **policy update**, the agent samples the data from the replay buffers and updates its policy network π_ϕ.

By interleaving the exploration, hindsight relabeling, and policy update, PIC-ADG can gradually improve the policy without requiring pre-collected expert data. In embodied environments with large action spaces, sparse rewards, and long-horizon planning, RL methods often struggle to obtain stable policy gradients during training. Our method enables sample-efficient learning from the sparse rewards by relabeling new goals for the bad samples that the agent fails to achieve. In addition, PIC-ADG leverages the stability of supervised learning in the *policy update* stage, enabling it to outperform RL approaches on a wide range of decision-making tasks.

5.5 Experiment Setup

We evaluate the proposed method and baselines on VirtualHome and BabyAI.

5.5.1 VirtualHome

VirtualHome is a 3D embodied environment featuring partial observability, large action spaces, and long time horizons. We evaluate policies' performance from three aspects: (1) performance on in-distribution tasks; (2) generalization to novel scenes; and (3) compositional generalization to novel tasks.

In-Distribution. The predicate types and their counts in the goal are randomly sampled from the same distribution as the training data. The objects are initially placed in the environment according to common-sense layouts (*e.g.*, plates appear inside the kitchen cabinets rather than the bathtub).

Novel Scenes. The objects are placed in random positions in the initial environment without common-sense constraints (*e.g.*, apples may appear inside the dishwasher).

Novel Tasks. The components of all goal predicates are never seen together during training (*e.g.*, both plates and fridges appear in training goals, but `Inside(plate, fridge)` only appears in the test set. (See Appendix D.6 for more details.)

We evaluate the success rates of different methods on each test set. A given episode is scored as successful if the policy completes its entire goal within the maximum allowed steps of the environment. On each of the 3 test subsets, we use 5 different random seeds and test 100 tasks under each seed. Thus there are $1,500$ examples used to evaluate each model.

5.5.2 BabyAI

BabyAI is a 2D grid world environment for instruction following. Observations in BabyAI are $7 \times 7 \times 3$ grids describing a partial and local egocentric view of the state of the environment. We evaluate the methods on four representative tasks: *GoToRedBall*, *GoToLocal*, *PickupLoc*, and *PutNextLocal*. Performing well on the test set requires the models to generalize to new environment layouts and goals, resulting in new combinations of tasks not seen in training. For each method, we compute success rates over 500 episodes on each task.

5.6 Experiments

We first show the results of the proposed method and baselines for embodied decision-making tasks using expert data in Section 5.6.1. We then show our results when using actively gathered data in Section 5.6.2.

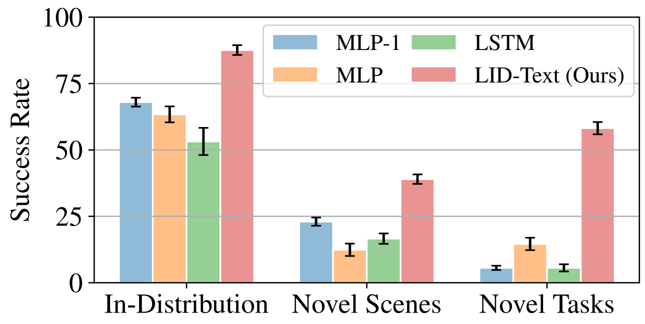


Figure 5-3: **Comparisons of the proposed method and baselines on VirtualHome.** All the methods are trained on expert data using imitation learning. *MLP-1*, *MLP*, and *LSTM* are baselines without using the pre-trained LM. The proposed method, *PIC-Text (Ours)*, outperforms all baselines.

5.6.1 Embodied decision-making with pre-trained language model (PIC)

Results on VirtualHome. We evaluate the following methods: **PIC-Text (Ours)** is the proposed method that converts all environment inputs into text descriptions. The pre-trained LM is fine-tuned for decision-making (conditioned on goals, observations, and histories) as described in Section 5.4.1. **Recurrent Network**. We compare our method with a recurrent baseline using an LSTM [54] to encode the history information. The hidden representation from the last timestep, together with the goal and current observation, is used to predict the next action. **MLP** and **MLP-1**. We perform additional comparisons with baselines that do not use recurrent networks or pre-trained LMs. *MLP* and *MLP-1* take the goal, histories, and the current observation as input and send them to the multilayer perceptron neural network (MLP) to predict actions. *MLP-1* has three more average-pooling layers than *MLP* that average the features of tokens in the goal, history actions, and the current observation, respectively, before sending them to the MLP layer.

Each method is trained on $20K$ demos from the VirtualHome-Imitation Learning dataset and then evaluated on the three test subsets: **In-Distribution**, **Novel Scenes**, and **Novel Tasks**. In Figure 5-3, *PIC-Text (Ours)*, which initializes the policy with a pre-trained LM, has higher success rates than other methods. This difference is most pronounced in the **Novel Tasks** setting, where test tasks require combinatorial

Table 5.1: **Success rates on BabyAI tasks.** All the methods are trained on offline expert data using imitation learning. *PIC-Text (Ours)* outperforms BabyAI-Ori, the method used in the original paper [59].

Tasks	**Methods**	**Number of Demos**				
		100 ↑	**500 ↑**	**1K ↑**	**5K ↑**	**10K ↑**
GoToRedBall	BabyAI-Ori [59]	81.0	96.0	99.0	99.5	99.9
	PIC-Text (Ours)	**93.9**	**99.4**	**99.7**	**100.0**	**100.0**
GoToLocal	BabyAI-Ori [59]	55.9	84.3	98.6	**99.9**	99.8
	PIC-Text (Ours)	**64.6**	**97.9**	**99.0**	99.5	99.5
PickupLoc	BabyAI-Ori [59]	28.0	58.0	93.3	97.9	**99.8**
	PIC-Text (Ours)	**28.7**	**73.4**	**99.0**	**99.6**	**99.8**
PutNextLocal	BabyAI-Ori [59]	**14.3**	16.8	43.4	81.2	97.7
	PIC-Text (Ours)	11.1	**93.0**	**93.2**	**98.9**	**99.9**

generalization across goals that are never seen during training. Here, *PIC-Text (Ours)* dramatically (43.6%) improves upon all baselines. Such combinatorial generalization is necessary to construct general purpose agents, but is often difficult for existing approaches. Our results suggest that pre-trained LMs can serve as a computational backbone for combinatorial generalization.

Results on BabyAI. We use the standard training and test data provided by [59]. In BabyAI, performing well on unseen test tasks with new environment layouts and goals requires combinatorial reasoning. In Table 5.1, we report the success rate of models trained on different numbers of demos. **BabyAI-Ori** [59] is the method used in the original paper. **PIC-Text (Ours)** is the proposed method that converts policy inputs into a text sequence. Given enough training data, *i.e.*, 10K demos, both methods achieve high success rates, but *PIC-Text (Ours)* outperforms BabyAI-Ori with less training data, indicating the proposed method improves sample efficiency when generalizing to novel tasks.

5.6.2 Pre-trained language model with active data gathering (PIC-ADG)

We compare **PIC-ADG**, the proposed LM framework for decision-making using actively gathered data (Section 5.4.2), to a variety of baselines that do not use

Table 5.2: **Comparisons of methods without using expert data on VirtualHome.** *PIC-ADG (Ours)* is the only successful approach.

	In-Distribution ↑	**Novel Scenes ↑**	**Novel Tasks ↑**
Random	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Goal-Object	0.8 ± 0.5	0.0 ± 0.0	0.4 ± 0.4
PPO	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
DQN+HER	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
PIC-ADG (Ours)	**46.7 ± 2.7**	**32.2 ± 3.3**	**25.5 ± 4.1**

pre-collected expert data on VirtualHome.

Random. The agent selects the next action randomly from the valid action space at that state. **Goal-Object.** The agent randomly selects an object that is in the goal and in the valid action space to interact with. For example, given a goal of "`Inside(apple, fridge):1`", this baseline might choose "grab apple", "open fridge", or other actions containing "apple" or "fridge". **Online RL.** We compare with PPO [137], one of the most commonly used online RL methods. For fair comparisons, we equip PPO with the same main policy network as the proposed method. Our implementation is based on Stable Baselines3 [123]. **Hindsight Experience Replay.** We compare with DQN+HER used in [6] and modify its main policy network to be the same as the proposed method.

Quantitative results. We compare PIC-ADG with baselines on VirtualHome in Table 5.2. Each experiment is performed 5 times with different random seeds. The **Random** baseline is always 0, indicating the tasks in VirtualHome cannot be easily solved by a random policy. **Goal-Object** is better than *Random* because *Goal-Object* has access to objects in the goal and it samples actions from a much smaller action space. The online RL baseline, **PPO**, fails to solve tasks in VirtualHome featured by partial observation, large state/action space, and long-term horizon. **DQN+HER** works well on simple tasks in 2D environments, but they cannot tackle VirtualHome tasks either, requiring nontrivial planning and reasoning. PIC-ADG does not require expert data and can solve complicated tasks in 3D embodied environments, which cannot be easily achieved using RL.

Table 5.3: **The proposed method with active data gathering.** PIC-ADG (Ours) can be used as a policy initializer for online RL or a data provider for offline RL.

	In-Distribution ↑	**Novel Scenes ↑**	**Novel Tasks ↑**
PIC-ADG (Ours)	46.7 ± 2.7	**32.2 ± 3.3**	25.5 ± 4.1
PPO (PIC-ADG Init)	**53.7 ± 3.5**	30.2 ± 3.4	**27.8 ± 2.7**
DT (PIC-ADG Data)	42.4 ± 1.5	21.6 ± 2.48	16.8 ± 1.0

Policy initializer and data provider. PIC-ADG can further be used to initialize the weights for fine-tuning RL policies and to gather data for offline learning. As shown in Table 5.2, directly training RL, *e.g.*, PPO, fails to solve tasks in VirtualHome. However, after using the policy trained by PIC-ADG to initialize the PPO policy, we may effectively learn an interactive policy with good performance. In Table 5.3, **PPO (PIC-ADG Init)** is initialized from PIC-ADG and further fine-tuned to solve the tasks in VirtualHome. After initialization, PPO improves its success rate by 53.7% on the *In-Distribution* setting (See PPO results in Table 5.2 and Table 5.3). In addition, PIC-ADG can provide data for offline learning. PIC-ADG saves the relabeled data in replay buffers. We train Decision Transformer (DT) [14] using the data collected by PIC-ADG. See **DT (PIC-ADG Data)** in Table 5.3 [*].

5.7 Analysis: Understanding the Sources of Compositional Generalization

The pre-trained LM policy, fine-tuned on either expert data or actively gathered data, exhibits effective combinatorial generalization. Is this simply because LMs are effective models of relations between natural language descriptions of states and actions [3], or because they provide a more general framework for combinatorial generalization in decision-making? We hypothesize and investigate three possible factors to understand the sources of such combinatorial generalization. We use policies trained on the expert data as an example to explain the experiments.

[*]Note that the results of PIC-Text in Fig. 5-3 and results of PIC-ADG in Table 5.2 are not directly comparable because the difficulty level of the evaluated tasks is different. See Appendix D.6 for more details.

Table 5.4: **Success rates of policies trained with different input encodings in the *Novel Tasks* setting on VirtualHome.** The text encoding is the most sample-efficient, but all models converge to similar performance given sufficient training data.

Methods	**Number of Demos**					
	100 $\uparrow$	**500** $\uparrow$	**1K** $\uparrow$	**5K** $\uparrow$	**10K** $\uparrow$	**20K** $\uparrow$
PIC-Text (Ours)	**8.8** $\pm$ **1.4**	**22.2** $\pm$ **1.7**	26.8 $\pm$ 1.0	46.0 $\pm$ 1.0	**58.2** $\pm$ **1.2**	58.2 $\pm$ 1.6
PIC-Index (Ours)	6.4 $\pm$ 0.6	18.0 $\pm$ 3.8	18.8 $\pm$ 1.0	45.5 $\pm$ 2.1	54.6 $\pm$ 0.8	57.8 $\pm$ 0.9
PIC-Unnatural (Ours)	6.8 $\pm$ 1.3	18.6 $\pm$ 2.1	**27.0** $\pm$ **1.1**	**47.2** $\pm$ **1.7**	55.8 $\pm$ 0.8	**58.8** $\pm$ **0.9**

5.7.1 Input encoding scheme

We first hypothesize that converting environment inputs into natural language contributes to combinatorial generalization as the LMs are trained on language data. We explore the role of *natural language* by investigating three alternative ways of encoding policy inputs to our model without using natural language strings: two in VirtualHome, and one in BabyAI. BabyAI results are in Appendix D.1.

Index encoding in VirtualHome. Rather than natural language strings, *PIC-Index (Ours)* converts policy inputs into integer indices. *PIC-Index (Ours)* retains the discrete, serial format of the goal, history, and observation, but replaces each word with an integer, and replaces the embedding layer from the pre-trained LM with a new embedding layer trained from scratch. For example, *grab apple* is mapped to (5,3) based on the positions of *grab* and *apple* in the vocabulary set.

Unnatural string encoding in VirtualHome. *PIC-Unnatural (Ours)* replaces the *natural language* tokens (e.g., converting the goal "`On(fork, table):1`" to *put one fork on the table*) with random ones (e.g., converting `On(fork, table)` to *brought wise character trees fine yet*). This is done by randomly permuting the entire vocabulary that maps each token to a new token. Such a permutation breaks the semantic information in natural strings.

PIC-Index (Ours) and *PIC-Unnatural (Ours)* have the same policy network as *PIC-Text (Ours)*. All are fine-tuned on the expert data. The averaged results using 5 different random seeds on the Novel Tasks setting are reported in Table 5.4. Given few training data, *e.g.*, 100 demos, all the models perform poorly, with success rates lower

Table 5.5: **Experiments on sequential inputs and weight initialization.** Fine-tuning the pre-trained weights and the usage of sequential encoding are important for combinatorial generalization.

	In-Distribution ↑	**Novel Tasks ↑**
PIC-Text (Ours)	87.6 ± 1.9	**58.2 ± 2.3**
No-Seq	74.0 ± 2.3	2.0 ± 0.6
No-Pretrain	**90.8 ± 2.0**	47.0 ± 2.8
No-FT	51.2 ± 4.5	17.0 ± 2.9

than 10%. *PIC-Text (Ours)* achieves higher success rates than *PIC-Index (Ours)* and *PIC-Unnatural (Ours)* when dataset size increases, *e.g.*, *PIC-Text (Ours)* is around 4% higher than *PIC-Index (Ours)* and *PIC-Unnatural (Ours)* with 500 training demos. When the training dataset is further enlarged, *e.g.*, 20K demos, success rates of all approaches reach similar performance. This result indicates that the effectiveness of pre-trained LMs in compositional generalization is not unique to natural language strings, but can be leveraged from arbitrary encodings, although adapting the model to arbitrary encodings may require more training data.

5.7.2 Sequential input representation

Next, we explore whether generalization requires sequential processing mechanisms in transformer-based LMs. We investigate whether the LM pre-trained policy will still be effective when the input encoding is not sequential. **No-Seq** encodes the goal as a single vector by averaging all goal embeddings. History and observation features are obtained in the same way. All features are then sent to the pre-trained LM to predict actions. As shown in Table 5.5, removing sequential structure significantly hurts performance on *Novel Tasks*. *No-Seq* achieves good performance on test tasks that are closer to training tasks, but cannot generalize well to more challenging unseen tasks. Thus, combinatorial generalization in pre-trained LMs may be attributed in part to transformers' ability to process sequential input representations effectively.

5.7.3 Favorable weight initialization

Finally, we investigate if the favorable weight initialization from LM pre-training enables effective generalization of the proposed model. **No-Pretrain** does not initialize the policy using the pre-trained LM, but instead trains the policy on the expert data from scratch. In Table 5.5, we find that removing the pre-trained weights can fit the in-domain data and thus performs well on the *In-Distribution* setting. However, its success rate is 11.2% lower than the proposed model on the *Novel Tasks* setting, indicating the pre-trained weights are important for effective generalization, but not necessary for effective data fitting. We further test a baseline, **No-FT**, that keeps the pre-trained weights of the language model but freezes them while training the rest of the model on our expert data. Freezing the pre-trained weights without fine-tuning significantly hurts the performance on both settings, suggesting that fine-tuning the transformer weights is essential for effective combinatorial generalization.

Together, these results suggest that sequential input representations (vs. fixed-dimensional feature vectors) and favorable weight initialization are both important for combinatorial generalization, however, the input encoding schemes (e.g., as a natural language string vs. an arbitrary encoding scheme) have little influence. These results point to the potential broader applicability of pre-trained LMs as a computational backbone for compositional embodied decision-making, where arbitrary inputs, such as language, images, or grids, may be converted to sequential encodings.

5.8 Qualitative Results

In Figure 5-4, we show examples of *PIC-Text (Ours)* completing tasks in VirtualHome and BabyAI. We show two successful examples from VirtualHome on the *In-Distribution* and *Novel Tasks* settings, and two successful examples from BabyAI on solving the *GoToLocal* and *PickupLoc* tasks. We only show short trajectories or extract a sub-trajectory for saving space.

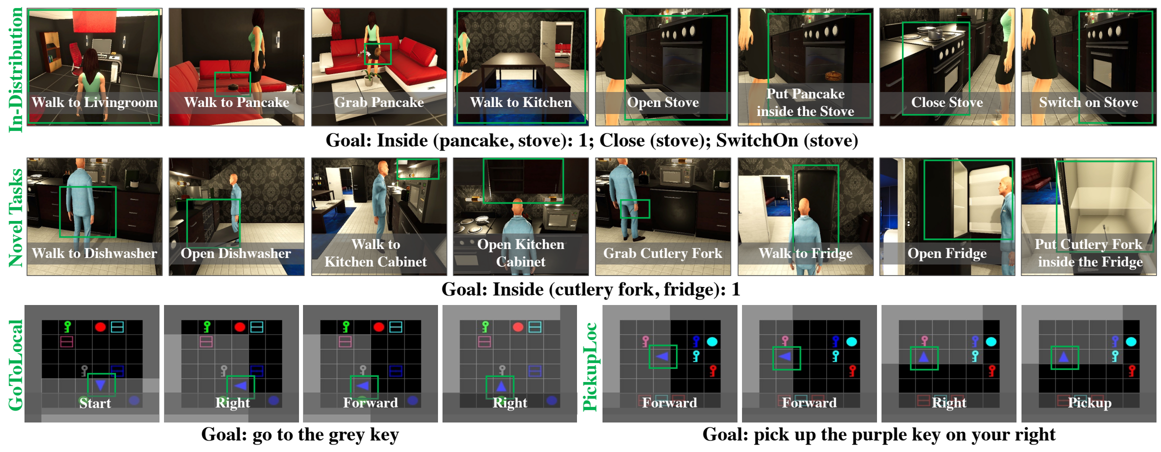


Figure 5-4: **Qualitative results of our model on VirtualHome and BabyAI.** We only show a sub-trajectory in each example to save space. The interacted objects are labeled by green bounding boxes.

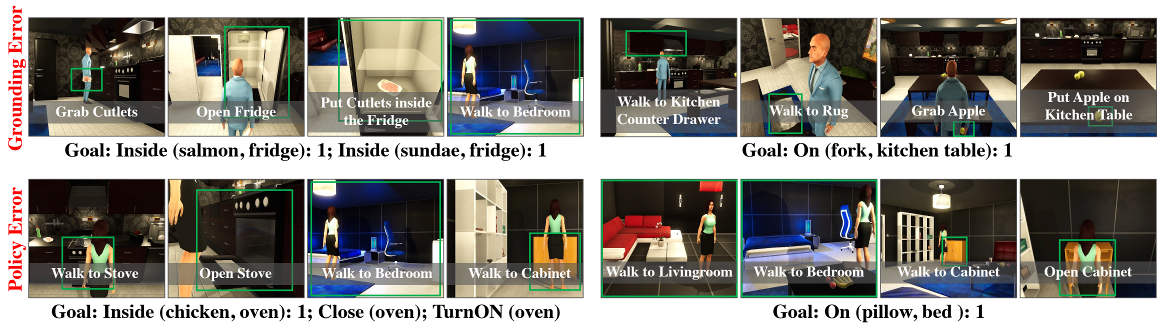

Grounding Error
Grab Cutlets
Open Fridge
Put Cutlets inside the Fridge
Walk to Bedroom
Goal: Inside (salmon, fridge): 1; Inside (sundae, fridge): 1
Walk to Kitchen Counter Drawer
Walk to Rug
Grab Apple
Put Apple on Kitchen Table
Goal: On (fork, kitchen table): 1
Policy Error
Walk to Stove
Open Stove
Walk to Bedroom
Walk to Cabinet
Goal: Inside (chicken, oven): 1; Close (oven); TurnON (oven)
Walk to Livingroom
Walk to Bedroom
Walk to Cabinet
Open Cabinet
Goal: On (pillow, bed): 1

Figure 5-5: **Failure cases.** We show failure cases caused by the grounding error and policy error. The interacted objects are labeled by green bounding boxes.

Failure case analysis. In Figure 5-5, we show some failure cases of the proposed method. We observed two main types of failure cases: grounding error and policy error. For failures caused by the grounding error, the agent interacts with a wrong object that is not related to the given goal, *e.g.*, the agent puts *cutlets* instead of the *salmon* inside the fridge. For failures caused by the policy error, the agent cannot find the target objects or does not interact with them. The proposed method that converts policy inputs into sequential encodings and feeds them to the general LM framework can accomplish decision-making tasks efficiently, however, there are still challenging tasks that the policy fails to accomplish. Larger LMs, *e.g.*, GPT-3 [13], may improve the success rate of those challenging tasks.

5.9 Conclusion

In this chapter, we introduced a general approach to sequential decision-making that converts goals, histories, and observations into sequences and processes them using a policy initialized with a pre-trained LM. We integrated an active data-gathering procedure into the proposed method to enable policy learning without using expert data. Our analysis showed that input representation and favorable weight initialization both contribute to the combinatorial generalization while the input encoding scheme has little influence. Our results demonstrate that the proposed method enables effective combinatorial generalization across different environments, and highlight the promise of LM pre-training for more general decision-making problems.

The success of using language models for decision-making tasks involving new combinations shows that we can transfer the compositional skills from large pre-trained models. The data we needed to fine-tune these models is much smaller compared to what was used to train them in the first place. By shifting this compositional knowledge from pre-trained models, we can achieve compositionality more efficiently.

Chapter 6

Epilogue

We are standing at the threshold of a remarkable era in the advancement of Artificial Intelligence (AI). This era is marked by the advent of colossal AI models such as GPT-4 [109] and DALL-E 2 [125], which have not only astonished the scientific community but also permeated daily life. These systems, driven by deep neural networks, epitomize the transformative nature of AI technology, highlighting the intriguing possibilities of what the future may hold.

As we navigate this age of exponential growth in AI research and development, we are met with vital questions that deserve our reflection and scrutiny. In the relentless pursuit of larger and more complex models, one cannot help but notice a dichotomy between the learning paradigms in artificial systems and the organic cognitive processes observed in human beings. Human learning is characterized by its remarkable ability to swiftly assimilate and synthesize new information, leveraging existing knowledge and often operating with sparse input. In stark contrast, contemporary large-scale models require vast amounts of training data and typically require extensive fine-tuning to adapt to new tasks.

In our research, we argue that large-scale models alone cannot encapsulate the full potential of artificial intelligence. To create more effective, adaptive, and resilient AI in real-world applications, we need systems that integrate new features without undermining existing functions. Compositionality emerges as a potent framework to navigate this challenge.

Our approach emphasizes the blending of the strengths of pre-trained models with compositional structures. We design logical compositional operators in the probability domain to construct composition structures. Additionally, we obtain efficient pre-trained models by transferring compositionality from other well-developed models without the need for large datasets.

We advocate for the construction of decentralized and composable AI systems, recognizing the significance of both pre-trained models and compositional structures. Our work represents a stride toward the construction of more structured, controllable, and interpretable representations of the world. It is a step that not only aligns more closely with the nature of human cognition but also paves the way for the development of more robust and effective AI systems.

While our current system boasts significant advancements, it is not without its limitations, as detailed in each chapter of this thesis. However, we firmly believe in compositionality as an essential facet of intelligence. Our conviction is that future endeavors will further equip AI systems with a more robust and coherent compositionality. This will enable the model to engage in continual learning in our ever-changing and dynamic world.

Appendix A

Supplementary: Compositional Visual Generation with Energy-Based Models

A.1 Partition Function

We estimate the magnitude of the partition function of an EBM by evaluating the energy it assigns to all data points it is trained on. The histogram of energies is shown in Figure A-1. Due to a combination of L2 normalization and spectral normalization, the EBMs we evaluated have different architectures but similar histograms.

In Figure A-1, The energy histograms are the CelebA model trained on the smiling concept, the CelebA model trained on the attractive concept, and the CIFAR-10 model trained on CIFAR-10 objects [30]. We find that all energy histograms are similar, exhibiting minimum and maximum energies between -0.01 and 0.01. This is true even for the CIFAR-10 model, which uses a significantly different dataset.

In scenarios where partition functions are different, our defined disjunction operator does not fail drastically. If two unnormalized probability distributions have partition function values of w_1 and w_2, then models will be sampled with proportion $\frac{w_1}{w_1+w_2}$ and $\frac{w_2}{w_1+w_2}$, which is not a dramatic failure in disjunction.

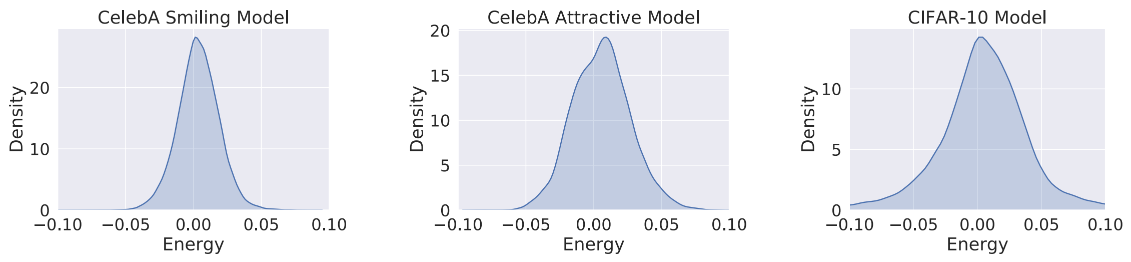


Figure A-1: **Energy histograms.** Models trained on CelebA smiling (left), CelebA attractive (middle), and a pretrained CIFAR-10 model from [30] (right). The EBMs we evaluate have different architectures but similar histograms.

A.2 Composing Visual Concepts

In Section 2.4, we introduced composing visual concepts using energy-based models. In this part, we first show additional experiments in Appendix A.2.1. We then provide the model architecture details (Appendix A.2.2) of experiments used in Section 2.4.

A.2.1 Disjoint compositionality results

We further evaluate compositionality when conditioned factors are mutually disjoint from each other. In particular, we train EBM models on frog and truck image classes in CIFAR-10. In Figure A-2, we illustrate resulting generations. We find that when conditioning on both classes, our resultant generations exhibit characteristics of each individual class.

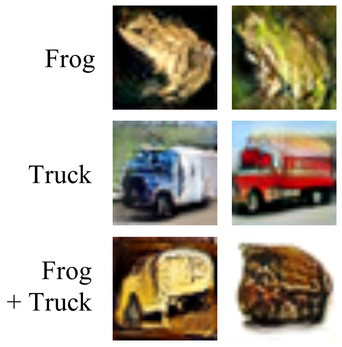


Figure A-2: **Hybrid combinations of frog and truck EBMs.**

A.2.2 Model architecture details

We list the EBM architectures used for the Mujoco Scenes dataset in Figure A-3a and for the Celeba dataset in Figure A-3b. The baseline model used for comparisons in Section 2.4.4 is shown in Figure A-3c.

3x3 conv2d, 64
ResBlock down 64
ResBlock down 128
ResBlock down 128
ResBlock down 256
Global Mean Pooling
Dense $\rightarrow$ 1

(a) Model architecture of EBM trained on the Mujoco Scenes dataset.

3x3 conv2d, 64
ResBlock down 64
ResBlock down 128
ResBlock down 256
ResBlock down 512
ResBlock down 1024
ResBlock 1024
Global Sum Pooling
dense $\rightarrow$ 1

(b) Model architecture of EBM trained on the CelebA dataset.

Dense $\rightarrow$ 4096
Reshape $\rightarrow$ 256x4x4 ResBlock up 256
ResBlock up 128
ResBlock up 64
ResBlock up 64
3x3 conv2d, 3

(c) Model architecture of the baseline model used in Section 2.4.4.

Figure A-3: **Model architecture of methods trained on different datasets.**

A.3 Composing Visual Relations

In Section 2.5, we introduced composing visual relations using energy-based models. In this part, we first present additional experiments from Appendices A.3.1 to A.3.3. We then show the model architecture details of various approaches and their implementation details in Appendix A.3.4 and Appendix A.3.5, respectively.

A.3.1 Comparison with more baseline approaches

We compare our method with more baselines on image generation in Table A.1.

StyleGAN2. We used the unconditional StyleGAN2 [65] as one of the baselines. We train the unconditional StyleGAN2 and the ResNet-18 classifier separately on each dataset. For training, we use the default setting provided by [65]. To generate an image with respect to a particular relation, we optimize the underlying latent code to minimize the loss from the classifier.

StyleGAN2 (CLIP). StyleGAN2 (CLIP) is the same as StyleGAN2 except that StyleGAN2 (CLIP) uses the text encoder of the CLIP model [119] to encode relational

Table A.1: **Accuracy of object relations in the generated images.** We compare our method with baselines on the three test sets, *i.e. 1R*, *2R*, and *3R*, of Relational CLEVR and iGibson, respectively.

Dataset	**Model**	**Image Generation (%)**		
		1R Acc	2R Acc	3R Acc
Relational CLEVR	StyleGAN2	10.68	2.46	0.54
	StyleGAN2 (CLIP)	65.98	9.56	1.78
	StyleGAN2 (CLIP) (Multi-Relations)	66.62	9.60	1.68
	Scene Graph GAN	83.72	14.18	4.48
	EBM (CLIP) (Full Sentence)	4.75	0.24	0.00
	Ours (CLIP)	94.79	48.42	18.00
	Ours (Learned Embed)	**97.79**	**69.55**	**37.60**
iGibson	StyleGAN2	12.46	2.24	0.60
	StyleGAN2 (CLIP)	49.20	17.06	5.10
	StyleGAN2 (CLIP) (Multi-Relations)	36.94	13.42	6.86
	Scene Graph GAN	54.64	0.02	0.00
	EBM (CLIP) (Full Sentence)	34.25	8.05	3.47
	Ours (CLIP)	74.02	43.04	**19.59**
	Ours (Learned Embed)	**78.27**	**45.03**	19.39

scene descriptions. We follow the same configuration as StyleGAN2 to train StyleGAN2 (CLIP).

StyleGAN2 (CLIP) (Multi-Relations). StyleGAN2 (CLIP) (Multi-Relations) has the same model architecture as StyleGAN2 (CLIP) but is trained with more scene relations. In StyleGAN2 (CLIP), we only use a single scene relation during training, while StyleGAN2 (CLIP) (Multi-Relations) uses $1 \sim 3$ scene relations.

Scene Graph GAN. We apply the models from [62] and utilize the extracted scene graphs as input to train a conditional StyleGAN2. As there are no object bounding boxes available in our setting, we set the input bounding box to be the whole image frame, and our input scene graphs only consist of two objects and their relation.

EBM (CLIP) (Full Sentence). In this setting, we use the text encoder of CLIP to encode every word in the relational scene descriptions.

As shown in Table A.1, our approach achieves the highest accuracy among all the methods. Directly utilizing CLIP to encode a relational scene description such as "a large blue rubber cube to the left of a small red metal cube" to train an EBM

Table A.2: **Comparison of different methods on Relational CLEVR.** The accuracy of graph-based relational similarity proposed by [33] is reported.

Model	**Relational Similarity (%)**		
	1R Acc ↑	2R Acc ↑	3R Acc ↑
StyleGAN2	22.37	19.75	17.13
StyleGAN2 (CLIP)	37.50	28.62	28.75
Ours (Learned Emb)	**50.77**	**36.87**	**42.50**

“EBM (CLIP) (Full Sentence)” performs much worse than the proposed method “Ours (CLIP)” and “Ours (Learned Embed)”.

A.3.2 Additional evaluation metric

We evaluate different methods based on their binary classification accuracy. In this part, we provide an additional metric to evaluate the image generation results. We investigate the performance of utilizing the graph-based relational similarity metric proposed by [33]. A graph-based relational similarity score is used to test the correct placement of objects, without requiring the model to draw the objects exactly in the same locations as the ground truth. Such a metric can construct scene graphs for both the generated and ground truth images without telling the model to draw objects precisely at the exact locations. However, it heavily relies on the pre-trained object detector and localizer. The pre-trained object detector or localizer could generate false predictions on both real and generated images, especially when the generated images are out of the training distribution.

The evaluation metric used in [33] focuses more on local matching while our binary classification focuses on global matching. We report the results of two baselines and our approach using the new evaluation metric. The image generation results on the Relational CLEVR dataset are shown in Table A.2. The conclusion obtained by using this new metric is coherent with using our binary classification metric: our proposed method outperforms the baselines.

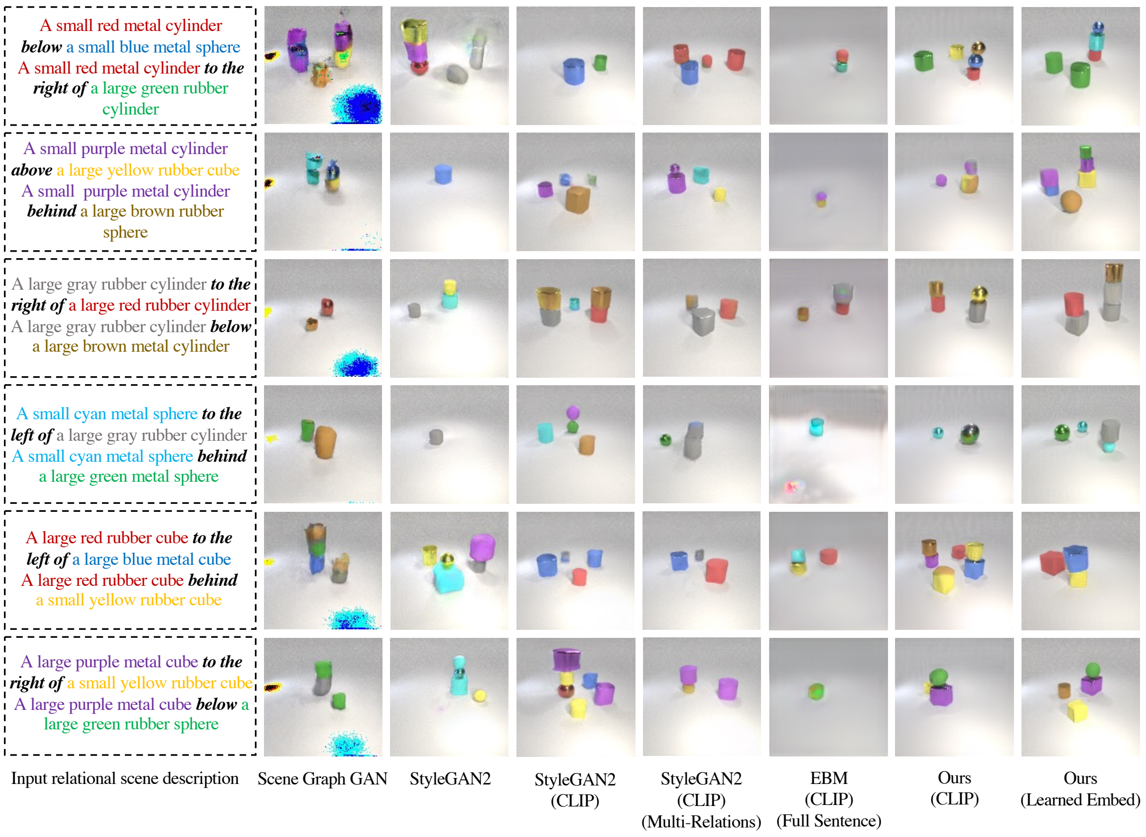

A small red metal cylinder ***below*** a small blue metal sphere
A small red metal cylinder ***to the right of*** a large green rubber cylinder
A small purple metal cylinder ***above*** a large yellow rubber cube
A small purple metal cylinder ***behind*** a large brown rubber sphere
A large gray rubber cylinder ***to the right of*** a large red rubber cylinder
A large gray rubber cylinder ***below*** a large brown metal cylinder
A small cyan metal sphere ***to the left of*** a large gray rubber cylinder
A small cyan metal sphere ***behind*** a large green metal sphere
A large red rubber cube ***to the left of*** a large blue metal cube
A large red rubber cube ***behind*** a small yellow rubber cube
A large purple metal cube ***to the right of*** a small yellow rubber cube
A large purple metal cube ***below*** a large green rubber sphere
Input relational scene description
Scene Graph GAN
StyleGAN2
StyleGAN2 (CLIP)
StyleGAN2 (CLIP) (Multi-Relations)
EBM (CLIP) (Full Sentence)
Ours (CLIP)
Ours (Learned Embed)

Figure A-4: **Image generation results on the Relational CLEVR dataset.** Images are generated based on 2 relational descriptions. Note that the models are trained on a single relational description, and the two composed scene relations are outside the training distribution. Our approaches "Ours (CLIP)" and "Ours (Learned Embed)" are able to generate images accurately based on the input scene descriptions.

A.3.3 Additional qualitative results

Image generation results on Relational CLEVR and iGibson. We show more qualitative results of image generation in Fig. A-4 and Fig. A-5. Our approach can generate images with correct relations, and can even generalize to relational scene descriptions that are out of the training distribution.

Image generation results on real-world datasets. In terms of image generation on real scenes, we train and evaluate our model on two real-world datasets, the Blocks dataset [81] and the Visual Genome dataset [76].

The Blocks dataset is from [81]. We train our model using two types of object

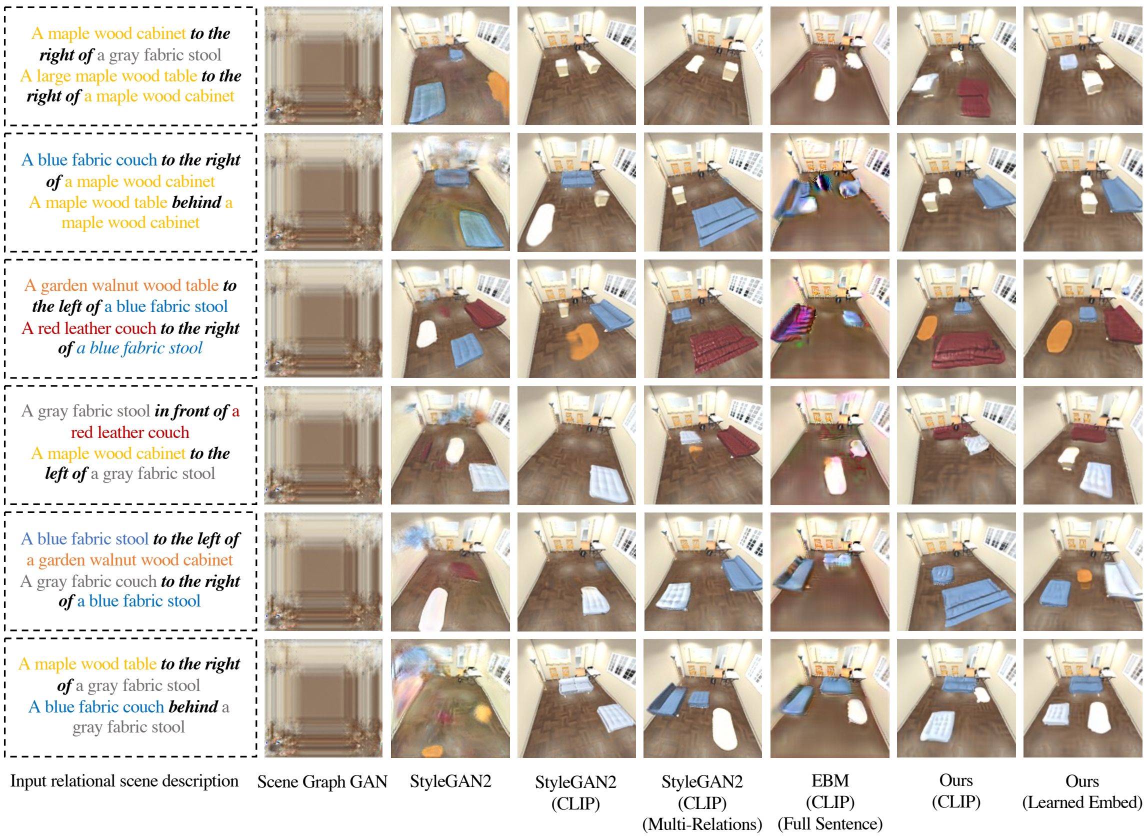

A maple wood cabinet *to the right of* a gray fabric stool
A large maple wood table *to the right of* a maple wood cabinet
A blue fabric couch *to the right of* a maple wood cabinet
A maple wood table *behind* a maple wood cabinet
A garden walnut wood table *to the left of* a blue fabric stool
A red leather couch *to the right of* a blue fabric stool
A gray fabric stool *in front of* a red leather couch
A maple wood cabinet *to the left of* a gray fabric stool
A blue fabric stool *to the left of* a garden walnut wood cabinet
A gray fabric couch *to the right of* a blue fabric stool
A maple wood table *to the right of* a gray fabric stool
A blue fabric couch *behind* a gray fabric stool
Input relational scene description
Scene Graph GAN
StyleGAN2
StyleGAN2 (CLIP)
StyleGAN2 (CLIP) (Multi-Relations)
EBM (CLIP) (Full Sentence)
Ours (CLIP)
Ours (Learned Embed)

Figure A-5: **Image generation results on the iGibson dataset.** Images are generated based on 2 relational descriptions. Note that the models are trained on a single relational description, and the two composed scene relations are outside the training distribution. Our approaches "Ours (CLIP)" and "Ours (Learned Embed)" are able to generate images accurately based on the input scene descriptions.

relations, *e.g.*, "above" and "below". We show the images generated conditioned on two relational descriptions and three relational descriptions in Fig. A-6.

For the Visual Genome dataset [76], we train our models on a subset that consists of common objects and relations for computational efficiency. As shown in Fig. A-7, we find that the CLIP text encoder performs better, as it has seen large-scale image-text pairs that cover a wide range of relations, attributes, and objects.

Our approach is able to generate images (objects and their relations) matching the given language descriptions on the real-world Blocks dataset and the Visual Genome dataset. The quality of generated images on the Blocks dataset is great. However, the quality of results on the Visual Genome dataset is a bit worse. We believe the generation quality could be further improved given more powerful text encoders.

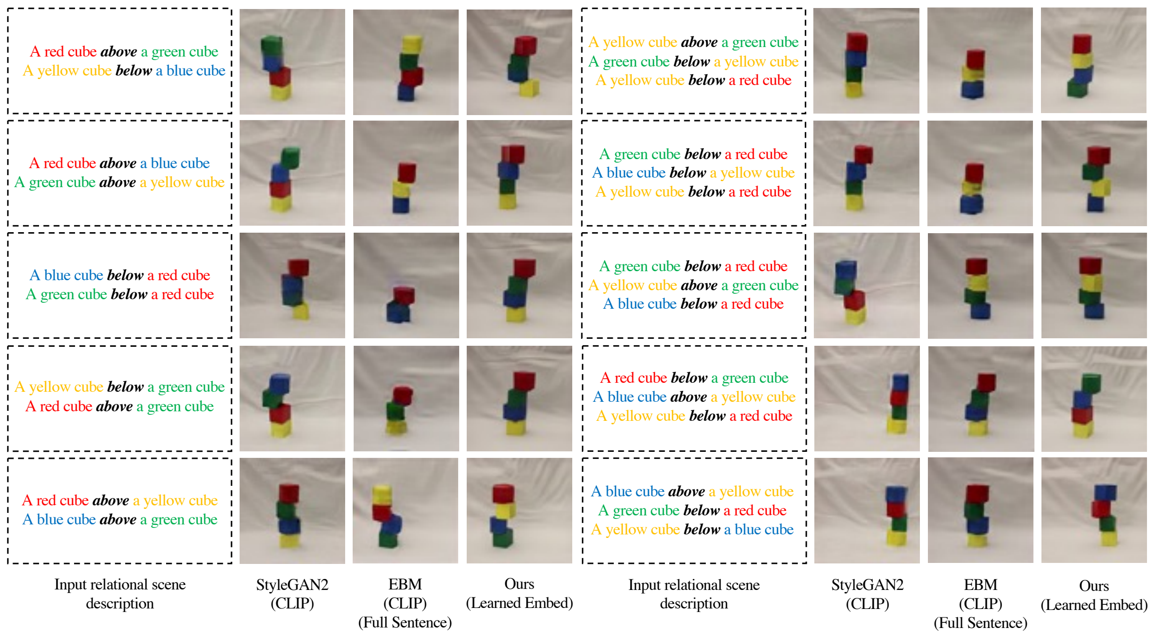

A red cube *above* a green cube
A yellow cube *below* a blue cube
A red cube *above* a blue cube
A green cube *above* a yellow cube
A blue cube *below* a red cube
A green cube *below* a red cube
A yellow cube *below* a green cube
A red cube *above* a green cube
A red cube *above* a yellow cube
A blue cube *above* a green cube
A yellow cube *above* a green cube
A green cube *below* a yellow cube
A yellow cube *below* a red cube
A green cube *below* a red cube
A blue cube *below* a yellow cube
A yellow cube *below* a red cube
A green cube *below* a red cube
A yellow cube *above* a green cube
A blue cube *below* a red cube
A red cube *below* a green cube
A blue cube *above* a yellow cube
A yellow cube *below* a red cube
A blue cube *above* a yellow cube
A green cube *below* a red cube
A yellow cube *below* a blue cube
Input relational scene description
StyleGAN2 (CLIP)
EBM (CLIP) (Full Sentence)
Ours (Learned Embed)
Input relational scene description
StyleGAN2 (CLIP)
EBM (CLIP) (Full Sentence)
Ours (Learned Embed)

Figure A-6: **Image generation results on the Block dataset.** Images are generated based on 2 or 3 relational descriptions. Note that the models are trained on a single relational description, and the composed scene relations (2 and 3 relational descriptions) are outside the training distribution. Our approach "Ours (Learned Embed)" is able to generate images accurately based on the input scene descriptions.

A.3.4 Model architecture details

We follow the implementation of EBMs from [28] in our experiments. Similar to [28], we use the multi-scale model architecture to compute energies as shown in Table A.3. Each model generates an energy value and the final energy $E_\theta(\boldsymbol{x})$ is the sum of energies from all the models listed in Table A.3. Given relational scene descriptions, we generate or edit images based on the final energy.

A.3.5 Implementation details

StyleGAN2. It takes 2 days to train the StyleGAN2 model and 2 hours to train the classifier using a single Tesla 32GB GPU on each dataset. We use the Adam optimizer [72] with $\beta_1 = 0$, $\beta_2 = 0.99$, and $\epsilon = 10^{-8}$ to train the model.

StyleGAN2 (CLIP). For StyleGAN2 (CLIP) and StyleGAN2 (CLIP) (Multi-Relations), it takes around 2 days to train each of them on each dataset using a

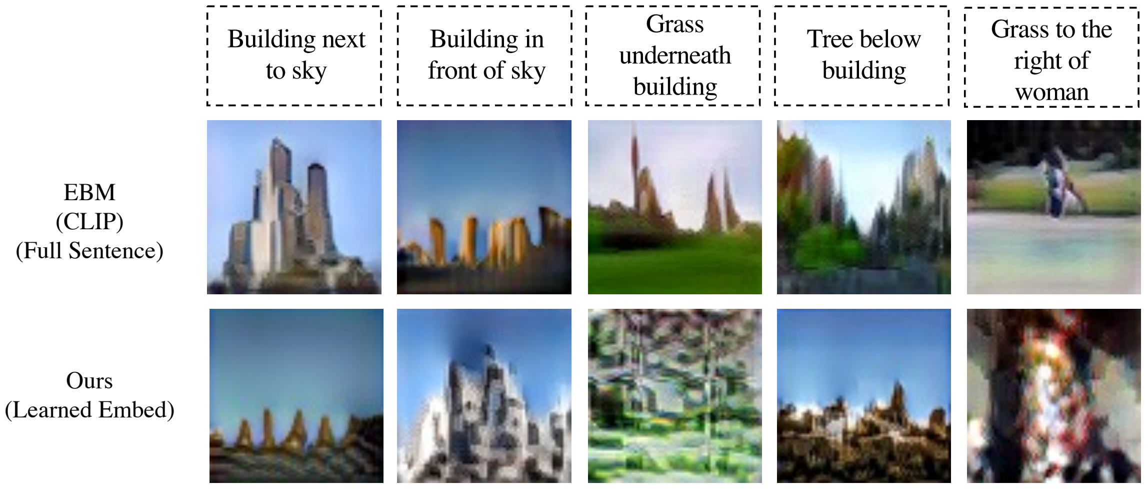


Figure A-7: **Image generation results on the Visual Genome dataset.** "EBM (CLIP) (Full Sentence)" performs better than "Ours (Learned Emb)" in generating more complex natural images because the pretrained CLIP text encoder has seen large-scale image-text pairs that cover a wide range of objects and their relations.

single Tesla 32GB GPU. We use the Adam optimizer [72] with $\beta_1 = 0$, $\beta_2 = 0.99$, and $\epsilon = 10^{-8}$ to train them.

Scene Graph GAN. We train the model on each dataset with the default training configuration provided in the codebase from [62] for 2 days using a single Tesla 32GB GPU. We use the Adam optimizer [72] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-4}$ to train the model.

EBMs (*i.e.*, Ours (CLIP), Ours (Learned Embed), EBM (CLIP) (Full Sentence)). In our experiments, we use the same setting to train models using EBMs, *i.e.*, Ours (CLIP), Ours (Learned Embed), and EBM (CLIP) (Full Sentence), for fair comparisons. We use the Adam optimizer [72] with learning rates of 10^{-4} and 2×10^{-4} on the Relational CLEVR and iGibson datasets, respectively. For MCMC sampling, we use a step size of 300 on the Relational CLEVR dataset, 750 on the iGibson dataset, and 300 on the Blocks dataset. On each dataset, the model is trained for 3 days on a single Tesla 32GB GPU.

To generate images at test time, we initialize an image sample from random noise. We then iteratively apply data augmentation on the image sample, followed by 20

Table A.3: **Model architectures.** We use the multi-scale model architecture to compute energies.

3x3 Conv2d 128
CondResBlock 128
CondResBlock Down 128
CondResBlock 128
CondResBlock Down 256
Self-Attention 256
CondResBlock 256
CondResBlock Down 256
CondResBlock 512
CondResBlock Down 512
Global Mean Pooling
Dense $\rightarrow$ 1

3x3 Conv2d 128
CondResBlock 128
CondResBlock Down 128
CondResBlock 128
CondResBlock Down 128
Self-Attention 256
CondResBlock 256
CondResBlock Down 256
Global Mean Pooling
Dense $\rightarrow$ 1

3x3 Conv2d 128
CondResBlock 128
CondResBlock Down 128
Self-Attention 128
CondResBlock 128
CondResBlock Down 128
Global Mean Pooling
Dense $\rightarrow$ 1

steps of Langevin sampling. To generate the final image, we ran 80 additional steps of Langevin sampling on the image sample.

To edit images at test time, we run 80 steps of Langevin sampling steps. The step size of Langevin sampling is inversely proportional to the number of scene relations, *i.e.*, more scene relations lead to a smaller Langevin sampling step size.

Appendix B

Supplementary: Compositional Visual Generation with Diffusion Models

We first demonstrate additional results in Appendix B.1. We then show the details of training classifiers in Appendix B.2. In Appendix B.3 and Appendix B.4, we show more details of our approach and baselines, respectively. We finally provide the implementation details in Appendix B.5.

B.1 Additional Results

We provide more qualitative results of the proposed method on composing concepts using the conjunction operator. Fig. B-1, Fig. B-2, Fig. B-3, and Fig. B-4 show more results of composing language descriptions. Fig. B-5 shows additional results on composing objects on the CLEVR dataset. Our approach can reliably generate images conditioned on multiple concepts, even for combinations that are outside the training distribution.

We further show results of composing facial attributes on the FFHQ dataset in Fig. B-6. Our model is trained to generate images conditioned on a single human facial attribute, but it can compose multiple attributes during inference without further training by using the conjunction and negation compositional operators. As shown in the fifth row of Fig. B-6, our model can compose “NOT Male” and “NOT (No Glasses)” and generate images with females wearing glasses. The proposed compositional operators allow our model to compose facial attributes recursively.

B.2 Details of Binary Classifiers

We provide more details of the binary classifiers in this section.

CLEVR. The CLEVR dataset consists of $30,000$ image-label pairs. We split the dataset into training and validation subsets. There are $24,000$ data pairs used for training and $6,000$ data pairs used for validation. We train a binary classifier to evaluate whether there is an object appearing at a particular position of an image. The classifier achieves an accuracy of 99.05% on the validation set, which is used to evaluate the quality of generated images.

Relational CLEVR. The Relational CLEVR [87] dataset contains $50,000$ images at 128×128 resolution. We split the dataset into $40,000$ training data and $10,000$ validation data. Then we train a binary classifier to evaluate whether an image contains an object relational description. The trained classifier achieves an accuracy of 99.80% on the validation set.

FFHQ. We use $30,000$ image-label pairs from CelebA-HQ [64] to train a classifier to evaluate the generated images. We split the dataset into the training ($24,000$ data pairs) and validation ($6,000$ data pairs) subsets. We select three attributes (*i.e.*, "smiling", "glasses", and "gender") to evaluate the compositional ability of our approach and baselines. We thus train three binary classifiers to evaluate the "smiling", "glasses", and "gender" concepts, respectively. Our classifiers achieve 95.01%, 99.20% and 97.49% accuracy on the validation sets of "smiling", "glasses", and "gender", respectively.

B.3 Details of Our Approach

Training. Our approach is implemented based on the code from [105, 103]. Ho *et al*. [52] introduce a technique to train the conditional and unconditional diffusion models at the same time by masking some labels as nulls. We use the same approach to train diffusion models. For each data point, its label has a 10% chance of being replaced by a null label which is used to estimate the unconditional score.

Inference. To generate FFHQ images, we first generate images at 64×64 resolution and then upsample the images to 256×256 using a sampler provided by [103]. For CLEVR images, we generate images at 128×128 resolution directly.

Label encoding. On the FFHQ dataset, we use three human facial attributes, *i.e.*, "smile", "glasses", and "gender". For the "smile" and "glasses" attributes, label 1 indicates an image containing the attribute; otherwise, the label is 0. For the "gender" attribute, label 0 indicates "male", while label 1 represents "female". We use the embedding layer $nn.Embedding(7, d)$ to encode the attribute labels. The first six dimensions represent the attribute labels and the last dimension indicates the null class. The labels are encoded as a d-dimension feature vector, which is then fused with the time embedding to estimate the score ϵ_θ.

On the CLEVR dataset, we encode the (x, y) coordinates using a linear layer $nn.Linear(2, d)$, where d is the dimension of the output feature. The coordinate embedding is then fused with the time embedding to estimate the score ϵ_θ.

B.4 Details of Baselines

StyleGAN2-ADA. On each dataset, we train a conditional StyleGAN2-ADA model using the "stylegan" configuration provided by [66] without using augmentations.

StyleGAN2. We use the pre-trained StyleGAN2 model [68] to evaluate its performance on facial image generation. As there is no pre-trained model for object generation, we use the same code to train a model on the CLEVR dataset for image generation conditioned on object positions. We use the "config-f" setting provided by [68]. To enable image generation conditioned on multiple concepts, we train a binary classifier on each task. During inference, we optimize the latent code z by decreasing the binary classification loss of the generated image and the given label.

We use the resultant latent code to generate images.

LACE. LACE [106] trains classifiers for image generation using the generated images from StyleGAN2 and labels provided by the neural network. For the CLEVR dataset, we first generate $10,000$ images using the above StyleGAN2 model that was trained on CLEVR. Then we modify the code to train a position annotator using a DenseNet [56] model provided by LACE to label the object positions of generated images. Lastly, we train a classifier conditioned on object coordinates using the code provided by [106]. For FFHQ, we use the off-the-shelf pre-trained model from [106] for comparison.

GLIDE. We use the small GLIDE model released by [103] in our experiments. We develop Composed GLIDE (Ours), a version of GLIDE that utilizes our compositional operators to combine textual descriptions, without further training. We compare it with the original GLIDE, which directly encodes the descriptions as a single long sentence. [103] also releases an upsampler model to upsample the generated images from a resolution of 64×64 to a resolution of 256×256. We use the upsampler model for both the GLIDE and Composed GLIDE (Ours).

Energy-based models (EBMs). We train energy-based models using the codebase from [28], where we encode discrete labels and continuous labels using an embedding layer and a linear layer, respectively. We use the inference code from [25] to compose multiple concepts.

B.5 Implementation Details

Each model is trained on a single Tesla V100 32GB GPU.

StyleGAN2-ADA. Each conditional StyleGAN2-ADA model is trained for two days. We use the Adam optimizer [72] with $\beta_1 = 0$ and $\beta_2 = 0.99$ to train the models.

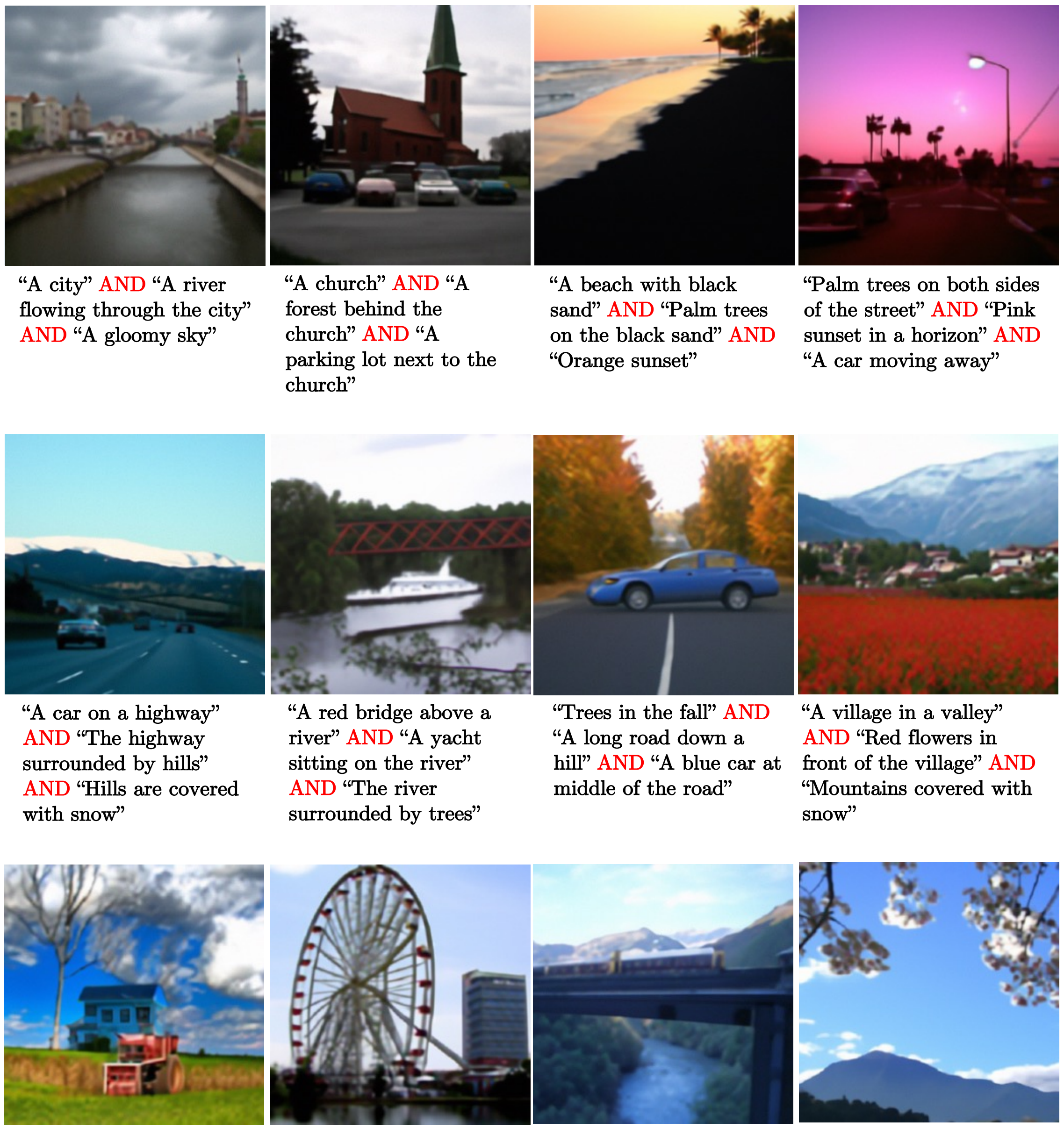
StyleGAN2. We train a StyleGAN2 model for two days on both CLEVR and Relational CLEVR datasets. We use the Adam optimizer [72] with $\beta_1 = 0$ and $\beta_2 = 0.99$ to train the StyleGAN2 models. It takes 2 hours to train a binary classifier.

The classifiers are trained using the Adam optimizer with $\beta_1 = 0$ and $\beta_2 = 0.99$. For the FFHQ dataset, We use the pre-trained model provided by [68].

LACE. LACE uses the pre-trained model provided by [68] on the FFHQ dataset. For both CLEVR and Relational CLEVR datasets, we directly reuse the trained StyleGAN2 model. It takes less than 10 minutes to train the classifier on each dataset.

EBMs. In our experiments, we use the same setting to train models on different datasets. We use the Adam optimizer [72] with a learning rate of 10^{-4}. For MCMC sampling, we use a step size of 300 and 80 iterations. Similarly, the model is trained for two days on each dataset.

Ours. To train diffusion models on CLEVR and FFHQ, we use $1,000$ diffusion steps, and the cosine noise schedule. We use the AdamW optimizer [91] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We train the diffusion models on CLEVR for $750,000$ iterations and FFHQ for $250,000$ iterations.



"A city" AND "A river flowing through the city" AND "A gloomy sky"

"A church" AND "A forest behind the church" AND "A parking lot next to the church"

"A beach with black sand" AND "Palm trees on the black sand" AND "Orange sunset"

"Palm trees on both sides of the street" AND "Pink sunset in a horizon" AND "A car moving away"

"A car on a highway" AND "The highway surrounded by hills" AND "Hills are covered with snow"

"A red bridge above a river" AND "A yacht sitting on the river" AND "The river surrounded by trees"

"Trees in the fall" AND "A long road down a hill" AND "A blue car at middle of the road"

"A village in a valley" AND "Red flowers in front of the village" AND "Mountains covered with snow"

"A blue house" AND "A red tractor on a farm" AND "A cloudy sky"

"A Ferris wheel" AND "A lake next to the Ferris wheel" AND "Buildings next to the lake"

"A train on a bridge" AND "A river under the bridge" AND "Mountains behind the train"

"A cloudy blue sky" AND "A mountain in the horizon" AND "Cherry Blossoms in front of the mountain"

Figure B-1: **Examples of composing language descriptions.** We provide more qualitative results of Composed GLIDE (Ours), a version of GLIDE [103] that utilizes our compositional operators to combine textual descriptions without further training.

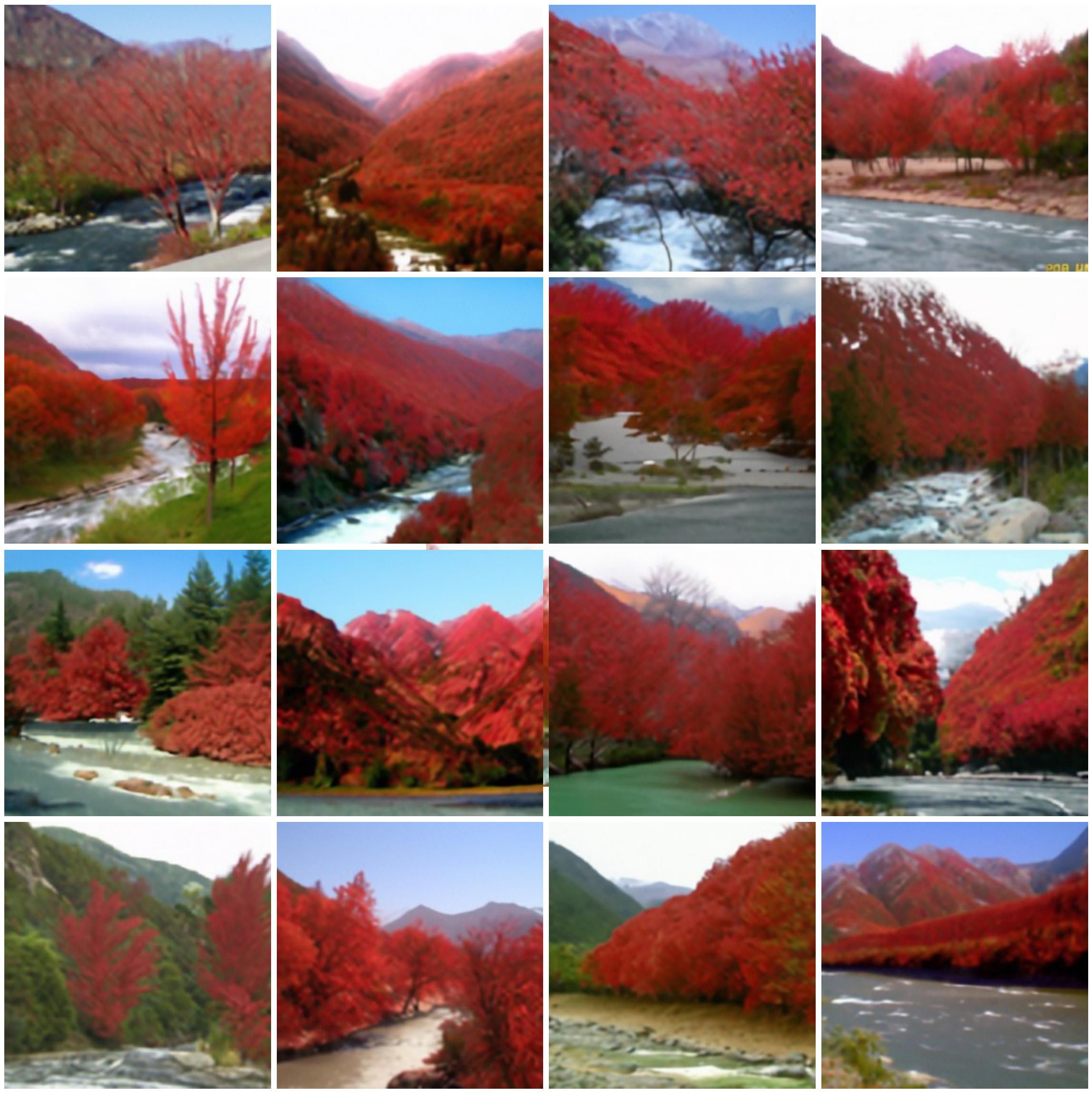

“A river leading into mountains” AND “Red trees on the side”

Figure B-2: **Examples of composing language descriptions**. Images generated by our method, Composed GLIDE (Ours).

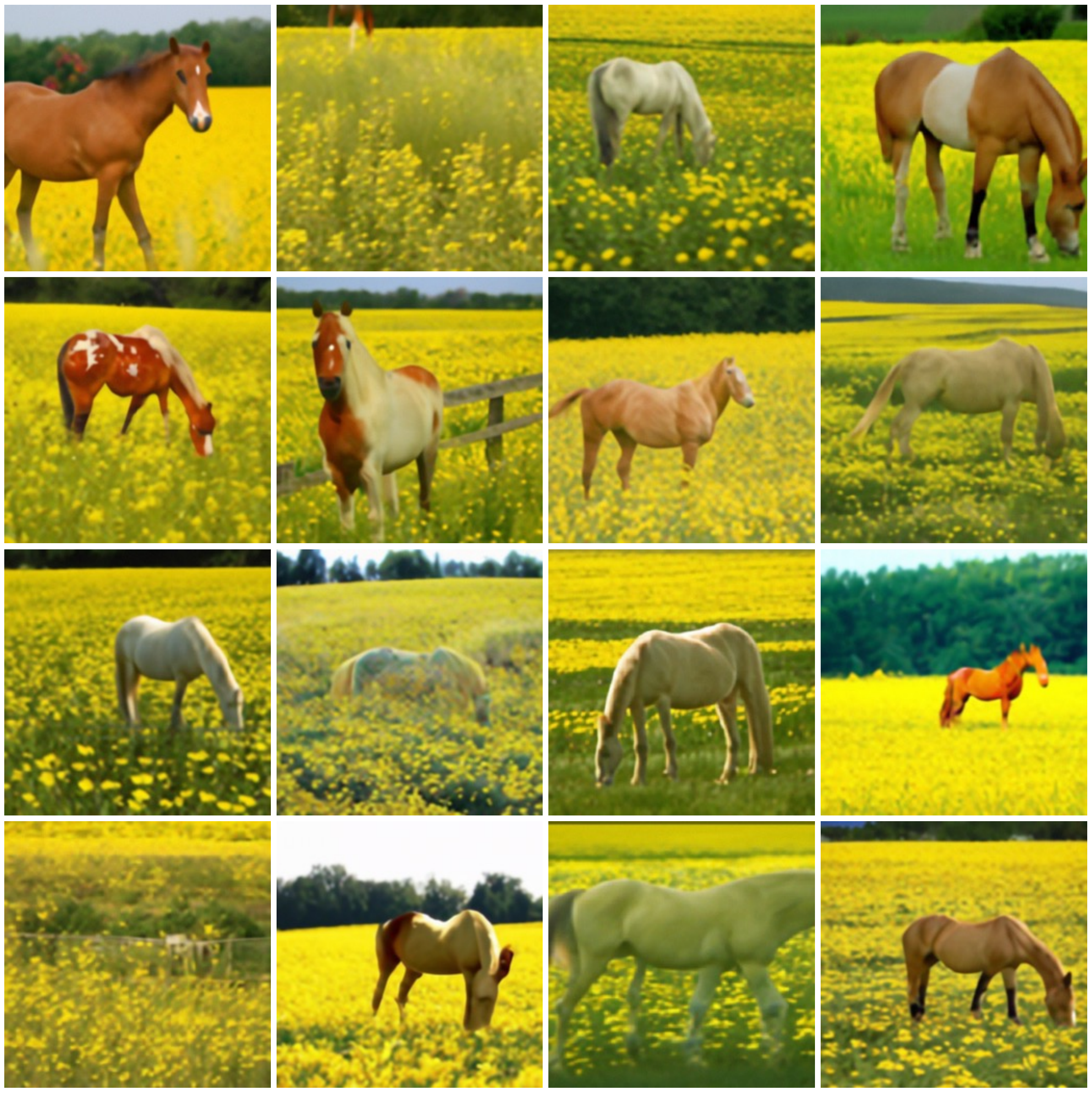


Figure B-3: **Examples of composing language descriptions.** Images generated by our method, Composed GLIDE (Ours).

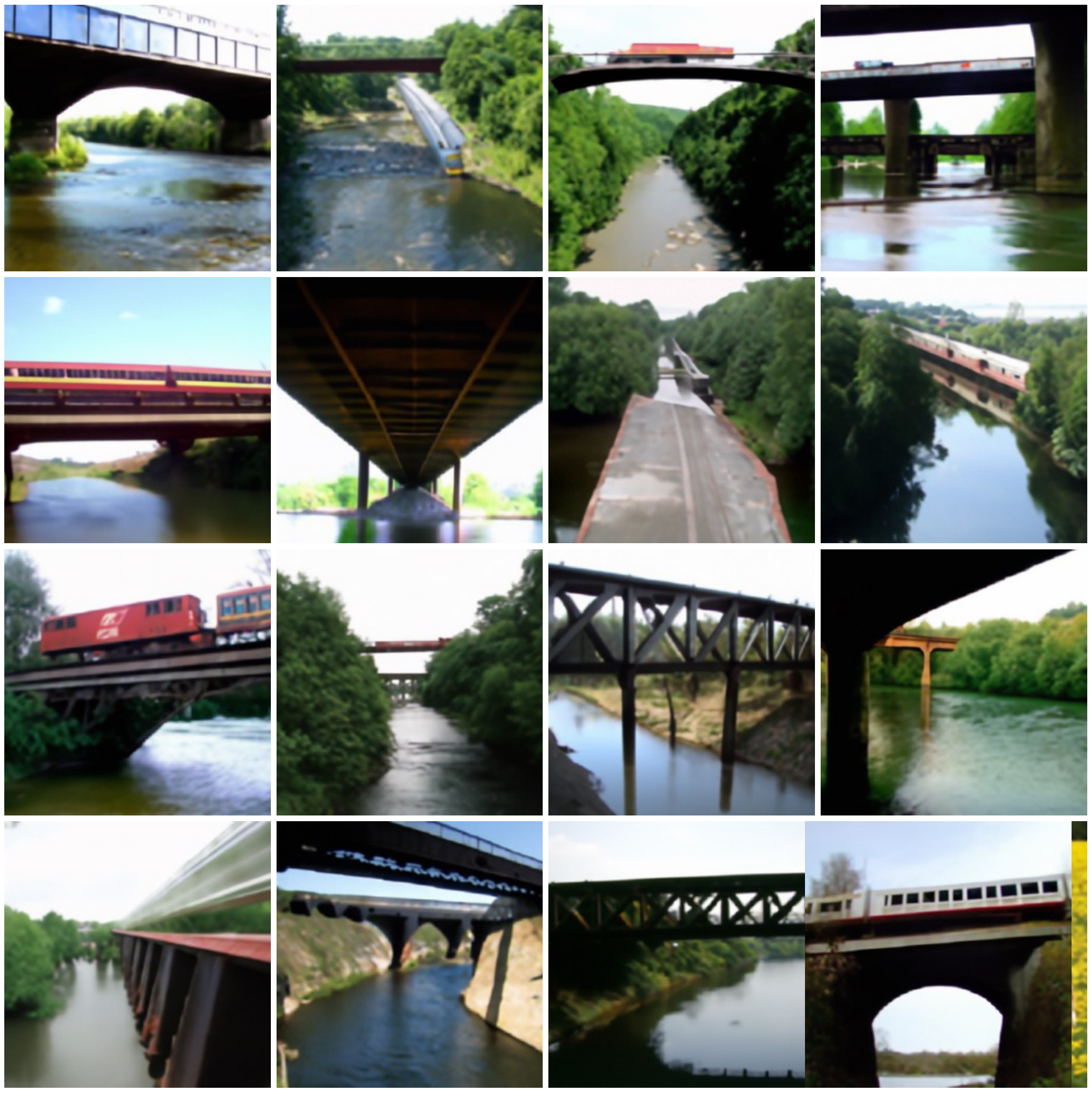
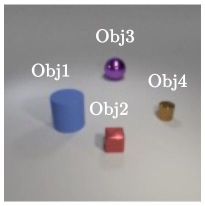
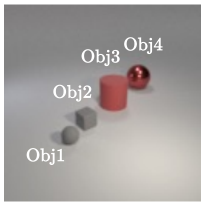

“A train on a bridge” AND “A river under the bridge”

Figure B-4: **Examples of composing language descriptions.** Images generated by our method, Composed GLIDE (Ours).

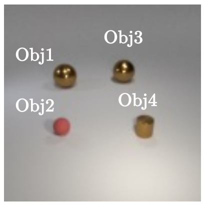
In-distribution (1-5 objects) Compositional Generation on CLEVR



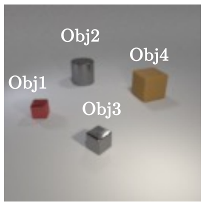
Obj1 (0.29, 0.47) AND
Obj2 (0.55, 0.31) AND
Obj3 (0.57, 0.68) AND
Obj4 (0.82, 0.45)



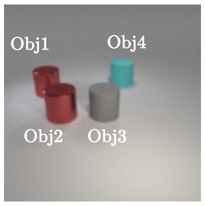
Obj1 (0.3, 0.3) AND
Obj2 (0.4, 0.4) AND
Obj3 (0.55, 0.55) AND
Obj4 (0.7, 0.65)



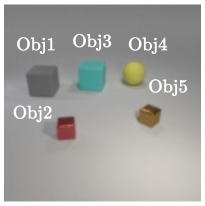
Obj1 (0.31, 0.64) AND
Obj2 (0.22, 0.31) AND
Obj3 (0.61, 0.68) AND
Obj4 (0.74, 0.37)



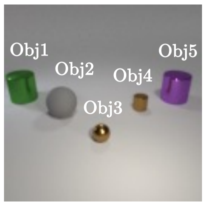
Obj1 (0.16, 0.46) AND
Obj2 (0.38, 0.68) AND
Obj3 (0.47, 0.32) AND
Obj4 (0.73, 0.59)



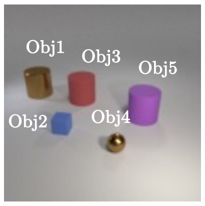
Obj1 (0.2, 0.65) AND
Obj2 (0.3, 0.5) AND
Obj3 (0.5, 0.5) AND
Obj4 (0.6, 0.65)



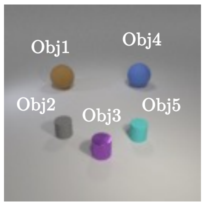
Obj1 (0.24, 0.61) AND
Obj2 (0.3, 0.38) AND
Obj3 (0.45, 0.62) AND
Obj4 (0.65, 0.68) AND
Obj5 (0.74, 0.43)



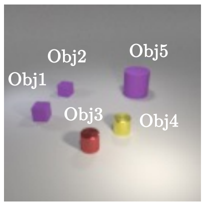
Obj1 (0.1, 0.6) AND
Obj2 (0.3, 0.5) AND
Obj3 (0.5, 0.35) AND
Obj4 (0.7, 0.5) AND
Obj5 (0.9, 0.6)



Obj1 (0.2, 0.66) AND
Obj2 (0.29, 0.39) AND
Obj3 (0.41, 0.58) AND
Obj4 (0.57, 0.29) AND
Obj5 (0.69, 0.5)

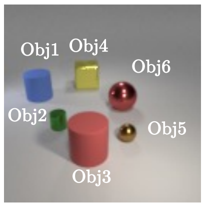


Obj1 (0.3, 0.65) AND
Obj2 (0.3, 0.35) AND
Obj3 (0.5, 0.3) AND
Obj4 (0.7, 0.65) AND
Obj5 (0.7, 0.35)

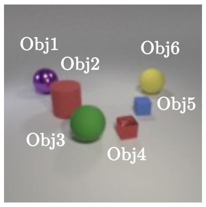


Obj1 (0.15, 0.42) AND
Obj2 (0.3, 0.58) AND
Obj3 (0.41, 0.3) AND
Obj4 (0.59, 0.4) AND
Obj5 (0.64, 0.61)

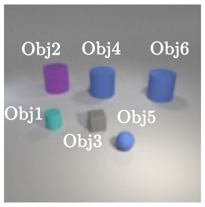
Out-of-distribution (> 5 objects) Compositional Generation on CLEVR



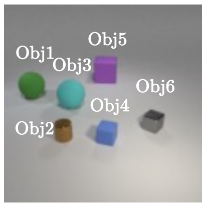
Obj1 (0.18, 0.59) AND
Obj2 (0.21, 0.35) AND
Obj3 (0.43, 0.31) AND
Obj4 (0.42, 0.63) AND
Obj5 (0.63, 0.33) AND
Obj6 (0.61, 0.55)



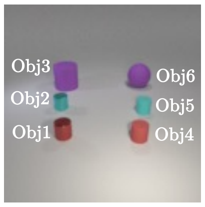
Obj1 (0.2, 0.65) AND
Obj2 (0.3, 0.5) AND
Obj3 (0.4, 0.4) AND
Obj4 (0.6, 0.4) AND
Obj5 (0.7, 0.5) AND
Obj6 (0.8, 0.65)



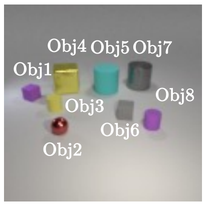
Obj1 (0.24, 0.41) AND
Obj2 (0.28, 0.62) AND
Obj3 (0.48, 0.4) AND
Obj4 (0.51, 0.6) AND
Obj5 (0.64, 0.29) AND
Obj6 (0.77, 0.58)



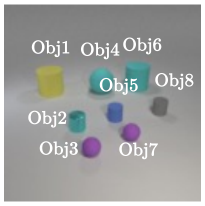
Obj1 (0.13, 0.63) AND
Obj2 (0.24, 0.33) AND
Obj3 (0.33, 0.54) AND
Obj4 (0.52, 0.36) AND
Obj5 (0.51, 0.67) AND
Obj6 (0.77, 0.41)



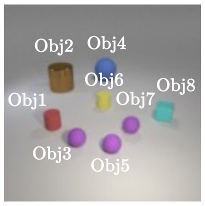
Obj1 (0.3, 0.35) AND
Obj2 (0.3, 0.5) AND
Obj3 (0.3, 0.65) AND
Obj4 (0.7, 0.35) AND
Obj5 (0.7, 0.5) AND
Obj6 (0.7, 0.65)



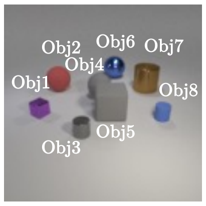
Obj1 (0.12, 0.57) AND
Obj2 (0.27, 0.35) AND
Obj3 (0.27, 0.51) AND
Obj4 (0.32, 0.61) AND
Obj5 (0.5, 0.63) AND
Obj6 (0.62, 0.47) AND
Obj7 (0.67, 0.62) AND
Obj8 (0.77, 0.38)



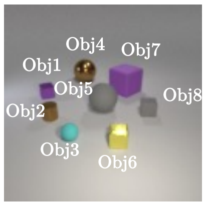
Obj1 (0.22, 0.62) AND
Obj2 (0.35, 0.4) AND
Obj3 (0.44, 0.26) AND
Obj4 (0.47, 0.59) AND
Obj5 (0.57, 0.45) AND
Obj6 (0.7, 0.63) AND
Obj7 (0.7, 0.3) AND
Obj8 (0.8, 0.5)



Obj1 (0.21, 0.37) AND
Obj2 (0.26, 0.65) AND
Obj3 (0.35, 0.27) AND
Obj4 (0.47, 0.59) AND
Obj5 (0.55, 0.27) AND
Obj6 (0.5, 0.5) AND
Obj7 (0.64, 0.4) AND
Obj8 (0.8, 0.47)



Obj1 (0.13, 0.43) AND
Obj2 (0.24, 0.67) AND
Obj3 (0.4, 0.4) AND
Obj4 (0.49, 0.5) AND
Obj5 (0.5, 0.6) AND
Obj6 (0.57, 0.68) AND
Obj7 (0.73, 0.65) AND
Obj8 (0.81, 0.47)



Obj1 (0.22, 0.57) AND
Obj2 (0.25, 0.45) AND
Obj3 (0.33, 0.33) AND
Obj4 (0.4, 0.65) AND
Obj5 (0.48, 0.51) AND
Obj6 (0.56, 0.34) AND
Obj7 (0.61, 0.6) AND
Obj8 (0.71, 0.48)

Figure B-5: **Examples of composing objects.** During inference, our model can generate images that contain multiple objects by composing their probability distributions using the conjunction operator. Note that the training set only contains images with fewer than five objects, but our model can compose more than five objects during inference.

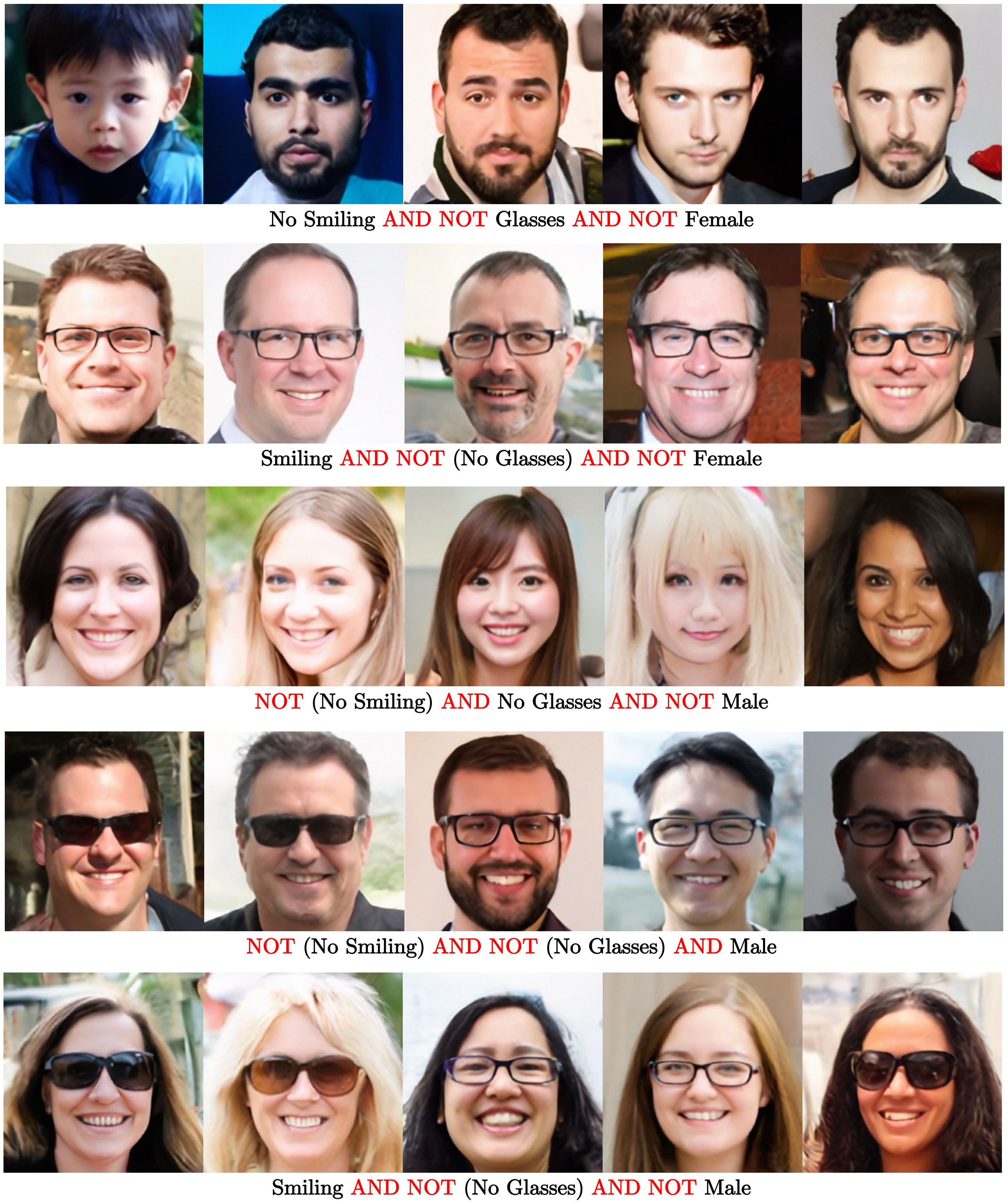

No Smiling AND NOT Glasses AND NOT Female
Smiling AND NOT (No Glasses) AND NOT Female
NOT (No Smiling) AND No Glasses AND NOT Male
NOT (No Smiling) AND NOT (No Glasses) AND Male
Smiling AND NOT (No Glasses) AND NOT Male

Figure B-6: **Examples of composing human facial attributes.** During inference, our model can generate images that contain multiple attributes by composing their probability distributions using the conjunction and negation operators.

Appendix C

Supplementary: Composing Ensembles of Pre-trained Models via Iterative Consensus

We first show experimental details of each task in Appendix C.1. We then show the ethics statement of the Amazon Mechanical Turk experiment for video question answering in Appendix C.2. We finally show additional experimental results in Appendix C.3.

C.1 Experimental Details

In this section, we provide more experimental details of each task. We use TITAN RTX 24GB GPUs for all the experiments.

C.1.1 Image generation

We use the reverse diffusion process of GLIDE, a text-guided diffusion model, as the generator to generate image proposals. At each step of the diffusion process (corresponding to a step of iterative refinement), we use the gradient from an ensemble of scorers to guide and update the generated proposals. We iteratively repeat this procedure until the final step.

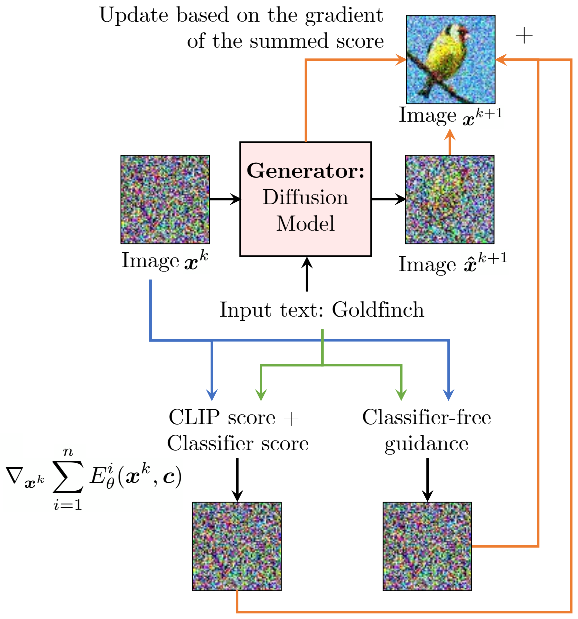

Update based on the gradient of the summed score
+
Image $\boldsymbol{x}^{k+1}$
Generator: Diffusion Model
Image $\boldsymbol{x}^k$
Image $\hat{\boldsymbol{x}}^{k+1}$
Input text: Goldfinch
CLIP score + Classifier score
Classifier-free guidance
$\nabla_{\boldsymbol{x}^k} \sum_{i=1}^{n} E_\theta^i(\boldsymbol{x}^k, \boldsymbol{c})$

Figure C-1: **Overview of image generation.** We use the reverse diffusion process of GLIDE [103], a text-guided diffusion model, as the generator to generate image proposals. At each step of the diffusion process (corresponding to a step of the iterative refinement), we use the gradient from an ensemble of scorers, such as CLIP [119], to guide and update the generated proposals. The image $\boldsymbol{x}^k$ generated at iteration k is first sent to the diffusion model to generate an image proposal $\hat{\boldsymbol{x}}^{k+1}$. The scorers provide feedback to refine the generated result. The scores generated by different scorers are summed, and their gradient with respect to $\boldsymbol{x}^k$ is used to compute the next reverse prediction $\boldsymbol{x}^{k+1}$. Classifier-free guidance [53] can be treated as an implicit classifier that directly provides pixel-wise gradient feedback to the generated image. We iteratively repeat this procedure until the final step. Our framework enables the use of ensembles of different pre-trained models as scorers, significantly improving the zero-shot results by leveraging the strengths of multiple expert models.

As shown in Fig. C-1, the image $\boldsymbol{x}^k$ generated at iteration k is first sent to the diffusion model to generate an image proposal $\hat{\boldsymbol{x}}^{k+1}$. The scorers provide feedback to refine the generated result. The scores generated by different scorers are summed, and their gradient with respect to $\boldsymbol{x}^k$ is used to compute the next reverse prediction $\boldsymbol{x}^{k+1}$. The classifier-free guidance [53] can be treated as an implicit classifier that directly provides pixel-wise gradient feedback to the generated image. Our framework enables the use of ensembles of different pre-trained models as scorers, significantly improving the zero-shot results by leveraging the strengths of multiple expert models.

Our implementation for image generation is modified based on the code of

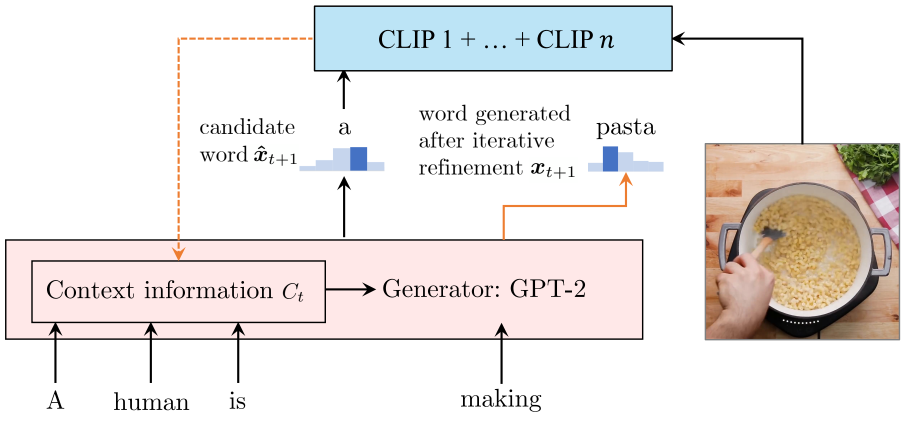

CLIP 1 + … + CLIP n
candidate word $\hat{\boldsymbol{x}}_{t+1}$
a
word generated after iterative refinement $\boldsymbol{x}_{t+1}$
pasta
Context information C_t
Generator: GPT-2
A
human
is
making

Figure C-2: **Overview of video frame captioning for video question answering.** We use GPT-2 as the generator and a set of CLIP models as scorers to generate captions for each video frame. To integrate feedback from scorers to the generator, similar to ZeroCap [152], we define a context cache C_t (a set of embedding functions in GPT-2) that stores the context information generated so far, which is updated iteratively based on the feedback from scorers. To update C_t, we first use G to generate a set of candidate words $\hat{\mathbf{X}}_{t+1} = \{\hat{\boldsymbol{x}}_{t+1}\}$, and then use the feature distance (after softmax) between each sentence (the concatenation of previous words and each new word $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \hat{\boldsymbol{x}}_{t+1}\}$, where $\hat{\boldsymbol{x}}_{t+1} \in \hat{\mathbf{X}}_{t+1}$) and the video frame as the probability of them matching. The CLIP score is the cross-entropy loss $\mathcal{L}_{\text{CLIP}}$ between this new probability distribution and the original distribution of the next word obtained from the generator G (see Equation 4 in [152]). The gradient of summed scores (multiple CLIP models) is propagated to G to update C_t (see Equation 5 in [152]). After several iterations, the updated C_t is used to generate the next token $\boldsymbol{x}_{t+1} = G(\boldsymbol{x}_t, C_t)$. We repeat this process until we generate the entire caption. We cascade the captions of multiple video frames and questions about this video to prompt GPT-3 for video question answering.

GLIDE [103] and the classifier guidance diffusion [23]. We use DDIM to sample images from GLIDE in 100 steps. The guidance scale is set to 3.

C.1.2 Video question answering

In video question answering, we use the proposed method to generate captions for the video frames and then use GPT-3 to summarize the captions to answer questions. We use GPT-2 as the generator and a set of CLIP models as scorers to generate captions for each video frame. The CLIP models [119, 130] are from the Huggingface library [164]:

- CLIP-32: `https://huggingface.co/openai/clip-vit-base-patch32`.

- CLIP-14: `https://huggingface.co/openai/clip-vit-large-patch14`.
- CLIP-multilingual: `https://huggingface.co/sentence-transformers/clip-ViT-B-32-multilingual-v1`.

Fig. C-2 shows the framework for generating frame captions. Given a video frame I, we generate a sequence of words to describe it. To integrate feedback from scorers to the generator, similar to ZeroCap [152], we define a context cache C_t (a set of embedding functions in GPT-2, such as the embedding functions, K, Q, V, in the Transformer blocks.) that stores the context information generated so far, which is updated iteratively based on the feedback from scorers. The prediction of the next word from the generator G is given by $\boldsymbol{x}_{t+1} = G(\boldsymbol{x}_t, C_t)$, where G is the pre-trained language model. ZeroCap uses the following loss function to optimize C_t:

$$\arg\min_{C_t}\Big(\mathcal{L}_{\text{CLIP}}\left(\mathrm{G}\left(\boldsymbol{x}_t, C_t\right), I\right) + \lambda\mathcal{L}_{\text{CE}}\big(\mathrm{G}\left(\boldsymbol{x}_t, C_t\right), \hat{\boldsymbol{x}}_{t+1}\big)\Big), \tag{C.1}$$

where I is the feature of a video frame and $\hat{\boldsymbol{x}}_{t+1}$ is the next word predicted by the original language model. The CLIP loss $\mathcal{L}_{\text{CLIP}}$ optimizes C_t to make the newly generated sentence describe the video frame. The second loss $\mathcal{L}_{\text{CE}}$ ensures the newly generated sentence is close to the sentence generated by the original language model.

Our implementation is based on the code of ZeroCap [152]. The context cache C_t is updated using:

$$C_t \longleftarrow C_t + \alpha\frac{\nabla_{C_t} p\left(\boldsymbol{x}_{t+1} \mid C_t\right)}{\left\|\nabla_{C_t} p\left(\boldsymbol{x}_{t+1} \mid C_t\right)\right\|^2}, \tag{C.2}$$

where $p(\boldsymbol{x}_{t+1}|C_t)$ is the probability of predicting word $\boldsymbol{x}_{t+1}$ given C_t. Optimizing Eq. (C.1) can be achieved by conducting the gradient descent using Eq. (C.2). In our experiments, we use 5 steps of gradient descent. The learning rate α is set to 0.3.

In the video question answering tasks, we compose multiple CLIP scores and use their composed score to optimize C_t:

$$\arg\min_{C_t}\Big(\mathcal{L}_{\text{CLIP-32}}\left(\mathrm{G}\left(\boldsymbol{x}_t, C_t\right), I\right) + \mathcal{L}_{\text{CLIP-14}}\left(\mathrm{G}\left(\boldsymbol{x}_t, C_t\right), I\right) \tag{C.3}$$

$$+\,\mathcal{L}_{\text{CLIP-multilingual}}\left(\mathrm{G}\left(\boldsymbol{x}_t, C_t\right), I\right) + \lambda\mathcal{L}_{\text{CE}}\big(\mathrm{G}\left(\boldsymbol{x}_t, C_t\right), \hat{\boldsymbol{x}}_{t+1}\big)\Big). \tag{C.4}$$

```
# Q: how many people are there in the video
# A: 2
# Q: what is behind the person in white clothes
# A: tree
# Q: what is in front of the person with braid
# A: chair
...
# Q: what is the person in white doing
# A: tie hair
# Q: what happened to the person in gray after he threw a goal
# A: clap with your teammates
# Summarize the following descriptions and answer the question as shown above:
a Video showing the new Hair tutorial; a video showing young blond hair clip attaching to
top pony tail of teens hair; ...; a video on the head hair clip website showing blonde long
hair twisted in two knots.

# Q: is the person with a golden hair long hair
```

Figure C-3: **Prompts given to GPT-3 for video question answering.** Text in black contains the question-answer pairs randomly sampled from the ActivityNet-QA training dataset. Text in blue has the video frame captions generated by the proposed method. Text in orange is the question about this video that needs to be answered.

After several iterations, the updated C_t is used to generate the next token $\boldsymbol{x}_{t+1} = G(\boldsymbol{x}_t, C_t)$. We repeat this process until we generate the entire caption.

To answer the video questions, we cascade the generated captions of the video frames and the questions about this video to prompt GPT-3 to generate answers. For each video, we delete the first 10 frames and the last 10 frames to remove the beginning or ending advertisements. We then take 30 video frames evenly from the rest frames and send them to GPT-3. To guide GPT-3 to generate proper answers, we randomly select 30 question-answer pairs from the training set of ActivityNet-QA [172] and use them as part of the prompt of GPT-3. As shown in Fig. C-3, the prompt of GPT-3 consists of examples of question-answer pairs, the video frame captions generated by the proposed method, and the question about this video that needs to be answered. The text generated by GPT-3 is used as the answer to the question asked. We also used the profanity check tool (`https://github.com/vzhou842/profanity-check`) to remove the improper answers.

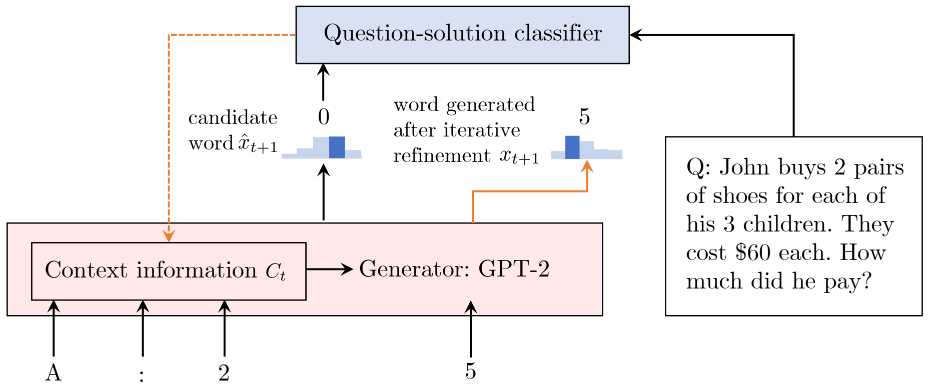

Question-solution classifier
candidate word $\hat{x}_{t+1}$
0
word generated after iterative refinement x_{t+1}
5
Q: John buys 2 pairs of shoes for each of his 3 children. They cost $60 each. How much did he pay?
Context information C_t
Generator: GPT-2
A
:
2
5

Figure C-4: **Overview of solving grade school math problems.** We use GPT-2 as the generator and treat the grade school math problem as a text-generation problem. The scorer, a pre-trained question-solution classifier, provides the generator feedback to guide the next token's generation $\boldsymbol{x}_{t+1}$. We follow the approach used in VQA to iteratively optimize the generations based on feedback from scorers. Our generator G first generates a set of candidate words $\hat{\mathbf{X}}_{t+1} = \{\hat{\boldsymbol{x}}_{t+1}\}$, and then the classifier predicts the probability of each solution (the concatenation of previous words and each new word $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \hat{\boldsymbol{x}}_{t+1}\}$, where $\hat{\boldsymbol{x}}_{t+1} \in \hat{\mathbf{X}}_{t+1}$) matching the given question. The classifier score is the cross-entropy loss between this new probability distribution and the original distribution of the next word obtained from the generator G. The gradient of the classifier score is used to update C_t through iterative refinement (see Equation 5 in [152]). The updated C_t is used to predict the next word $\boldsymbol{x}_{t+1} = G(\boldsymbol{x}_t, C_t)$. We repeat this process until we generate the complete solution.

C.1.3 Grade school math

We treat the grade school math problem as a text generation problem. As shown in Fig. C-4, we use GPT-2 as the generator and a pre-trained question-solution classifier as the scorer. The pre-trained classifier is a binary classifier trained on the training set of GSM8K [17]. Given a math problem, such as "Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?", and an answer, such as "72". If the answer is correct for the given problem, then the label is 1; otherwise, the label is 0.

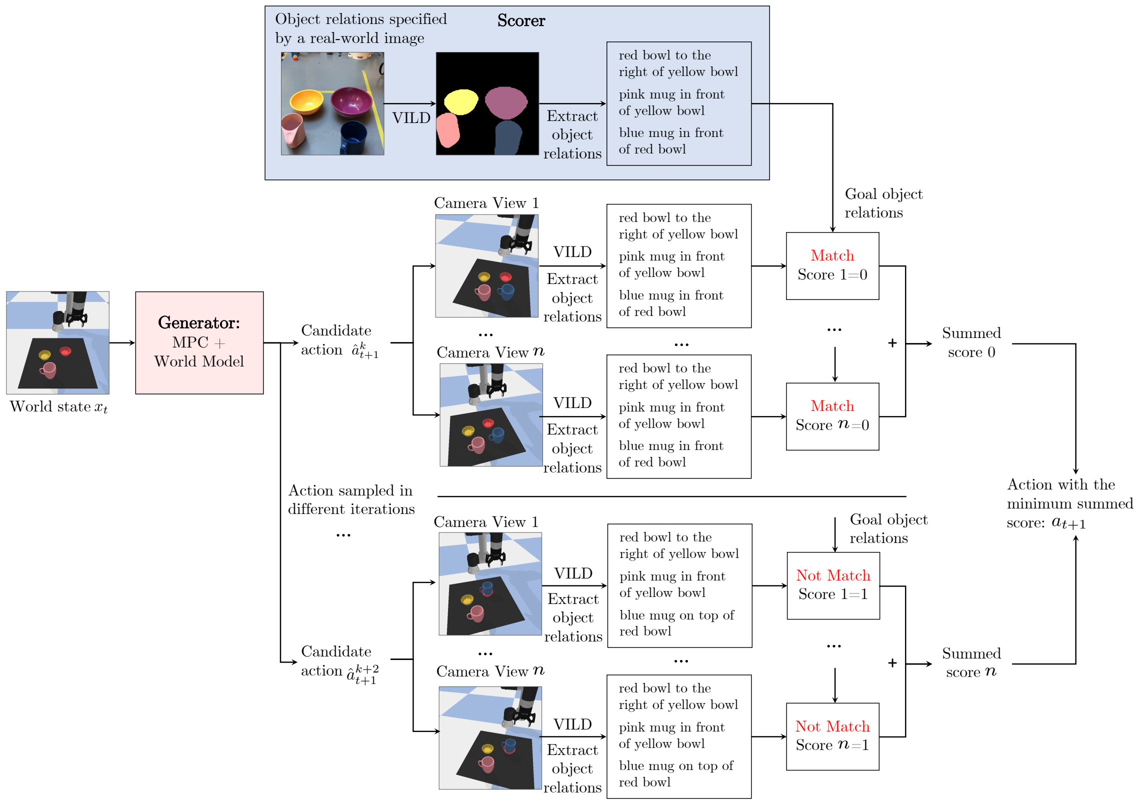
After training, the classifier is used as the scorer to provide feedback to the generator to guide the next token's generation $\boldsymbol{x}_{t+1}$. Similar to VQA, the generator G first generates a set of candidate words $\hat{\mathbf{X}}_{t+1} = \{\hat{\boldsymbol{x}}_{t+1}\}$, and then the classifier predicts the probability of each solution (the concatenation of previous words and each new word $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \hat{\boldsymbol{x}}_{t+1}\}$, where $\hat{\boldsymbol{x}}_{t+1} \in \hat{\mathbf{X}}_{t+1}$) matching the given question.

The classifier score is the cross-entropy loss between this new probability distribution and the original distribution of the next word obtained from the generator G (the way to compute the classifier score is the same as computing the CLIP score in VQA). We also used the cross-entropy loss $\mathcal{L}_{\text{CE}}$ in Equation 2 of ZeroCap [152] to ensure the generated sentence is grammatically sound. The context cache C_t is updated in the same way as Equation 5 in [152], but we use the classifier score when providing the feedback to C_t. The updated C_t is used to predict the next word $\boldsymbol{x}_{t+1} = G(\boldsymbol{x}_t, C_t)$. We repeat this process until we generate the complete solution. Similarly to the video question task, we use 5 steps of gradient descent. The learning rate α is set to 0.3.

C.1.4 Robot manipulation

In robot manipulation, we use the proposed method to manipulate objects in Ravens [173] to conform to a set of object relations specified by text descriptions or real-world images. We use MPC+World Model as the generator and ViLD [42] as the scorer. As shown in Figure C-5, given a real-world image, our model manipulates objects in the environment to achieve a state with objects having the same object relations as the given image. We first use ViLD to generate a 2D segmentation of the real-world image and the corresponding text label, such as "mug", for each segment. We then use the relative pixel-wise offsets of segmentation masks and the text labels to infer a set of object relations (top panel of Figure C-5).

Given the current world state $\boldsymbol{x}_t$, we aim to generate an action a_{t+1} so that the new world state after executing a_{t+1} has object relations closer to the object relations in the given image. To do this, we first use the generator (MPC+World Model) to generate a set of candidate actions $\{\hat{a}^k_{t+1}\}$ and the corresponding world states $\{\hat{\boldsymbol{x}}^k_{t+1}\}$ after executing each candidate action. For each new world state $\hat{\boldsymbol{x}}^k_{t+1}$, we render n 2D images from n camera views. Each rendered image is sent to VILD to get a segmentation map and text labels. We project the objects into 3D space based on the segmentation map and the depth map of the image. We then obtain the object relations based on their 3D positions and the predicted text labels. We compare the object relations obtained from each rendered image and the object relations obtained



Object relations specified by a real-world image
Scorer
VILD
Extract object relations
red bowl to the right of yellow bowl
pink mug in front of yellow bowl
blue mug in front of red bowl
Goal object relations
World state x_t
Generator: MPC + World Model
Candidate action $\hat{a}^k_{t+1}$
Camera View 1
Camera View n
Match
Score 1=0
Match
Score n=0
Summed score 0
Action sampled in different iterations
Candidate action $\hat{a}^{k+2}_{t+1}$
blue mug on top of red bowl
Not Match
Score 1=1
Not Match
Score n=1
Summed score n
Action with the minimum summed score: a_{t+1}

Figure C-5: **Overview of robot manipulation.** We use MPC+World Model as the generator and ViLD as the scorer to manipulate objects to conform to a set of object relations specified by text descriptions or real-world images. **Top:** Given a real-world image, we first use ViLD to generate a 2D segmentation of the real-world image and the corresponding text label, such as "mug", for each segment. We then use the relative pixel-wise offsets of segmentation masks and the text labels to infer a set of object relations. **Bottom:** Given the current world state $\boldsymbol{x}_t$, we aim to generate an action a_{t+1} so that the new world state after executing a_{t+1} has object relations closer to the object relations in the given image. To do this, we first use the generator (MPC+World model) to generate a set of candidate actions $\{\hat{a}^k_{t+1}\}$ and the corresponding world states $\{\hat{\boldsymbol{x}}^k_{t+1}\}$ after executing each candidate action. For each new world state $\hat{\boldsymbol{x}}^k_{t+1}$, we render n 2D images from n camera views. Each rendered image is sent to VILD to get a segmentation map and text labels. We project the objects into 3D space based on the segmentation map and the depth map of the image. We then obtain the object relations based on their 3D positions and predicted text labels. We compare the object relations obtained from each rendered image and the object relations obtained from the real-world image to compute the score. The score is 0 if the relations are matching; otherwise, 1. We sum the scores from each rendered image to obtain the final score. We choose the action a_{t+1} that leads to a world state with the minimum summed score. We execute a_{t+1} in the environment and get a new state $\boldsymbol{x}_{t+1}$. We repeat this process until the task is accomplished or we are at the final step T.

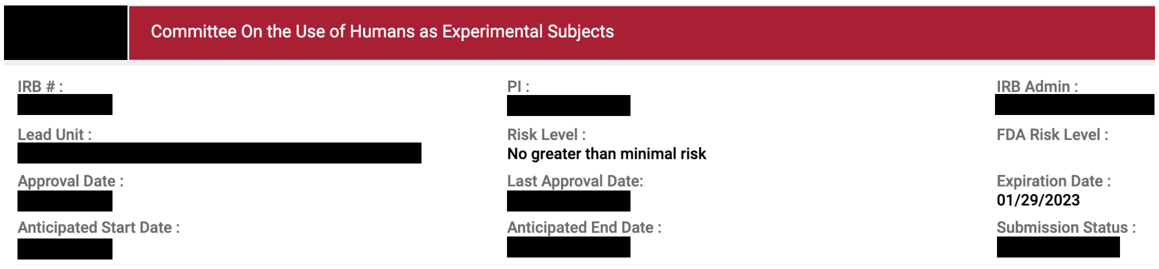
from the real-world image to compute the score. The score is 0 if the relations are matching; otherwise, 1. We sum the scores from each rendered image to obtain the final score. We choose the action a_{t+1} that leads to a world state with the minimum summed score. We execute a_{t+1} in the environment and get a new state $\boldsymbol{x}_{t+1}$. We repeat this process until the task is accomplished or we are at the final step T, where T equals the number of relations extracted from the real-world image.

C.1.5 A unified framework for composing pre-trained models

Our method shares some similar architecture with existing works, such as ZeroCap [152] and CLIP-guided diffusion models [103]. However, the focus of our paper is to propose a general framework for composing different pre-trained models across a variety of tasks, and these particular methods are concrete instantiations of our proposed framework. In addition, in this work, we also illustrate how we may combine ensembles of different pre-trained models as scorers to leverage the "wisdom of the crowds" where each scorer provides complementary feedback to the generator, compensating for the potential weaknesses of other scorers. Through iterative optimization and the composition of multiple scorers, our method shows effective zero-shot generalization ability on various multimodal tasks.

C.2 Ethics Statement of Amazon Mechanical Turk Experiments

For the video question answering tasks, we ask workers from Amazon Mechanical Turk to evaluate the generated answer based on the video and the asked question. Before showing the questions and answers to the workers, we used the profanity check tool (`https://github.com/vzhou842/profanity-check`) to remove the improper questions and answers. As shown in Fig. C-6, this experiment was approved by the Committee on the Use of Humans as Experimental Subjects. A screenshot of the task is shown in Fig. C-7. The instructions shown to participants are listed as follows:



[redacted] Committee On the Use of Humans as Experimental Subjects

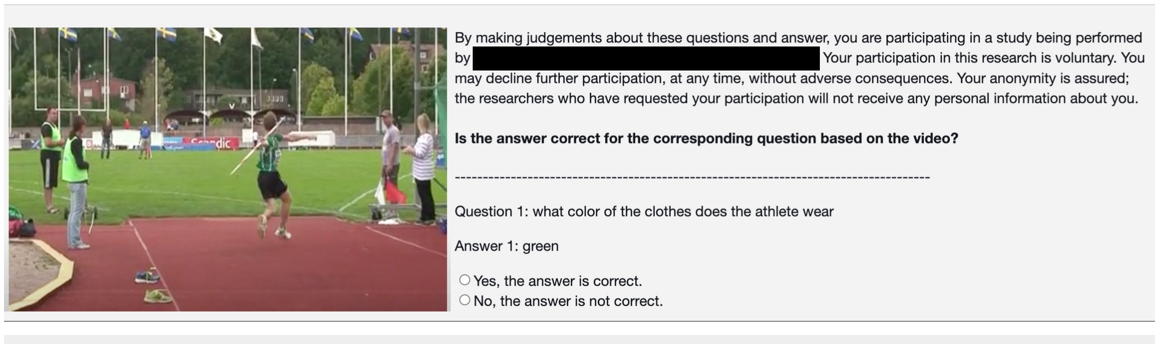
IRB # : [redacted]	PI : [redacted]	IRB Admin : [redacted]
Lead Unit : [redacted]	Risk Level : No greater than minimal risk	FDA Risk Level :
Approval Date : [redacted]	Last Approval Date: [redacted]	Expiration Date : 01/29/2023
Anticipated Start Date : [redacted]	Anticipated End Date : [redacted]	Submission Status : [redacted]

Figure C-6: **Human experiments approval form.** Screenshot of the approval form from the Committee on the Use of Humans as Experimental Subjects.

> **Instructions:** By making judgments about these questions and answers, you are participating in a study being performed by [XXX]. Your participation in this research is voluntary. You may decline further participation, at any time, without adverse consequences. Your anonymity is assured; the researchers who have requested your participation will not receive any personal information about you.

Given a video, a question, and a generated answer, the workers from Amazon Mechanical Turk measure whether the answer is correct for the given question and video. Each video shows three question-answer pairs (only one question-answer pair is shown in the screenshot). The answers are generated by different methods. The workers are not told which method generates each answer. The workers are asked to choose “yes” or “no”. If the worker thinks the answer matches the given video and question, they should choose “yes”; otherwise, “no”.

To control the quality, each task is evaluated by three different workers. The workers are required to have an approval rate greater than 98%. Our test shows that each task takes around 10 seconds, but the workers are given up to one hour to complete each task. The workers are paid \$0.05 for finishing each task with an estimated hourly payment of \$18, more than the United States federal minimum wage. There are 33 workers in total who joined our experiment.


By making judgements about these questions and answer, you are participating in a study being performed by [redacted] Your participation in this research is voluntary. You may decline further participation, at any time, without adverse consequences. Your anonymity is assured; the researchers who have requested your participation will not receive any personal information about you.

Is the answer correct for the corresponding question based on the video?

Question 1: what color of the clothes does the athlete wear

Answer 1: green

○ Yes, the answer is correct.
○ No, the answer is not correct.

Figure C-7: **Screenshot of Amazon Mechanical Turk we used for the video question answering experiment.** Workers are shown a video, three questions, and the answer to each question. The answers are generated by different methods. The workers are not told which method generates each answer. The workers are asked to select "yes" or "no" based on their measurement of whether the answer is correct for the given video and question.

C.3 Additional Results

C.3.1 Image generation results

We show more image generation results using different scorer models in Fig. C-8. We find that most of the time, the proposed framework, either using a single scorer model or composing multiple scorer models, works well as shown in the first four examples in the left column of Fig. C-8. In some hard cases, some scorer models might fail (the rest examples in Fig. C-8). However, there is no discernible trend on which scorer is better on what tasks. The results of composing multiple scorer models are significantly better than using a single one, as different scorer models capture different aspects of the information. For example, in the fifth example in the left column of Fig. C-8, our method PIC with classifier-free guidance (CLS-FREE) and PIC with a pre-trained classifier (CLS) cannot generate an image with "tench", but PIC with the pre-trained CLIP (CLIP) can generate the correct result and the composed model (CLS-FREE + CLS + CLIP) also works. This is why we consider composing multiple scorers: to leverage the strength of each expert model and improve the worst case.

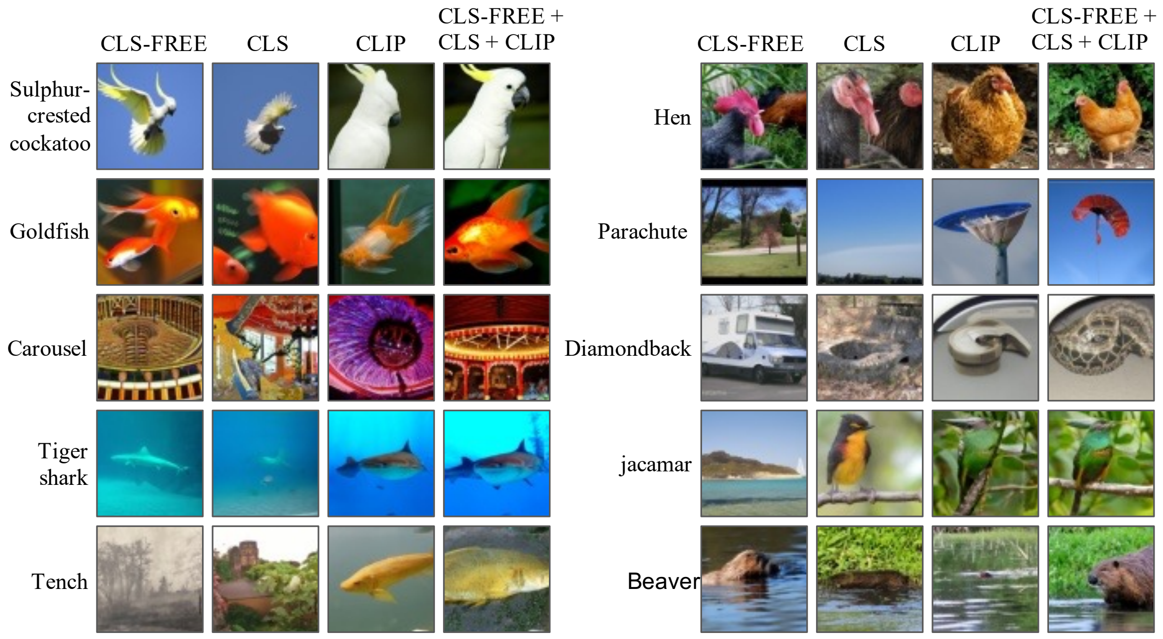


Figure C-8: **Qualitative results.** Image generation results using different scorer models. Composing multiple scorers (CLS-FREE + CLS + CLIP) achieves the best performances.

Table C.1: **Our framework can be applied to other generators.** In the image generation task, we use a new generator, *i.e.*, Stable-Diffusion [131]. Using a more powerful pre-trained model can further boost performance. Image generation results on ImageNet are reported.

Method Name	Generator	Scorer	**IS** ↑	**FID** ↓	**KID** ↓
PIC (G+E)	GLIDE	CLS-FREE	25.926	29.219	5.325
PIC (G+E)	Stable-Diffusion	CLS-FREE	31.689	29.546	6.562

C.3.2 Image generation with different generator

Our framework can be applied to other generators as well. For example, we changed the generator, GLIDE, to Stable Diffusion [131]. The results of composing Stable Diffusion and classifier-free guidance (CLS-FREE) are shown in Table C.1. Using a more powerful pre-trained model can further boost performance.

C.3.3 Composing scorer models

The key idea of our method is to compose ensembles of pre-trained models, and the way to combine them can be variant, such as adding all the scores as we used or taking

Table C.2: **Different ways to compose scorer models.** Composing scorers using their summed score generates the best results. Image generation results on ImageNet are reported.

Method Name	**Generator**	**Scorer**	**IS ↑**	**FID ↓**	**KID ↓**
PIC (G+E1)	GLIDE	CLIP	25.017	30.462	6.174
PIC (G+E2)	GLIDE	CLS	22.077	30.871	7.952
PIC (G+E1+E2) Sum	GLIDE	CLIP + CLS	**30.438**	**29.543**	**5.435**
PIC (G+E1+E2) Max	GLIDE	CLIP + CLS	24.782	30.657	6.040
PIC (G+E1+E2) Weighted	GLIDE	CLIP + CLS	24.728	30.669	6.420

the best scorer. Here we add two additional experiments for comparison: (1) using the best scorer (scorer that provides the highest score) and (2) using the weighted scores and adding them together. Our method composes pre-trained models without training or finetuning. Thus we did not learn separated weights for different models. We use the scorers (after softmax) of each scorer model as their weights and then compose the scorers using the weighted summed score. We compare these two baselines with our method which uses the summation of all scores. As shown in Table C.2, using a summed score generates the best results.

Appendix D

Supplementary: Pre-Trained Language Models for Interactive Decision-Making

We first show the convolutional encoding in BabyAI in Appendix D.1. We then describe the environment details in Appendix D.2 and the implementation details of the proposed model in Appendix D.3. We show the algorithm of interactive evaluation in Appendix D.4 and the data gathering procedure in Appendix D.5. The goal predicates used in VirtualHome test subsets are shown in Appendix D.6. We visualize the attention weights in language models in Appendix D.7.

D.1 Convolutional Encoding in BabyAI

In this part, we show the third way of encoding policy inputs in BabyAI. We test a new model, **PIC-Conv (Ours)**, that converts environment inputs into *convolutional embeddings*. We pass the $7 \times 7 \times 3$ grid observation in BabyAI to convolutional layers and obtain a $7 \times 7 \times d$ feature map, where d is the feature dimension. We flatten the feature map and get a sequence of features to describe the observation. The rest of the model is the same as *PIC-Text (Ours)*. Table D.1 shows the results of policies using the *text encoding* and *convolutional encoding*. *PIC-Text (Ours)* and *PIC-Conv (Ours)*

Table D.1: **Success rate of policies trained with *text encoding* vs. *convolutional encoding* on BabyAI.** The text encoding is more sample-efficient, but both models converge to near-perfect performance given sufficient training data.

Tasks	Methods	Number of Demos				
		100	**500**	**1K**	**5K**	**10K**
GoToRedBall	PIC-Text (Ours)	**93.9**	**99.4**	99.7	**100.0**	**100.0**
	PIC-Conv (Ours)	92.5	98.8	**100.0**	**100.0**	**100.0**
GoToLocal	PIC-Text (Ours)	64.6	**97.9**	**99.0**	99.5	99.5
	PIC-Conv (Ours)	**69.5**	86.0	98.2	**99.9**	**99.9**
PickupLoc	PIC-Text (Ours)	**28.7**	**73.4**	**99.0**	**99.6**	99.8
	PIC-Conv (Ours)	25.0	58.8	95.1	**99.6**	**100.0**
PutNextLocal	PIC-Text (Ours)	11.1	**93.0**	**93.2**	**98.9**	**99.9**
	PIC-Conv (Ours)	**17.9**	53.6	91.3	97.7	99.5

have similar results given enough training data, but *PIC-Text (Ours)* is slightly better when there are fewer training data. This conclusion is coincident with the results on VirtualHome.

Different input encoding schemes have only a negligible impact on model performance: the effectiveness of pre-training is not limited to utilizing natural strings, but in fact extends to arbitrary sequential encodings.

D.2 Environments

We use **BabyAI** [59] and **VirtualHome** [117] to evaluate the proposed method. While both environments feature complex goals, the nature of these goals, as well as the state and action sequences that accomplish them, differ substantially across environments.

D.2.1 VirtualHome

VirtualHome is a 3D realistic environment featuring partial observability, large action spaces, and long time horizons. It provides a set of realistic 3D homes and objects that can be manipulated to perform household organization tasks.

Goal Space. For each task, we define the goal as a set of predicates and multiplicities. For example, `Inside(apple, fridge):2`; `Inside(pancake, fridge):1`; means "put two apples and one pancake inside the fridge". In each task, the initial environment (including initial object locations), the goal predicates, and their orders and multiplicities are randomly sampled. There are 59 different types of predicates in total.

Observation Space. The observation in VirtualHome, by default, is a graph describing a list of objects and their relations in the current partial observation. Each object has an object name, a state, *e.g.*, *open, close, clean*, and 3D coordinates.

Action Space. Agents can navigate in the environment and interact with objects. To interact with an object, the agent must predict an action name and the index of the interested object, *e.g.*, `Open(5)` to open the object with index (5). The agent can only interact with objects that are in the current observation or execute the navigation actions, such as `Walk(bathroom)`. For some actions, such as `open`, the agent must be close to the object. There are also strict preconditions for actions, *e.g.*, the agent must `grab` an object before it can `put` the object on a target position. As a result of these constraints, the subset of actions available to the agent changes at every timestep.

We evaluate the success rates of different methods on VirtualHome. A given episode is scored as successful if the policy completes its entire goal within T steps, where $T = 70$ is the maximum allowed steps of the environment.

D.2.2 BabyAI

BabyAI is a 2D grid world environment designed to evaluate the instruction following. Different from VirtualHome, the observation in BabyAI by default is a 7×7 grid describing a partial and local egocentric view of the state of the environment. Each tile in the grid contains at most one object, encoded using 3 integer values: one for the object type, one for the object color, and a state for doors indicating whether it is open, closed, or locked. The goals in BabyAI are language instructions, *e.g.*, "put the blue key next to the purple ball". BabyAI has 7 actions, *e.g.*, "turn left", "pick up", and "drop".

D.3 Implementation Details of PIC in VirtualHome

In Appendix D.3.1, we provide more details of the model architecture. We then introduce the training detail in Appendix D.3.2.

D.3.1 Model architecture details in VirtualHome

In this part, we provide more details about the policy network we used in VirtualHome. Our policy model consists of three parts, *i.e.*, inputs, the pre-trained LM, and outputs. As shown Fig. D-1, we encode the inputs to the policy—including goal g, history h_t, and the current partial observation o_t—as sequences of embeddings. These embeddings are passed to the LM (using its pre-trained embedding layer F_θ) and used to obtain contextualized token representations. These token representations are averaged to generate a context feature f_c, which is then passed to a fully connected layer to predict the next action a_t. The output action in VirtualHome consists of a verb and an object. For brevity, we omit the time subscript t from now on.

In VirtualHome, the partial observation o of the environment state can be represented as a list of objects in the agent's view. We represent each object by its name, *e.g.*, "oven", a state description, *e.g.*, "open, clean", and position both in the world and relative to the agent. In this part, we provide more details of how **PIC-Text (Ours)** encodes the name, state, and position of each object in the observation. Figure D-2 shows the model architecture we used to encode the observation.

Name encoding. For each object node, we serialize its object name as an English phrase s^o. We extract its tokens and features using the tokenizer and the embedding layer of the pre-trained LM, respectively. Since one object name might generate several English tokens using the tokenizer from the pre-trained LM, *e.g.*, the tokens of "kitchencabinet" is $[15813, 6607, 16212, 500]$, we take the averaged features of all the tokens in the object name and obtain a "name" feature $f_i^{o,\text{name}}$ for each object node as shown in Figure D-2.

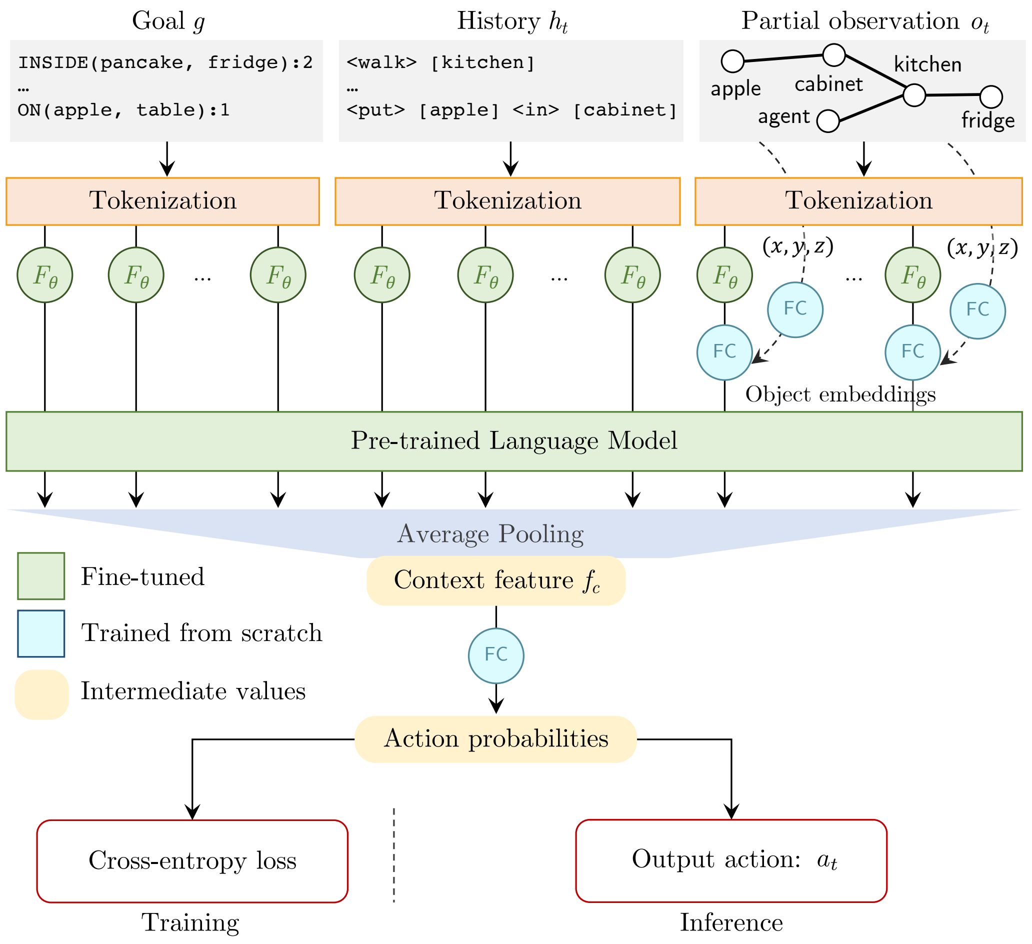


Figure D-1: **Policy network in VirtualHome.** The observation, goal, and history are first converted into sequences and then passed through an embedding layer F_θ. The combined sequence is passed through a pre-trained LM, and the output tokens are pooled into a context feature vector for action prediction.

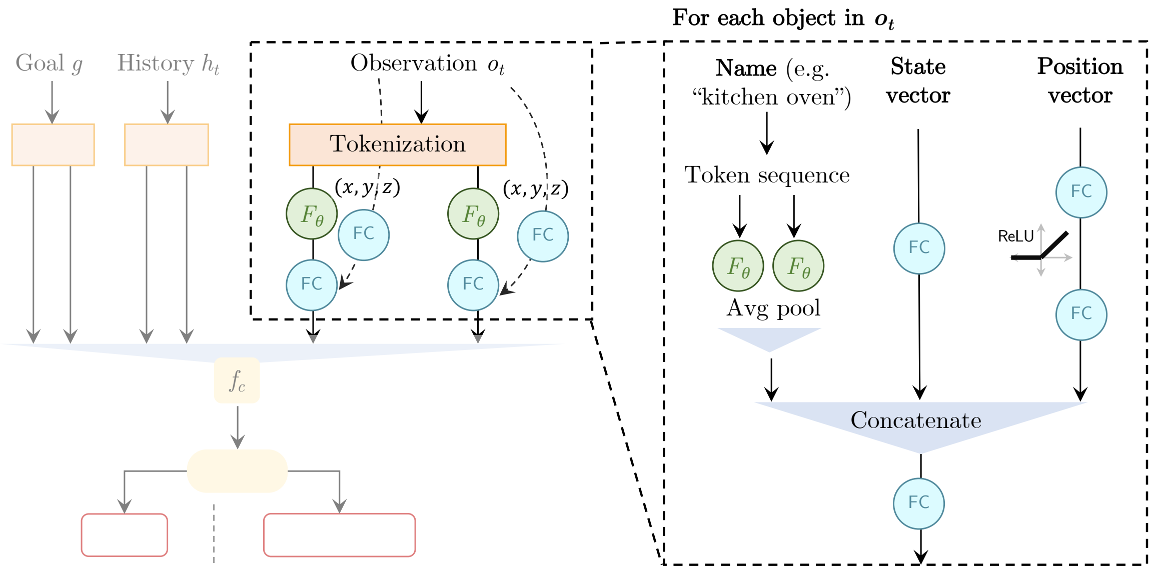


Figure D-2: **Object encoding.** In VirtualHome, the partial observation of the environment state can be represented as a list of objects in the agent's view. Each object is represented by a name, a state vector, and a position vector. **Object name encoding:** each object's name is an English phrase. We tokenize the phrase, embed the tokens, and average the embeddings. **Object state encoding:** each object is assigned one of six states: "clean", "closed", "off", "on", "open", or "none". This state is represented as a 6-dimensional binary vector and passed through a fully-connected layer. **Object position encoding:** an object's position vector is a 6-dimensional vector containing its world coordinates alongside its displacement to the agent (*i.e.*, the difference in their world coordinates). This position vector is passed through two fully connected layers. These three features are concatenated and passed through a fully connected layer to obtain the representation of an object in the current observation.

State encoding. Some objects have a state description, *e.g.*, "oven: open, clean". There are six types of object states in the environment: "clean", "closed", "off", "on", "open", and "none". For each object node, we use a binary vector to represent its state. Taking the "oven" as an example, if the oven is open and clean, its state vector would be $[1, 0, 0, 0, 1, 0]$. This state vector is then passed through a fully connected layer to generate a state feature $f_i^{o,\text{state}}$ of the object o_i.

Position encoding. To encode the position information of each object o_i, we take their world coordinates $\{o_{i,x}, o_{i,y}, o_{i,z}\}$ and their spatial distance to the agent $\{a_x, a_y, a_z\}$ to generate a position vector $[o_{i,x}, o_{i,y}, o_{i,z}, o_{i,x} - a_x, o_{i,y} - a_y, o_{i,z} - a_z]$. This position vector is then passed through two fully connected layers with a ReLU layer in the middle to generate a position feature $f_i^{o,\text{position}}$ of the object o_i.

The final feature f_i^o of each object node is obtained by passing the concatenation of its name feature $f_i^{o,\text{name}}$, state feature $f_i^{o,\text{state}}$, and position feature $f_i^{o,\text{position}}$ through a fully connected layer. The observation at a single step can be written as a set of features $\{f_1^o, \cdots, f_N^o\}$, where N is the number of objects in the current observation.

D.3.2 Implementation details

Our proposed approach and baselines are trained on Tesla 32GB GPUs. We train every single model on 1 Tesla 32GB GPU. All experiments used the AdamW optimizer with the learning rate of 10^{-5}. We utilize a standard pre-trained language model, GPT-2, in our experiments. GPT-2 is trained on the Webtext dataset [122] using the Huggingface library [164].

D.4 Interactive Evaluation

The algorithm for interactive evaluation is shown in Algorithm 2.

Algorithm 2: Interactive evaluation

```
A set of task goals $G$ (each goal has a corresponding initial state);
Load the learned policy $\pi_\phi$;
Successful trajectory count: $n = 0$;
for example=1, $N_{test}$ do
    Sample a goal $g$ and an initial state;
    for $t = 0, T$ do
        Sample an action $a_t$ from policy $\pi_\phi(a_t|g, h_t, o_t)$;
        Execute the action $a_t$ and get a new observation $o_{t+1}$;
        if success then
            $n = n + 1$;
            break;
        end
    end
end
success rate: $r = n/N_{\text{test}}$;
```

D.5 Data Gathering Details in VirtualHome

In this part, we provide more data gathering details in VirtualHome for training the decision-making policies. We introduce the expert data collection and active data gathering in Appendix D.5.1 and Appendix D.5.2, respectively.

D.5.1 Expert data collection

VirtualHome-imitation learning dataset. To train the models, we collect a set of expert trajectories in VirtualHome using regression planning (RP) [75]. We follow the implementation of the regression planner used in [118]. Given a task described by goal predicates, the planner generates an action sequence to accomplish this task. As shown in Figure D-3, the agent has a belief about the environment, *i.e.*, an imagined distribution of object locations. As the agent explores the environment, its belief of the world becomes closer to the real world. At every step, the agent updates its belief based on the latest observation (see [118]), finds a new plan using the regression planner, and executes the first action of the plan. If the subtask (described by the goal predicate) has been finished, the agent will select a new unfinished subtask; otherwise, the agent will keep doing this subtask until it finishes.

Similarly to previous work [144, 141, 118], we generate training data using a

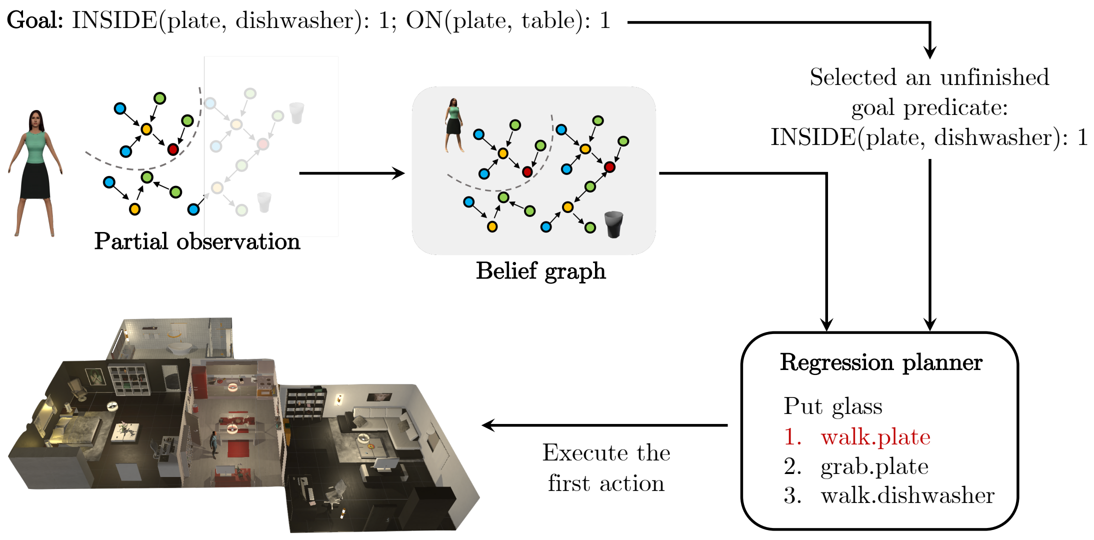


Figure D-3: **Regression planner**. Given a task described by goal predicates, the planner generates an action sequence to accomplish this task. The agent has a belief about the environment, *i.e.*, an imagined distribution of object locations. As the agent explores the environment, its belief of the world becomes closer to the real world. At every step, the agent updates its belief based on the latest observation, finds a new plan using the regression planner, and executes the first action of the plan. If the subtask (described by the goal predicate) has been finished, the agent will select a new unfinished subtask, otherwise, the agent will keep doing this subtask until finishing it.

planner that has access to privileged information, such as full observation of the environment and information about the pre-conditions and effects of each action. The planner allows an agent to robustly perform tasks in partially observable environments and generate expert trajectories for training and evaluation. We generate $20,000$ trajectories for training and $3,000$ trajectories for validation. Each trajectory has a goal, an action sequence, and the corresponding observations after executing each action.

D.5.2 Active data gathering

The algorithm for active data gathering is shown in Algorithm 3. To sample the goal and initial state, we first generate a set of initial states in VirtualHome using the code released by [118]. For each initial state, we are able to get a set of feasible tasks that can be accomplished in this environment. For example, in an initial state, if the

Algorithm 3: Active Data Gathering

```
Given: a goal relabel function $f_l$;
Initialize: policy $\pi_\phi$; goal set $G$; training replay buffer $\mathcal{R}_{train} = \{\}$; validation
  replay buffer $\mathcal{R}_{val} = \{\}$;
for iteration=1, N do
    for example=1, M do
        Sample a goal $g$ from $G$ and an initial state $s_1$;
        for $t = 1, T$ do
            Sample an action from policy $\pi_\phi(a_t|g, h_t, o_t)$ or sample an action
              randomly;
            Execute $a_t$ and obtain a new observation $o_{t+1}$;
        end
        Store the trajectory $(o_1, a_1, \cdots, o_T, a_T, g)$ in the replay buffer $\mathcal{R}_{train}$ or $\mathcal{R}_{val}$;
    end
    Relabel each failure trajectory $d = (o_1, a_1, \cdots, o_T, a_T)$ in the replay buffers and
      get new goal $g' = f_l(d)$;
    Put new goals $g'$ in the goal set $G$;
    for  $k = 1, K$ do
        repeat
            Sample data from $\mathcal{R}_{train}$ and update policy $\pi_\phi$;
        until training episode ends;
        Get validation accuracy using the data from $\mathcal{R}_{val}$;
    end
    $\pi_\phi = \pi_{\text{val\_best}}$
end
```

apple is on the kitchen table, a feasible task goal could be “put the apple inside the fridge”. In contrast, “put the banana inside the fridge” is not a feasible task if there is no banana in the initial state.

We collect 9893 initial states, and randomly sample an initial state and its feasible goal every time when we reset the environment. After each data collection iteration, we obtain a set of new goals using the goal relabel function. We save the goal and its corresponding initial state in the replay buffers and use the same strategy to sample the goal and initial state in the next iteration.

The *hindsight relabeling* stage is the key component for active data gathering. Here we provide more implementation details of how we relabel “failed” trajectories with new goals in the *hindsight relabeling* stage. For each “failed” trajectory, we extract its useful sub-trajectories and relabel a task goal g' for it. We design a goal relabel function f_l that generates a goal based on the sequence of observations and actions. To do this, we first use a hand-designed program to detect what tasks are contained in a "failed" trajectory. This program finds useful tasks based on the keywords in the action list. For example, in Fig. D-4, the program knows the trajectory containing a task of "`On(apple, kitchen table):1`" based on the action "$[put] < apple >< kitchentable >$".

The selected sub-trajectories are not always optimal. We thus design a rule to filter out bad trajectories, *i.e.*, for trajectories with the same goal, selecting the "shorter" ones. One example is shown in Fig. D-5. Suppose that there are two trajectories having the same goal, *e.g.*, "`On(apple, kitchen table):1`". The first trajectory has actions that are redundant or not related to the task, such as "$[walk] < bathroom >$" and "$[walk] < kitchen >$" while the second trajectory is more optimal given the goal. We select the second trajectory and delete the first trajectory from the replay buffer. Note that the "shorter" does not mean fewer actions, but fewer actions that are not related to the task. The *hindsight relabeling* stage allows sample-efficient learning by reusing the failure cases. The relabeled data are used to train policies in the *policy update* stage.

D.6 Test Sets in VirtualHome

In this part, we provide more details of each test set. We first introduce the test sets used for evaluating the proposed model trained on expert data, *i.e.*, PIC, in Section D.6.1. We then show the test sets used for evaluating the proposed model with active data gathering, *i.e.*, PIC-ADG, in Section D.6.2.

D.6.1 PIC test sets

In Table D.2, we provide a detailed description of each test subset, including the count of goal predicate types and the number of goal predicates in each task. The **In-Distribution** setting has 37 goal predicates in total, and each task has $2 \sim 10$

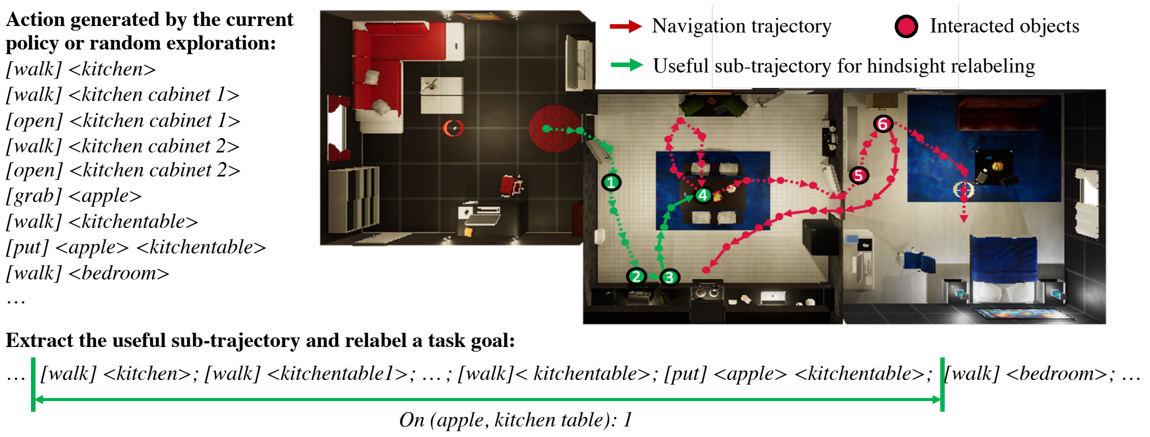

Action generated by the current policy or random exploration:
[walk] <kitchen>
[walk] <kitchen cabinet 1>
[open] <kitchen cabinet 1>
[walk] <kitchen cabinet 2>
[open] <kitchen cabinet 2>
[grab] <apple>
[walk] <kitchentable>
[put] <apple> <kitchentable>
[walk] <bedroom>
…
Navigation trajectory
Interacted objects
Useful sub-trajectory for hindsight relabeling
1
2
3
4
5
6
Extract the useful sub-trajectory and relabel a task goal:
… [walk] <kitchen>; [walk] <kitchentable1>; … ; [walk]< kitchentable>; [put] <apple> <kitchentable>; [walk] <bedroom>; …
On (apple, kitchen table): 1

Figure D-4: **Hindsight relabeling.** We first use a hand-designed program to detect what tasks are contained in the collected trajectory. This program finds tasks based on the keywords in the action list. For example, the program knows the trajectory containing a task of "`On(apple, kitchen table):1`" based on the action "$[put] < apple >< kitchentable >$". Then the program extracts all previous actions related to this task using hand-designed rules.

goal predicates. The tasks are drawn from the same distribution as the training tasks. The **Novel Scenes** setting also has 37 goal predicates, and each task has $2 \sim 10$ goal predicates. The objects are randomly placed in the initial environment. The **Novel Tasks** setting has 22 goal predicates in total, and each task has $2 \sim 8$ goal predicates. The tasks are never seen during training.

D.6.2 PIC-ADG test sets

One limitation of active data gathering is that it relies on hand-designed rules for task relabeling. In addition, it is sometimes challenging to define effective rules to extract useful sub-trajectories and get high-quality hindsight labels, especially when trajectories are long and tasks become more complex. Thus we only relabel short sub-trajectories, where the goal consists of a single goal predicate, *e.g.*, "`On(apple, kitchen table):1`". During testing, we evaluate the success rate of approaches on solving such tasks as well, *i.e.*, the count of the goal predicate equals 1. The types of goal predicates are the same as Appendix D.6.1, *i.e.*, 37 goal predicates in the *In-Distribution* setting and the *Novel Scenes* setting, and 22 goal predicates in the *Novel Tasks* setting.

Goal: *On (apple, kitchen table): 1*

Action list 1:
… [walk] <livingroom>; [grab] <apple>; [walk] <kitchen>; [walk] <bathroom>; [walk] <kitchen>; [put] <apple> <kitchentable> …

Action list 2:
… [walk] <livingroom>; [grab] <apple>; [walk] <kitchen>; [put] <apple> <kitchentable> …

Figure D-5: **Sub-trajectory selection.** Suppose there are two trajectories having the same goal, *e.g.*, "`On(apple, kitchen table):1`". The first trajectory has actions that are redundant or not related to the task, such as "$[walk] < bathroom >$" and "$[walk] < kitchen >$" while the second trajectory is more optimal given the goal. We select the second trajectory and delete the first trajectory from the replay buffer. Note that the "shorter" does not mean fewer actions, but fewer actions that are not related to the task.

Table D.2: **Test sets used for evaluating the proposed model trained on expert data.** We show the count of goal predicate types and the number of goal predicates used in each task.

Test Sets	Predicate Types	#Predicate Per Task	Compared with the training set
In-Distribution	37	$2 \sim 10$	Tasks are drawn from the same distribution as training tasks.
Novel Scenes	37	$2 \sim 10$	The objects are randomly placed in the initial environment.
Novel Tasks	22	$2 \sim 8$	Tasks are never seen during training.

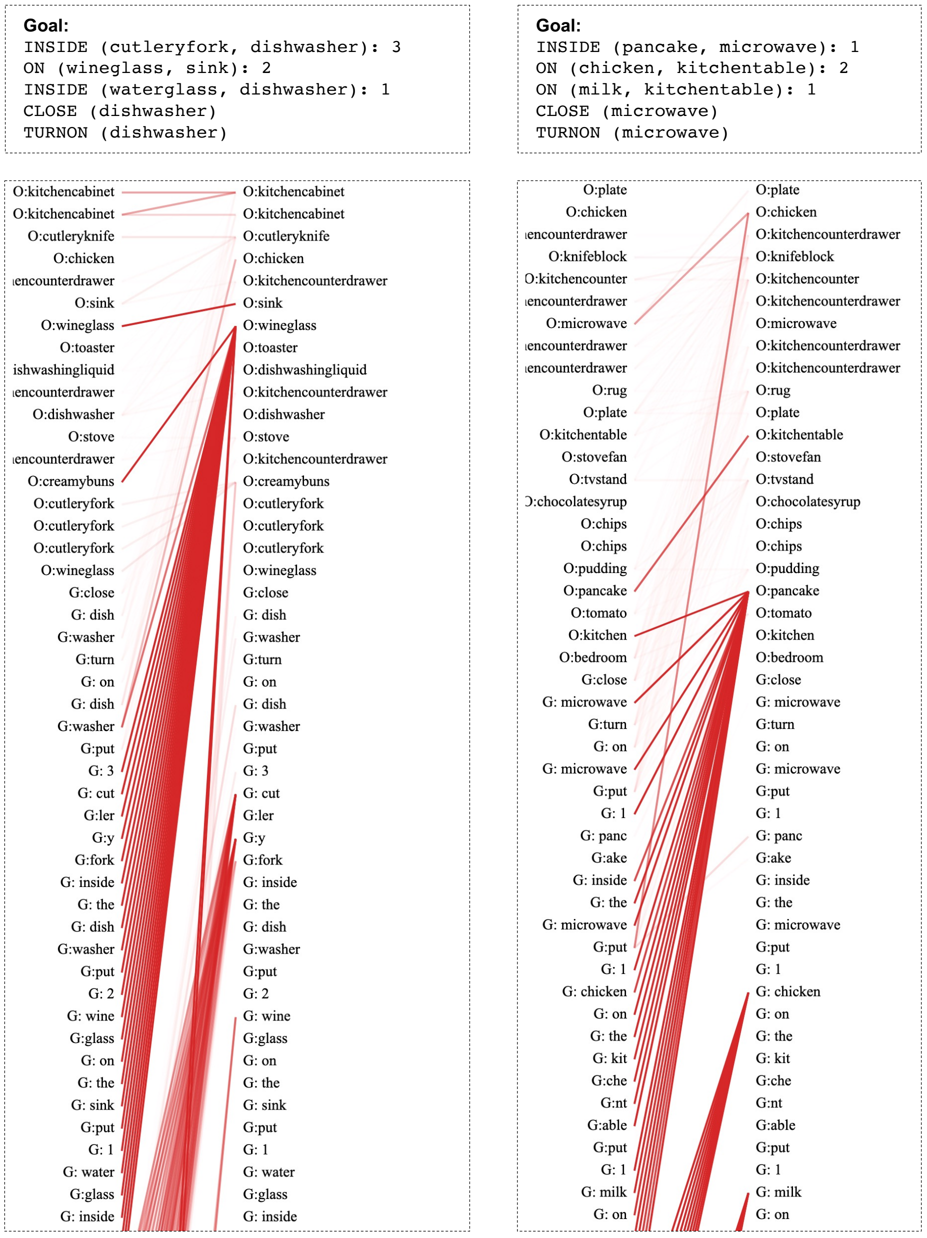
D.7 Visualization of Attention Weights

To better understand how LM pre-trained policies make decisions, we visualize the attention weights from the self-attention layers of GPT-2 [156] in Figure D-6 and Figure D-7. In the inference time, when we are decoding the actions, we save the self-attention weights with respect to different layers and different heads. Then, we use BertViz library [158] to visualize normalized attention weights. We show the attention weights from the input to the output of **PIC-Text (Ours)**. The order of tokens in the input and output is observation, goal, and history. In Figure D-6 and Figure D-7, the left side is the query side. The boldness of the lines is proportional to the attention weight.

Figure D-6 illustrates the attention weights of a layer named "Head 3 Layer 2". We show attention weights on two different tasks. We find that "Head 3 Layer 2" can capture objects in the goal predicates, such as "wineglass" and "cutleryfork" in the left figure, and “pancake” and “chicken” in the right figure (the figures are cropped for visualization).

Figure D-7 illustrates the attention weights of layers named “Head 1 Layer 2” (left) and “Head 4 Layer 11” (right). Given the goal predicates, history, and the current observation, the policy predicts the next action as “grab milk”. We find that “Head 1 Layer 2” is able to capture objects in the goal predicates, such as “milk”, “pancake”, and “chicken” while “Head 4 Layer 11” focuses on the interacted object in the predicted action, such as “milk”.

The attention weights from different self-attention layers are significantly different—some self-attention layers assign high attention weight to objects in the goal predicates while some layers focus on the interacted object. There are also some layers that do not have interpretable meanings. The attention weights just provide us with an intuition of how does the internal language model works, more quantified results are reported in the main text.



```
Goal:
INSIDE (cutleryfork, dishwasher): 3
ON (wineglass, sink): 2
INSIDE (waterglass, dishwasher): 1
CLOSE (dishwasher)
TURNON (dishwasher)
```

```
Goal:
INSIDE (pancake, microwave): 1
ON (chicken, kitchentable): 2
ON (milk, kitchentable): 1
CLOSE (microwave)
TURNON (microwave)
```

Figure D-6: **Attention weights of a layer named "Head 3 Layer 2".** We show attention weights on two different tasks. We find that "Head 3 Layer 2" is able to capture objects in the goal predicates, such as "wineglass" and "cutleryfork" in the left figure, and "pancake" and "chicken" in the right figure (the figures are cropped for visualization).

Goal:
```
INSIDE (pancake, microwave): 1
ON (chicken, kitchentable): 2
ON (milk, kitchentable): 1
CLOSE (microwave)
TURNON (microwave)
```

Action:
```
[grab] <milk>
```

Goal:
```
INSIDE (pancake, microwave): 1
ON (chicken, kitchentable): 2
ON (milk, kitchentable): 1
CLOSE (microwave)
TURNON (microwave)
```

Action:
```
[grab] <milk>
```

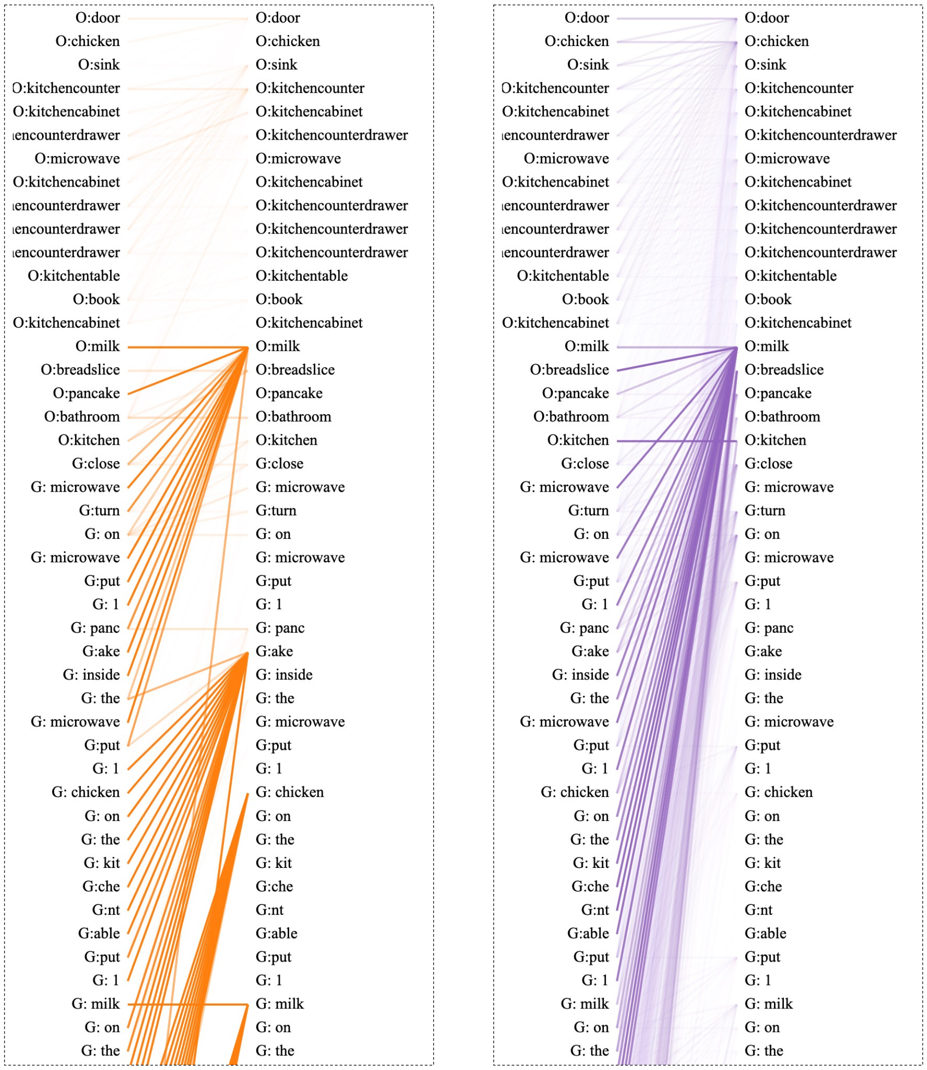


Figure D-7: **Attention weights of layers named "Head 1 Layer 2" (left) and "Head 4 Layer 11" (right).** Given the goal predicates, history, and the current observation, the policy model predicts the next action as "grab milk". We find that "Head 1 Layer 2" can capture objects in the goal predicates, such as "milk", "pancake", and "chicken" while "Head 4 Layer 11" focuses on the interacted object in the predicted action, such as "milk".

Bibliography

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

[3] Prithviraj Ammanabrolu and Mark O Riedl. Playing text-adventure games with graph-based deep reinforcement learning. *arXiv preprint arXiv:1812.01628*, 2018.

[4] Jacob Andreas. Measuring compositionality in representation learning. *arXiv preprint arXiv:1902.07181*, 2019.

[5] Jacob Andreas and Dan Klein. Learning with latent language. In *North American Association for Computational Linguistics*, 2022.

[6] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[7] Mike Lewis Athul Paul Jacob and Jacob Andreas. Multitasking inhibits semantic drift. In *North American Association for Computational Linguistics*, 2021.

[8] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

[9] David Bau, Alex Andonian, Audrey Cui, YeonHwan Park, Ali Jahanian, Aude Oliva, and Antonio Torralba. Paint by word. *arXiv preprint arXiv:2103.10951*, 2021.

[10] Sagie Benaim, Michael Khaitov, Tomer Galanti, and Lior Wolf. Domain intersection and domain difference. In *ICCV*, 2019.

[11] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[12] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[13] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[14] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

[15] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

[16] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[17] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[18] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[19] DCGM. Gender, age, and emotions extracted for flickr-faces-hq dataset (ffhq). `https://github.com/DCGM/ffhq-features-dataset`, 2020.

[20] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[23] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[25] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. *Advances in Neural Information Processing Systems*, 33:6637–6647, 2020.

[26] Yilun Du, Shuang Li, Yash Sharma, Josh Tenenbaum, and Igor Mordatch. Unsupervised learning of compositional energy concepts. *Advances in Neural Information Processing Systems*, 34:15608–15620, 2021.

[27] Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020.

[28] Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020.

[29] Yilun Du, Toru Lin, and Igor Mordatch. Model based planning with energy based models. *CoRL*, 2019.

[30] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[31] Susan T Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

[32] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[33] Alaaeldin El-Nouby, Shikhar Sharma, Hannes Schulz, Devon Hjelm, Layla El Asri, Samira Ebrahimi Kahou, Yoshua Bengio, and Graham W Taylor. Keep drawing it: Iterative language-based image generation and editing. *arXiv preprint arXiv:1811.09845*, 2, 2018.

[34] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[35] Jerry A Fodor and Ernest Lepore. *The compositionality papers*. Oxford University Press, 2002.

[36] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. In *International Conference on Learning Representations*, 2021.

[37] Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

[38] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, and Richard Zemel. Learning the stein discrepancy for training and evaluating energy-based models without sampling. In *International Conference on Machine Learning*, 2020.

[39] Klaus Greff, Raphaël Lopez Kaufmann, Rishab Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. *arXiv preprint arXiv:1903.00450*, 2019.

[40] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pages 1462–1471. PMLR, 2015.

[41] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10696–10706, 2022.

[42] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. *arXiv preprint arXiv:2104.13921*, 2021.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[44] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[45] Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[46] Irina Higgins, Nicolas Sonnerat, Loic Matthey, Arka Pal, Christopher P Burgess, Matko Bosnjak, Murray Shanahan, Matthew Botvinick, Demis Hassabis, and Alexander Lerchner. Scan: Learning hierarchical compositional visual concepts. *ICLR*, 2018.

[47] Felix Hill, Sona Mokra, Nathaniel Wong, and Tim Harley. Human instruction-following with deep reinforcement learning via transfer-learning from text. *arXiv preprint arXiv:2005.09382*, 2020.

[48] Geoffrey E Hinton. Products of experts. *International Conference on Artificial Neural Networks*, 1999.

[49] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[50] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006.

[51] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[52] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[53] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[54] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[55] 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.

[56] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[57] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022.

[58] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[59] David Yu-Tung Hui, Maxime Chevalier-Boisvert, Dzmitry Bahdanau, and Yoshua Bengio. Babyai 1.1, 2020.

[60] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[61] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[62] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1219–1228, 2018.

[63] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[64] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[65] Tero Karras, M. Aittala, Janne Hellsten, S. Laine, J. Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *ArXiv*, abs/2006.06676, 2020.

[66] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020.

[67] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[68] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[69] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[70] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2426–2435, June 2022.

[71] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *ArXiv*, abs/1606.03439, 2016.

[72] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[73] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[74] Nikita Kitaev, Steven Cao, and Dan Klein. Multilingual constituency parsing with self-attention and pre-training. *arXiv preprint arXiv:1812.11760*, 2018.

[75] Richard E Korf. Planning as search: A quantitative approach. *Artificial intelligence*, 33(1):65–88, 1987.

[76] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[77] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.

[78] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

[79] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. In *Predicting structured data*, 2006.

[80] Jie Lei, Tamara L Berg, and Mohit Bansal. Revealing single frame bias for video-and-language learning. *arXiv preprint arXiv:2206.03428*, 2022.

[81] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *International conference on machine learning*, pages 430–438. PMLR, 2016.

[82] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[83] Shuang Li, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, and Igor Mordatch. Composing ensembles of pre-trained models via iterative consensus. *arXiv preprint arXiv:2210.11522*, 2022.

[84] Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022.

[85] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[86] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[87] Nan Liu, Shuang Li, Yilun Du, Josh Tenenbaum, and Antonio Torralba. Learning to compose visual relations. *Advances in Neural Information Processing Systems*, 34:23166–23178, 2021.

[88] 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.

[89] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[90] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[91] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[92] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019.

[93] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

[94] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *European Conference on Computer Vision*, pages 259–274. Springer, 2020.

[95] Gary Marcus, Ernest Davis, and Scott Aaronson. A very preliminary analysis of dall-e 2. *arXiv preprint arXiv:2204.13807*, 2022.

[96] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[97] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[98] Andriy Mnih and Geoffrey Hinton. Learning nonlinear constraints with contrastive backpropagation. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 1302–1307. IEEE, 2005.

[99] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[100] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[101] Ron Mokady, Sagie Benaim, Lior Wolf, and Amit Bermano. Mask based unsupervised content transfer. *CoRR*, abs/1906.06558, 2018.

[102] Ron Mokady, Amir Hertz, and Amit H Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021.

[103] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[104] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

[105] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[106] Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. *Advances in Neural Information Processing Systems*, 34, 2021.

[107] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. *arXiv preprint arXiv:1903.12370*, 2019.

[108] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[109] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[110] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *CoRR*, abs/1802.07569, 2018.

[111] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022.

[112] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[113] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, June 2018.

[114] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[115] Emmanouil Antonios Platanios, Adam Pauls, Subhro Roy, Yuchen Zhang, Alexander Kyte, Alan Guo, Sam Thomson, Jayant Krishnamurthy, Jason Wolfe, Jacob Andreas, et al. Value-agnostic conversational semantic parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3666–3681, 2021.

[116] Ori Press, Tomer Galanti, Sagie Benaim, and Lior Wolf. Emerging disentanglement in auto-encoder based unsupervised image content transfer. In *International Conference on Learning Representations*, 2019.

[117] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

[118] Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Joshua B Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-ai collaboration. *arXiv preprint arXiv:2010.09890*, 2020.

[119] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[120] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[121] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

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

[123] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[124] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[125] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. In *arXiv preprint arXiv:2204.06125*, 2022.

[126] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[127] Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, SM Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas. Few-shot autoregressive density estimation: Towards learning to learn distributions. *arXiv preprint arXiv:1710.10304*, 2017.

[128] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[129] Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*, 2022.

[130] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[131] 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.

[132] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[133] Chitwan Saharia, William Chan, Huiwen Chang, Chris A Lee, Jonathan Ho, Tim Salimans, David J Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. *arXiv preprint arXiv:2111.05826*, 2021.

[134] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[135] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

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

[137] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[138] Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. In *Association for Computational Linguistics*, 2022.

[139] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[140] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Shyamal Buch, Claudia D'Arpino, Sanjana Srivastava, Lyne P Tchapmi, et al. igibson, a simulation environment for interactive tasks in large realistic scenes. *arXiv preprint arXiv:2012.02924*, 2020.

[141] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Shyamal Buch, Claudia D'Arpino, Sanjana Srivastava, Lyne P Tchapmi, et al. igibson, a simulation environment for interactive tasks in large realisticscenes. *arXiv preprint arXiv:2012.02924*, 2020.

[142] Alon Shoshan, Nadav Bhonker, Igor Kviatkovsky, and Gerard Medioni. Gancontrol: Explicitly controllable gans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14083–14093, 2021.

[143] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[144] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[145] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.

[146] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[147] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[148] 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.

[149] Yunfu Song and Zhijian Ou. Learning neural random fields with inclusive auxiliary generators. *arXiv preprint arXiv:1806.00271*, 2018.

[150] Swimmer963. What dall-e 2 can and cannot do, May 2022.

[151] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[152] Yoad Tewel, Yoav Shalev, Idan Schwartz, and Lior Wolf. Zero-shot image-to-text generation for visual-semantic arithmetic. *arXiv preprint arXiv:2111.14447*, 2021.

[153] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[154] Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

[155] Sjoerd van Steenkiste, Karol Kurach, and Sylvain Gelly. A case for object compositionality in deep generative models of images. *arXiv preprint arXiv:1810.10340*, 2018.

[156] 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.

[157] Ramakrishna Vedantam, Ian Fischer, Jonathan Huang, and Kevin Murphy. Generative models of visually grounded imagination. In *ICLR*, 2018.

[158] Jesse Vig. A multiscale visualization of attention in the transformer model. 2019.

[159] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[160] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*, 2019.

[161] Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao. Simvlm: Simple visual language model pretraining with weak supervision. *arXiv preprint arXiv:2108.10904*, 2021.

[162] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

[163] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

[164] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[165] Taihong Xiao, Jiapeng Hong, and Jinwen Ma. Elegant: Exchanging latent encodings with gan for transferring multiple face attributes. In *Proceedings of the European conference on computer vision (ECCV)*, pages 168–184, 2018.

[166] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644. PMLR, 2016.

[167] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1316–1324, 2018.

[168] Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. Just ask: Learning to answer questions from millions of narrated videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1686–1697, 2021.

[169] Zekun Yang, Noa Garcia, Chenhui Chu, Mayu Otani, Yuta Nakashima, and Haruo Takemura. Bert representations for video question answering. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1556–1565, 2020.

[170] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[171] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

[172] Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9127–9134, 2019.

[173] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv:2010.14406*, 2020.

[174] Andy Zeng, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, et al. Socratic models: Composing zero-shot multimodal reasoning with language. In *arXiv preprint arXiv:2204.00598*, 2022.

[175] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[176] Linqi Zhou, Yilun Du, and Jiajun Wu. 3D shape generation and completion through point-voxel diffusion. In *International Conference on Computer Vision*, 2021.

[177] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European conference on computer vision*, pages 592–608. Springer, 2020.

[178] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. Incorporating bert into neural machine translation. *arXiv preprint arXiv:2002.06823*, 2020.