COMPOSITIONAL LAW PARSING WITH LATENT RANDOM FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Human cognition has compositionality. We understand a scene by decomposing the scene into different concepts (e.g., shape and position of an object) and learning the respective laws of these concepts, which may be either natural (e.g., laws of motion) or man-made (e.g., laws of a game). The automatic parsing of these laws indicates the model's ability to understand the scene, which makes law parsing play a central role in many visual tasks. This paper proposes a deep latent variable model for Compositional LAw Parsing (CLAP), which achieves the human-like compositionality ability through an encoding-decoding architecture to represent concepts in the scene as latent variables. CLAP employs concept-specific latent random functions that can be instantiated with Neural Processes to capture the law of concepts. Our experimental results demonstrate that CLAP outperforms the baseline methods in multiple visual tasks such as intuitive physics, abstract visual reasoning, and scene representation. The law manipulation experiments illustrate CLAP's interpretability by modifying specific latent random functions on samples. For example, CLAP learns the laws of position-changing and appearance constancy from the moving balls in a scene, making it possible to exchange laws between samples or compose existing laws into novel laws.

1 INTRODUCTION

Compositionality is an important feature of human cognition (Lake et al., 2017). Humans can decompose a scene into individual concepts to learn the respective laws of these concepts, which can be either natural (e.g. laws of motion) or man-made (e.g. laws of a game). When observing a scene of a moving ball, one tends to parse the changing patterns of its appearance and position separately: The appearance stays consistent over time, while the position changes according to the laws of motion. By composing the laws of the ball's appearance and position, one can understand the changing pattern and predict the status of the moving ball.

Although compositionality has inspired a number of models in visual understanding such as representing handwritten characters through hierarchical decomposition of characters (Lake et al., 2011; 2015) and representing a multi-object scene with object-centric representations (Eslami et al., 2016; Kosiorek et al., 2018; Greff et al., 2019; Locatello et al., 2020), automatic parsing of laws in a scene is still a great challenge. For example, to understand the rules in abstract visual reasoning such as Raven's Progressive Matrices (RPMs) test, the comprehension of attribute-specific representations and the underlying relationships among them is crucial for a model to predict the missing images (Santoro et al., 2018; Steenbrugge et al., 2018; Wu et al., 2020). To understand the laws of motion in intuitive physics, a model needs to grasp the changing patterns of different attributes (e.g. appearance and position) of each object in a scene to predict the future (Agrawal et al., 2016; Kubricht et al., 2017; Ye et al., 2018). However, these methods usually employ neural networks to directly model changing patterns of the scene in a black box, and can hardly abstract the laws of individual concepts in an explicit, interpretable and even manipulatable way.

A possible solution to enable the abovementioned ability for a model is exploiting a function to represent a law of concept in terms of the representation of the concept itself. To represent a law that may depict arbitrary changing patterns, an expressive and flexible random function is required. The Gaussian Process (GP) is a classical family of random functions that achieves the diversity of function space through different kernel functions (Williams & Rasmussen, 2006). Recently pro-

posed random functions (Garnelo et al., 2018b; Eslami et al., 2018; Kumar et al., 2018; Garnelo et al., 2018a; Singh et al., 2019; Kim et al., 2019; Louizos et al., 2019; Lee et al., 2020; Foong et al., 2020) describe function spaces with the powerful nonlinear fitting ability of neural networks. These random functions have been used to capture the changing patterns in a scene, such as mapping timestamps to frames to describe the physical law of moving objects in a video (Kumar et al., 2018; Singh et al., 2019; Fortuin et al., 2020). However, these applications of random functions take images as inputs and the captured laws account for all pixels instead of expected individual concepts.

In this paper, we propose a deep latent variable model for Compositional LAw Parsing (CLAP). CLAP achieves the human-like compositionality ability through an encoding-decoding architecture (Hamrick et al., 2018) to represent concepts in the scene as latent variables, and further employ concept-specific random functions in the latent space to capture the law on each concept. By means of the plug-in of different random functions, CLAP gains generality and flexibility applicable to various law parsing tasks. We introduce CLAP-NP as an example that instantiates latent random functions with recently proposed Neural Processes (Garnelo et al., 2018b).

Our experimental results demonstrate that the proposed CLAP outperforms the compared baseline methods in multiple visual tasks including intuitive physics, abstract visual reasoning, and scene representation. In addition, the experiment on exchanging latent random functions on a specific concept and the experiment on composing latent random functions of the given samples to generate new samples both well demonstrate the *interpretability* and even *manipulability* of the proposed method for compositional law parsing.

2 Related Work

Since CLAP focuses on parsing compositional laws which are modeled by latent random functions, compositional scene representation and random functions are related to this paper.

Compositional Scene Representation Inspired by the compositionality in human cognition, compositional scene representation models understand a scene by object-centric representations. To this end, a class of models initialize object-centric representations first and update them by iterative processes such as Neural Expectation Maximization (Greff et al., 2017), iterative amortized inference (Greff et al., 2019), iterative cross-attention (Locatello et al., 2020), and two-stage inference (Emami et al., 2021). Another class of models extracts object-centric representations from a scene without iterative inference processes. AIR (Eslami et al., 2016) parses object-centric presence, appearance, and position representations as latent variables. SPACE (Lin et al., 2020b) divides a scene image into small regions and parses region-specific representations in parallel. GNM (Jiang & Ahn, 2020) models the layout of scenes. Recently, many models aim at parsing compositional scene representations in videos (Kosiorek et al., 2018; Jiang et al., 2019; Lin et al., 2020a).

Random Functions The GP (Williams & Rasmussen, 2006) is a classical family of random functions (Williams & Rasmussen, 2006) that regards the outputs of a function as a random variable of multivariate Gaussian distribution. GQN (Eslami et al., 2018) represents 3D scenes as functions by a well-designed representation network and generation network. NP (Garnelo et al., 2018b) captures function stochasticity by a Gaussian distributed latent variable instead of a deterministic representation in CNP (Garnelo et al., 2018a). Considering different ways of stochasticity modeling, context aggregation, and output prediction for NP, we can construct function spaces with different characteristics (Kim et al., 2019; Louizos et al., 2019; Singh et al., 2019; Lee et al., 2020; Foong et al., 2020). Some few-shot learning methods parse functions from data by learning global variables (Wu et al., 2018; Gordon et al., 2019) or adaptive kernels (Tossou et al., 2019; Patacchiola et al., 2020). We can also combine latent variable models with ODEs (Norcliffe et al., 2021) or SDEs (Li et al., 2020; Hasan et al., 2021) to describe random functions by the integral over latent states.

3 Preliminaries

A random function describes a distribution over function space $\mathcal{F}$, from which we can sample a function f mapping the input sequence $(\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$ to output sequence $(\boldsymbol{y}_1, ..., \boldsymbol{y}_N)$. The important problem for a random function is finding the underlying function over some given points and predicting function values at novel positions accordingly. In detail, given a set of context points

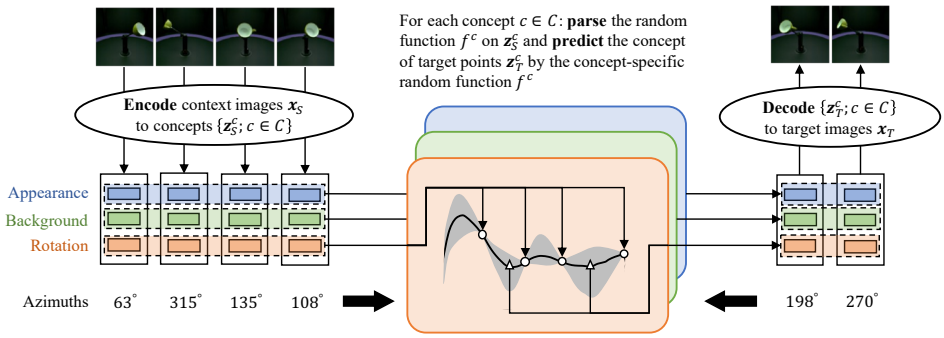


Figure 1: Overview of CLAP: to predict target images, CLAP first encodes given context images into representations of concepts, such as object appearance, background and angle of object rotation; then parses concept-specific latent random functions and computes the representations of concepts in the target images; finally decodes these concepts to compose the target images.

$D_S = \{(\boldsymbol{x}_s, \boldsymbol{y}_s)|s \in S\}$, we should find the most possible function that $\boldsymbol{y}_s = f(\boldsymbol{x}_s)$ for these context points, then use $f(\boldsymbol{x}_t)$ to predict $\boldsymbol{y}_t$ for target points $D_T = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)|t \in T\}$. The abovementioned prediction problem can be regarded as estimating a conditional probability $p(\boldsymbol{y}_T|\boldsymbol{x}_T, D_S)$.

Neural Process (NP) NP (Garnelo et al., 2018b) captures function stochasticity by a Gaussian distributed latent variable $\boldsymbol{g}$ and lets $p(\boldsymbol{y}_T|\boldsymbol{x}_T, D_S) = \int \prod_{t \in T} p(\boldsymbol{y}_t|\boldsymbol{g}, \boldsymbol{x}_t) q(\boldsymbol{g}|D_S) d\boldsymbol{g}$. In this case, $p(\boldsymbol{y}_T|\boldsymbol{x}_T, D_S)$ consists of a generative process $p(\boldsymbol{y}_t|\boldsymbol{g}, \boldsymbol{x}_t)$ and inference process $q(\boldsymbol{g}|D_S)$, which can be jointly optimized by variational inference. To estimate $p(\boldsymbol{y}_T|\boldsymbol{x}_T, D_S)$, NP first extracts the feature of each context point in D_S and sum them up to acquire the mean $\boldsymbol{\mu}$ and deviation $\boldsymbol{\sigma}$ of $\boldsymbol{g}$:

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = \mathcal{T}_f \left(\frac{1}{|S|} \sum_{s \in S} \mathcal{T}_a \left(\boldsymbol{y}_s, \boldsymbol{x}_s \right) \right). \tag{1}$$

Then NP samples the global latent variable $\boldsymbol{g}$ from $q(\boldsymbol{g}|D_S)$. NP can independently predict $\boldsymbol{y}_t$ at the target position $\boldsymbol{x}_t$ by the generative process $p(\boldsymbol{y}_t|\boldsymbol{g}, \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t|\mathcal{T}_p(\boldsymbol{g}, \boldsymbol{x}_t), \sigma^2 \boldsymbol{I})$. Neural networks $\mathcal{T}_a$, $\mathcal{T}_f$ and $\mathcal{T}_p$ increase the model capacity of NP to learn data-specific function spaces.

4 Model

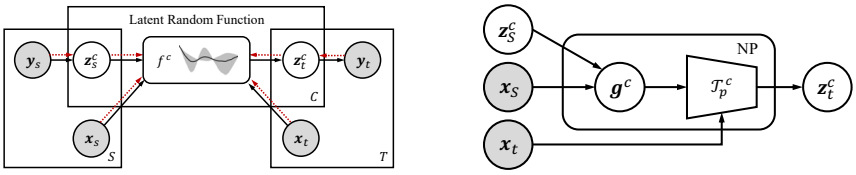
Figure 1 is an overview of CLAP. First, an encoder converts context images to individual concepts in the latent space. Then CLAP parses concept-specific random functions on the context visual concepts, which are exploited to predict the target concepts. Finally, a decoder maps the target concepts to target images. The concept-specific latent random functions achieve the compositional law parsing in CLAP. In the following part, we will introduce the generative process, variational inference, and parameter learning of CLAP. At the end of this section, we propose CLAP-NP, an instance that instantiates the latent random function with NP.

4.1 Latent Random Functions

Given D_S and D_T, the objective of CLAP is to maximize $\log p(\boldsymbol{y}_T|\boldsymbol{y}_S, \boldsymbol{x})$. In the framework of conditional latent variable models, we introduce a variational posterior $q(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{x})$ for the latent variable $\boldsymbol{h}$, and approximate log-likelihood by evidence lower bound (ELBO) (Sohn et al., 2015):

$$\log p\left(\boldsymbol{y}_T|\boldsymbol{y}_S, \boldsymbol{x}\right) \geq \mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{x})}\left[\log p\left(\boldsymbol{y}_T|\boldsymbol{h}\right)\right] - \mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{x})}\left[\log \frac{q\left(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{x}\right)}{p\left(\boldsymbol{h}|\boldsymbol{y}_S, \boldsymbol{x}\right)}\right] = \mathcal{L}. \tag{2}$$

The latent variable $\boldsymbol{h}$ includes the local information of images $\boldsymbol{z} = \{\boldsymbol{z}_n\}_{n=1}^N$ where $\boldsymbol{z}_n$ is the low-dimensional representation of $\boldsymbol{x}_n$. Moreover, we decompose $\boldsymbol{z}_n$ into independent concepts $\{\boldsymbol{z}_n^c|c \in C\}$ (e.g., an image of a single-object scene may have concepts such as object appearance, illumination, camera orientation, etc.) where C refers to the name set of these concepts. Assuming that these concepts satisfy their respective change patterns, we employ independent latent random



(a) CLAP (b) NP as the latent random function

Figure 2: The graphical model of CLAP where the generative process is indicated by black solid lines and the variational inference is indicated by red dotted lines. Panel (a) shows the framework of CLAP, where the latent random function f^c for the concept $c \in C$ can be instantiated by arbitrary random functions; Panel (b) describes how to instantiate the latent random function with NP.

functions to capture the law of each concept. Denote the latent random function on the concept c as f^c, the specific form of f^c depends on the way we model it. As Figure 2b shows, if adopting NP as the latent random function, we have $f^c(\boldsymbol{x}_t) = \mathcal{T}_p^c(\boldsymbol{g}^c, \boldsymbol{x}_t)$ where $\boldsymbol{g}^c$ is a Gaussian-distributed latent variable to control the randomness of f^c. Within the graphical model of CLAP in Figure 2a, the prior and posterior in $\mathcal{L}$ are factorized into

$$\begin{aligned} p(\boldsymbol{h}|\boldsymbol{y}_S, \boldsymbol{x}) &= \prod_{c\in C}\left(p\left(f^c|\boldsymbol{z}_S^c, \boldsymbol{x}_S\right)\prod_{t\in T} p\left(\boldsymbol{z}_t^c|f^c, \boldsymbol{z}_S^c, \boldsymbol{x}_S, \boldsymbol{x}_t\right)\prod_{s\in S} p\left(\boldsymbol{z}_s^c|\boldsymbol{y}_s\right)\right) \\ p\left(\boldsymbol{y}_T|\boldsymbol{h}\right) = \prod_{t\in T} p\left(\boldsymbol{y}_t|\boldsymbol{z}_t\right), &\quad q(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{x}) = \prod_{c\in C}\left(q(f^c|\boldsymbol{z}^c, \boldsymbol{x})\prod_{s\in S} q(\boldsymbol{z}_s^c|\boldsymbol{y}_s)\prod_{t\in T} q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)\right) \end{aligned} \tag{3}$$

Decomposing f into concept-specific laws $\{f^c|c \in C\}$ in the above process is critical for CLAP's compositionality. The compositionality makes it possible to parse only several concept-specific laws rather than the entire law on $\boldsymbol{z}$, which reduces the complexity of law parsing while increasing the interpretability of laws. The interpretability allows us to edit laws, for example, exchanging the law on some concepts and composing the laws of existing samples.

4.2 Parameter Learning

Based on Equation 3, we factorize ELBO as (See Appendix B.3 for the detailed derivation)

$$\begin{aligned} \mathcal{L} = &\underbrace{\sum_{t\in T}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log p\left(\boldsymbol{y}_t|\boldsymbol{z}_t\right)\right]}_{\text{Reconstruction term } \mathcal{L}_r} - \underbrace{\sum_{c\in C}\sum_{t\in T}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log\frac{q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)}{p\left(\boldsymbol{z}_t^c|f^c, \boldsymbol{z}_S^c, \boldsymbol{x}_S, \boldsymbol{x}_t\right)}\right]}_{\text{Target regularizer } \mathcal{L}_t} \\ &- \underbrace{\sum_{c\in C}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log\frac{q(f^c|\boldsymbol{z}^c, \boldsymbol{x})}{p\left(f^c|\boldsymbol{z}_S^c, \boldsymbol{x}_S\right)}\right]}_{\text{Function regularizer } \mathcal{L}_f} - \underbrace{\sum_{c\in C}\sum_{s\in S}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log\frac{q(\boldsymbol{z}_s^c|\boldsymbol{y}_s)}{p\left(\boldsymbol{z}_s^c|\boldsymbol{y}_s\right)}\right]}_{\text{Context regularizer } \mathcal{L}_s} \end{aligned} \tag{4}$$

Reconstruction Term $\mathcal{L}_r$ is the sum of the log-likelihood $\log p(\boldsymbol{y}_t|\boldsymbol{z}_t)$ of each target image. $p(\boldsymbol{y}_t|\boldsymbol{z}_t)$ is modelled by a decoder that converts concepts to the distribution parameter of target images. CLAP maximizes $\mathcal{L}_r$ to connect the latent space with the image space, enabling the decoder to generate high-quality images.

Target Regularizor $\mathcal{L}_t$ consists of Kullback–Leibler (KL) divergences between the posterior $q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)$ and the prior $p(\boldsymbol{z}_t^c|f^c, \boldsymbol{z}_S^c, \boldsymbol{x}_S, \boldsymbol{x}_t)$ of target concepts. In $q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)$, the distribution of target concepts is computed directly from the target image by the encoder, so it usually contains more accurate information. And in $p(\boldsymbol{z}_t^c|f^c, \boldsymbol{z}_S^c, \boldsymbol{x}_S, \boldsymbol{x}_t)$, the distribution of the target concept is estimated from the context. The minimization of $\mathcal{L}_t$ fills the gap between the posterior and prior and ensures the accuracy of predictors.

Function Regularizor $\mathcal{L}_f$ consists of KL divergences between the posterior $q(f^c|\boldsymbol{z}^c, \boldsymbol{x})$ and the prior $p(f^c|\boldsymbol{z}_S^c, \boldsymbol{x}_S)$. We use $\mathcal{L}_f$ as a measure of function consistency to ensure that model can obtain similar distributions for f^c with any subset $\boldsymbol{z}_S^c \subseteq \boldsymbol{z}^c$ and $\boldsymbol{x}_S \subseteq \boldsymbol{x}$ as inputs. To this end, CLAP

shares concept-specific function parsers to model $q(f^c|\boldsymbol{z}^c, \boldsymbol{x})$ and $p(f^c|\boldsymbol{z}_S^c, \boldsymbol{x}_S)$. And we need to design the architecture of function parsers so that they can take different subsets as input.

Context Regularizor $\mathcal{L}_s$ consists of KL divergences between the posterior $q(\boldsymbol{z}_s^c|\boldsymbol{y}_s)$ and the prior $p(\boldsymbol{z}_s^c|\boldsymbol{y}_s)$ of context concepts. To avoid the mismatch between the posterior and prior and reduce model parameters, the posterior and prior are parameterized with the same encoder. In this case, we have $p(\boldsymbol{z}_s^c|\boldsymbol{y}_s) = q(\boldsymbol{z}_s^c|\boldsymbol{y}_s)$ and further $\mathcal{L}_s = 0$. So we remove $\mathcal{L}_s$ from Equation 4.

Correct decomposition of concepts is essential for model performance. However, CLAP is trained without additional supervision to guide the learning of concepts. Although CLAP explicitly defines independent concepts in priors and posteriors, it is not sufficient to automatically decompose concepts for some complex data. To solve the problem, CLAP can introduce extra inductive biases by designing proper encoder and decoder to artificially decompose concepts, e.g., use spatial transformation layers to decompose spatial and non-spatial concepts (Skafte & Hauberg, 2019). We can also set hyperparameters to control the regularizers (Higgins et al., 2017), or add a total correlation (TC) regularizer $\mathcal{R}_{TC}$ to help concept learning (Chen et al., 2018). Here we choose to add extra regularizers to ELBO of CLAP, and finally, the optimization objective becomes

$$\underset{p,q}{\text{argmax}}\ \mathcal{L}^* = \underset{p,q}{\text{argmax}}\ \mathcal{L}_r - \beta_t \mathcal{L}_t - \beta_f \mathcal{L}_f - \beta_{TC} \mathcal{R}_{TC} \tag{5}$$

where β_t, β_f, β_{TC} are hyperparameters that regulate the importance of regularizers in the training process. We can compute $\mathcal{L}^*$ by Stochastic Gradient Variational Bayes (SGVB) estimator (Kingma & Welling, 2013) and update parameters by gradient descent optimizers (Kingma & Ba, 2015).

4.3 NP as Latent Random Function

In this subsection, we propose CLAP-NP as an example that instantiates the latent random function with NP. According to NP, we employ a latent variable $\boldsymbol{g}^c$ to control the randomness of the latent random function f^c for each concept in CLAP-NP.

Generative Process CLAP-NP first encodes each context image $\boldsymbol{x}_s$ to the concepts:

$$\begin{aligned} \{\boldsymbol{\mu}_{z,s}^c | c \in C\} &= \text{Encoder}(\boldsymbol{x}_s), & s \in S, \\ \boldsymbol{z}_s^c &\sim \mathcal{N}\left(\boldsymbol{\mu}_{z,s}^c, \sigma_z^2 \boldsymbol{I}\right), & c \in C, s \in S. \end{aligned} \tag{6}$$

To stabilize the learning of concepts, the encoder outputs only the mean of Gaussian distribution, with the standard deviation as a hyperparameter. Using the process in Equation 1 for each concept, a concept-specific function parser aggregates the contextual information and transform it to the mean $\boldsymbol{\mu}_g^c$ and standard deviation $\boldsymbol{\sigma}_g^c$ of $\boldsymbol{g}^c$. The concept-specific target predictor takes $\boldsymbol{g}^c \sim \mathcal{N}(\boldsymbol{\mu}_g^c, \text{diag}(\boldsymbol{\sigma}_g^2))$ and $\boldsymbol{x}_t$ as inputs and output the mean $\boldsymbol{\mu}_{z,t}^c$ of $\boldsymbol{z}_t$, leaving the standard deviation as a hyperparameter. To keep independence between concepts, CLAP-NP introduces identical but independent function parsers and target predictors. Once all of the concepts $\boldsymbol{z}_t^c \sim \mathcal{N}(\boldsymbol{\mu}_{z,t}^c, \sigma_z^2 \boldsymbol{I})$ for $c \in C$ are generated, we concatenate and decode them into target images:

$$\boldsymbol{x}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_{x,t}, \sigma_x^2 \boldsymbol{I}\right), \quad \boldsymbol{\mu}_{x,t} = \text{Decoder}\left(\{\boldsymbol{z}_t^c | c \in C\}\right), \quad t \in T. \tag{7}$$

To control the noise in sampled images, we introduce a hyperparameter σ_x as the standard deviation. As Figure 2 shows, we can rewrite the prior $p(\boldsymbol{h}|\boldsymbol{y}_S, \boldsymbol{x})$ in CLAP-NP as

$$p(\boldsymbol{h}|\boldsymbol{y}_S, \boldsymbol{x}) = \prod_{c \in C} \left(p\left(\boldsymbol{g}^c | \boldsymbol{z}_S^c, \boldsymbol{x}_S\right) \prod_{t \in T} p\left(\boldsymbol{z}_t^c | \boldsymbol{g}^c, \boldsymbol{x}_t\right) \prod_{s \in S} p\left(\boldsymbol{z}_s^c | \boldsymbol{y}_s\right) \right). \tag{8}$$

Inference and Learning In the variational inference, other than that the randomness of f^c is replaced by $\boldsymbol{g}^c$, the posterior of CLAP-NP is the same as that in Equation 3:

$$q(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{x}) = \prod_{c \in C} \left(q(\boldsymbol{g}^c | \boldsymbol{z}^c, \boldsymbol{x}) \prod_{s \in S} q(\boldsymbol{z}_s^c | \boldsymbol{y}_s) \prod_{t \in T} q(\boldsymbol{z}_t^c | \boldsymbol{y}_t) \right). \tag{9}$$

In the first stage, we compute the mean of both $\boldsymbol{z}_S$ and $\boldsymbol{z}_T$ by the encoder. Because CLAP shares the encoder in the prior and posterior, in the training process we can use the results in Equation 6 of the generative process instead of recalculating the mean of $\boldsymbol{z}_S$. Then we compute the distribution

Table 1: MSEs on BoBa, CRPM, and MPI3D dataset. The training configurations of datasets are given in the brackets, and the test configurations are displayed in table heads.

Model	BoBa-1 (η = 1∼4/12)		BoBa-2 (η = 1∼4/12)		CRPM-DT (η = 1∼2/9)	
	η = 4/12	η = 6/12	η = 4/12	η = 6/12	η = 2/9	η = 3/9
NP	420.7 ± 4.4	649.2 ± 6.0	1169.6 ± 17.4	1917.8 ± 24.8	59.3 ± 4.7	101.9 ± 12.2
GP	1165.5 ± 78.4	2064.9 ± 76.6	2062.6 ± 137.1	3556.8 ± 132.8	263.2 ± 19.0	425.3 ± 22.0
GQN	844.7 ± 36.6	1915.9 ± 34.1	1512.2 ± 22.9	3104.5 ± 40.9	178.3 ± 9.2	379.3 ± 8.0
CLAP-NP	**70.7 ± 2.7**	**151.2 ± 11.1**	**912.0 ± 52.4**	**1833.8 ± 91.9**	**19.6 ± 0.7**	**49.5 ± 10.7**

Model	CRPM-DC (η = 1∼2/9)		MPI3D (η = 1∼4/8)			
	η = 2/9	η = 3/9	η = 4/8	η = 6/8	η = 10/40	η = 20/40
NP	192.6 ± 13.1	323.2 ± 13.6	131.5 ± 3.9	230.5 ± 4.0	274.9 ± 4.2	562.8 ± 6.2
GP	468.3 ± 40.6	764.5 ± 39.0	268.9 ± 13.0	491.6 ± 12.9	279.3 ± 8.2	650.4 ± 12.5
GQN	414.3 ± 7.2	823.0 ± 15.3	**55.6 ± 0.7**	331.8 ± 2.8	772.1 ± 3.4	1148.9 ± 3.2
CLAP-NP	**119.5 ± 17.2**	**264.5 ± 27.9**	58.8 ± 2.3	**153.2 ± 6.3**	**105.5 ± 0.6**	**213.9 ± 1.0**

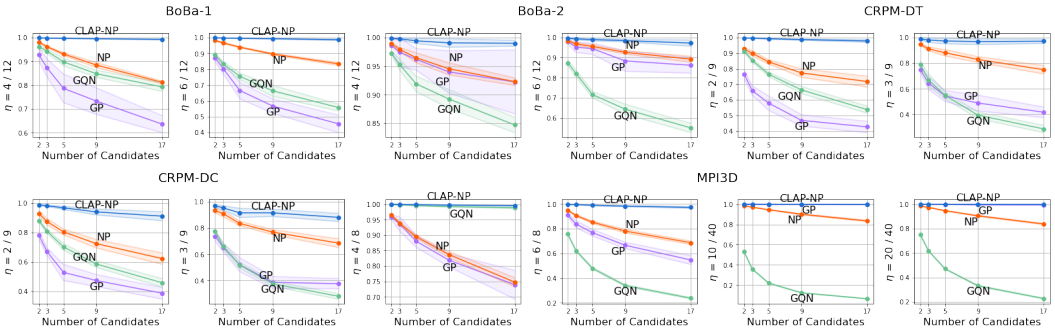


Figure 3: SA scores on BoBa, CRPM, and MPI3D dataset where the blue, orange, purple, and green lines denote scores of CLAP-NP, NP, GP, and GQN.

parameters of $\boldsymbol{g}^c$ by the shared concept-specific function parsers in generative process where the input in the variational inference are $\boldsymbol{z}^c$ and $\boldsymbol{x}$ instead of $\boldsymbol{z}_S^c$ and $\boldsymbol{x}_S$. With the above $p(\boldsymbol{h}|\boldsymbol{y}_S, \boldsymbol{x})$ and $q(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{x})$, some subterms in the ELBO of CLAP-NP become

$$\mathcal{L}_t = \sum_{c\in C}\sum_{t\in T}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log\frac{q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)}{p\left(\boldsymbol{z}_t^c|\boldsymbol{g}^c, \boldsymbol{x}_t\right)}\right], \quad \mathcal{L}_f = \sum_{c\in C}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log\frac{q(\boldsymbol{g}^c|\boldsymbol{z}^c, \boldsymbol{x})}{p\left(\boldsymbol{g}^c|\boldsymbol{z}_S^c, \boldsymbol{x}_S\right)}\right]. \quad (10)$$

5 Experiments

To evaluate the model's ability of compositional law parsing on different types of data, we use three datasets in the experiments: (1) **bouncing ball** (abbreviated as **BoBa**) dataset (Lin et al., 2020a) to validate the ability of intuitive physics; (2) **Continuous Raven's Progressive Matrix (CRPM)** dataset (Shi et al., 2021) to validate the ability of abstract visual reasoning; (3) **MPI3D** dataset (Gondal et al., 2019) to validate the ability of scene representation. We use NP (Garnelo et al., 2018b), GP with the deep kernel (Wilson et al., 2016), and GQN (Eslami et al., 2018) as baselines. We consider NP and GP as baselines because they are representative random functions that construct the function space in different ways, and CLAP-NP instantiates the latent random function with NP. GQN is adopted since it models the distribution of image-valued functions, which highly fits our experimental settings. NP is not used to model image-valued functions, so we add an encoder and decoder to NP's aggregator and predictor to help it handle high-dimensional images. For GP, we use a pretrained encoder and decoder to reduce the dimension of raw images. See Appendix C.1 and C.2 for a detailed introduction to datasets, hyperparameters, and architectures of models.

For quantitative experiments, we take Mean Squared Error (MSE) and Selection Accuracy (SA) as evaluation metrics. Since non-compositional law parsing will increase the risk of producing undesired changes on constant concepts and further cause pixel deviations in predictions, we adopt MSE as an indicator of semantic error. To compute the concept-level deviations between predictions

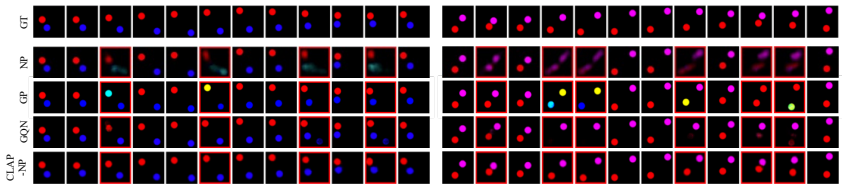


Figure 4: Prediction results (in red boxes) of BoBa-2 with η = 4/12 (left) and η = 6/12 (right).

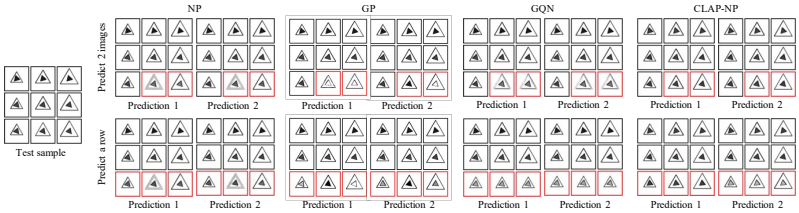


Figure 5: Prediction results (in red boxes) of CRPM-DT with η = 2/9 (top) and η = 3/9 (bottom). We provide two predictions of the same context to illustrate a special case: when a row of images is removed from the given matrix, all predictions with the progressive law of size and constant law of color are correct. In this special case, only CLAP-NP captures such prediction uncertainty.

and ground truths, which we think can be an indicator of semantic error, we introduce SA as another metric to calculate the distance on the representation space. We compute SA by picking out correct ground truths from a set of candidates because the representation spaces vary greatly among models. For a sample $\boldsymbol{x}$, we first combine its ground truth $\boldsymbol{x}_T$ with those from other K samples to establish the candidate set $\{\boldsymbol{d}_k\}_{k=1}^{K+1}$. Then these candidates and the prediction $\tilde{\boldsymbol{x}}_{\boldsymbol{T}} \sim p(\boldsymbol{x}_T|\boldsymbol{x}_S)$ are encoded into representations to calculate concept-level distances $\|\text{Encoder}(\tilde{\boldsymbol{x}}_T) - \text{Encoder}(\boldsymbol{d}_k)\|_2^2$. Finally, the candidate with the smallest distance is chosen, and SA is the selection accuracy across all test samples. In the following experiments, we use $\eta = N_T/N$ to denote the training or test configuration where N_T of N images in each sample are prediction targets.

5.1 Intuitive Physics

To evaluate the intuitive physics of models, which is the underlying knowledge for humans to understand the evolution of the physical world (Kubricht et al., 2017), we use BoBa dataset where physical laws of collisions between colored bouncing balls are described as functions mapping timestamps to frames. BoBa-1 and BoBa-2 are one-ball and two-ball instances of BoBa. When predicting target frames, models should learn both constant laws of appearance and physical laws of collisions. We set up two test configurations η = 4/12 and η = 6/12 for BoBa. The quantitative results in Table 1 and Figure 3 illustrate CLAP-NP's intuitive physics on both BoBa-1 and BoBa-2. Figure 4 visualizes the predictions of BoBa-2 where NP can only generate blurred images while GP and GQN can hardly understand the appearance constancy of balls. That is, balls in the predictions of GP have different colors, and GQN generates non-existent balls and ignores existing balls. Instead, CLAP-NP can separately capture the appearance constancy and collision laws. This benefits the law parsing in scenes with novel appearances and physical laws (Appendix D.7).

5.2 Abstract Visual Reasoning

Abstract visual reasoning is a manifestation of human's fluid intelligence (Cattell, 1963). And RPM (Raven & Court, 1998) is a famous nonverbal test widely used to estimate the model's abstract visual reasoning ability. RPM is a 3×3 image matrix, and the changing rule of the entire matrix consists of multiple attribute-specific subrules. In this experiment, we evaluate models on four instances of CRPM dataset and two test configurations η = 2/9 and η = 3/9. In Table 1 and Figure 3, CLAP-NP achieves the best results with the metrics MSE and SA. Figure 5 displays the predictions on CRPM-DT. In the results, NP produces ambiguous-edged outer triangles, GP predicts targets with incorrect laws, and GQN generates multiple similar images. It is worth stressing that the answer to a CRPM is

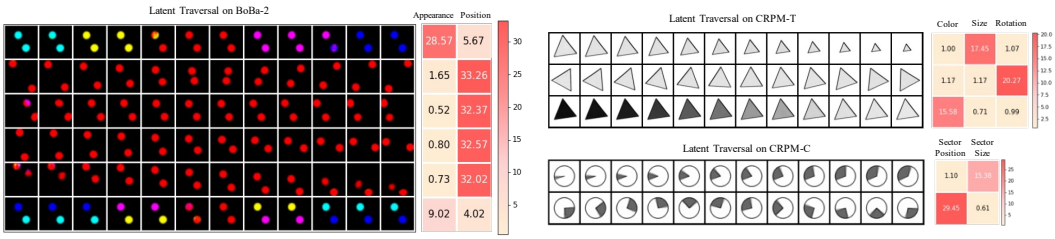


Figure 6: The qualitative latent traversal results and quantitative variance declines on BoBa-2, CRPM-T, and CRPM-C. High variance declines indicate significant correlations between laws and LRFs, by which we can determine whether a dimension encodes the law to edit.

not unique when we remove the entire line of images from it. We display examples of this situation in the second row of Figure 5, indicating that CLAP-NP can understand progressive and constant laws on different attributes rather than simple context-to-target mappings.

5.3 Scene Representation

In scene representation, we use MPI3D dataset where a robotic arm in scenes manipulates the altitude and azimuth of the object to produce images with different laws. The laws of scenes can be regarded as functions that project object altitudes or azimuths to RGB images. In this task, the key to predicting correct target images is to determine which law the scene follows by observing given context images. We train models under η = 1$\sim$4/8 and test them with more target or scene images (η = 4/8, 6/8, 10/40, 20/40). The MSE and SA scores in Table 1 and Figure 3 demonstrate that CLAP-NP outperforms NP in all configurations and GQN in η = 6/8, 10/40, 20/40. CLAP-NP has a slightly higher SA but lower MSE scores than GQN when η = 4/8. A possible reason is that the network of GQN has good inductive biases and more parameters to generate clearer images and achieve better MSE scores. However, such improvement in the pixel-level reconstruction has limited influence on the representation space, as well as the SA scores of GQN. GQN can hardly generalize the random functions to novel situations with more images, leading to performance loss when η = 6/8, 10/40, 20/40. GP's MSE scores on MPI3D indicate that it is suitable for scenes with more context images (e.g., η=10/40). With only sparse points, the prediction uncertainty of GP leads to a significant decrease in performance. However, compositionality allows CLAP-NP to parse concept-specific subrules, which improves the model's generalization ability in novel test configurations.

5.4 Compositionality of Laws

It is possible to exactly editing a law if one could find out which latent random functions (LRFs) encode it. Here we adopt two methods to find the correspondences between laws and LRFs. As Figure 6 shows, we can visualize the traversal in the latent space by perturbing and decoding concepts, and acquire the correspondences by observing changing patterns of these concepts. Another way is to compute variance declines between laws and LRFs (Kim & Mnih, 2018). Assume that there are M types of laws $\{l_m\}_{m=1}^{M}$. For each law, we can sample an instance by $\tilde{l}_m \sim p(l_m)$ and generate a batch of samples $\mathcal{B}_m \sim p(\mathcal{B}_m|\tilde{l}_m)$ that follows the law. First, we generate a batch of samples without fixing any laws and parse LRFs $\{f^c; c \in C\}$ for these samples to estimate the global variances $\mathrm{Var}(f^c)$. Then we generate a series of batches from $p(\mathcal{B}_m|\tilde{l}_m)$ in which all the samples follow the fix law $\tilde{l}_m \sim p(l_m)$ and estimate the in-batch variances $\mathrm{Var}_{\mathcal{B}_m}(f^c)$. Finally the variance decline between l_m and f^c is computed by $s_{m,c} = \mathbb{E}_{p(l_m)}\left[\mathbb{E}_{p(\mathcal{B}_m|l_m)}\left[(\mathrm{Var}(f^c) - \mathrm{Var}_{\mathcal{B}_m}(f^c))/\mathrm{Var}(f^c)\right]\right]$. In Figure 6, we display the variance declines of BoBa-2, CRPM-T, and CRPM-C, which will guide the manipulation of laws in the following experiments.

Latent Random Function Exchange The compositional modeling of laws allows us to swap the law of samples, which is shown at the top of Figure 7. If we want to exchange the law of appearance on two samples of BoBa-2, we first refer to the variance declines and find that LRFs in the first and last dimensions encode the law. Then we infer the LRFs of two samples (in CLAP-NP is to obtain the global latent variables) and swap LRFs of the first and last dimensions. Finally, we execute the

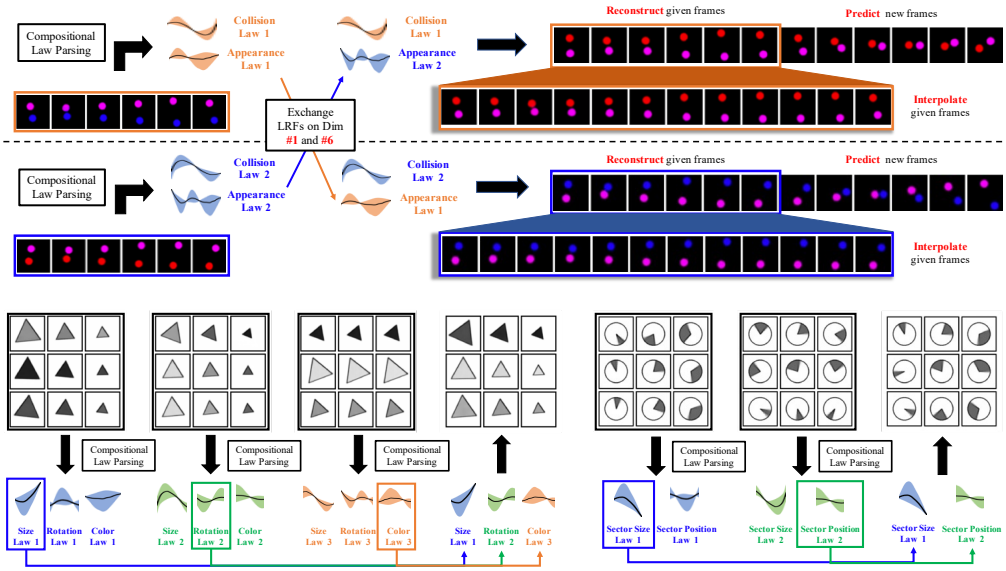


Figure 7: An illustration of law manipulation. At the top, we exchange the constant laws of appearance on samples of BoBa-2 by swapping LRFs on the first and last dimensions. At the bottom, we compose laws from existing samples of CRPM-T and CRPM-C to generate novel samples.

generative process to regenerate the edited samples, and further, make predictions and interpolations on them. Figure 7 visualizes the law manipulation process, which well illustrates the motivation of our model on compositional law parsing.

Latent Random Function Composition For samples of CRPM, we manipulate laws by combining the LRFs from existing samples to generate samples with novel laws. Figure 7 shows the results of law composition on CRPM-T. The first step is parsing LRFs for given samples. By querying the variance declines in Figure 6, we can find these LRFs respectively correspond to the law of size, rotation, and color. Since we have figured out the relations between laws and LRFs, the next step is to pick laws from three different samples respectively and compose them into an unseen mixture of laws. Finally, the mixed laws are decoded to generate a sample with novel changing patterns. This process is similar to the way a human designs RPMs (i.e., sets up separate sub-laws for each attribute and combines them into a complete PRM). From another perspective, law composition provides an option for us to generate novel samples from existing samples (Lake et al., 2015).

6 Conclusion and Limitation

Inspired by the compositionality in human cognition, we propose a deep latent variable model for Compositional LAw Parsing (CLAP). CLAP decomposes high-dimensional images into independent visual concepts in the latent space and employs latent random functions to capture the concept-changing laws, by which CLAP achieves the compositionality of laws. To instantiate CLAP, we propose CLAP-NP that uses NPs as latent random functions. The experimental results demonstrate the benefits of our model in intuitive physics, abstract visual reasoning, and scene representation. Through the experiments on latent random function exchange and composition, we further qualitatively evaluate the interpretability and manipulability of the learned functions in CLAP.

Limitations. (1) **Complexity of datasets.** Because compositional law parsing is an underexplored task, we first validate the effectiveness of SVCL on datasets with relatively clear and simple rules to avoid the influence of unknown confounders in complicated datasets. (2) **Setting the number of LRFs.** For scenes with multiple complex rules, we can empirically set an appropriate upper bound or directly put a large upper bound on the number of LRFs. However, using too many LRFs increases model parameters, and the redundant concepts decrease CLAP's computational efficiency. See Appendix E for detailed limitations and future works.

REFERENCES

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

Raymond B Cattell. Theory of fluid and crystallized intelligence: A critical experiment. *Journal of educational psychology*, 54(1):1, 1963.

Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2610–2620, 2018.

Patrick Emami, Pan He, Sanjay Ranka, and Anand Rangarajan. Efficient iterative amortized inference for learning symmetric and disentangled multi-object representations. In *International Conference on Machine Learning*, pp. 2970–2981. PMLR, 2021.

SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. *Advances in Neural Information Processing Systems*, 29, 2016.

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.

Andrew Foong, Wessel Bruinsma, Jonathan Gordon, Yann Dubois, James Requeima, and Richard Turner. Meta-learning stationary stochastic process prediction with convolutional neural processes. *Advances in Neural Information Processing Systems*, 33:8284–8295, 2020.

Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. Gp-vae: Deep probabilistic time series imputation. In *International conference on artificial intelligence and statistics*, pp. 1651–1661. PMLR, 2020.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. *Advances in Neural Information Processing Systems*, 32, 2019.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2019.

Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. *Advances in Neural Information Processing Systems*, 30, 2017.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pp. 2424–2433. PMLR, 2019.

Jessica B Hamrick, Kelsey R Allen, Victor Bapst, Tina Zhu, Kevin R McKee, Joshua B Tenenbaum, and Peter W Battaglia. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*, 2018.

Ali Hasan, João M Pereira, Sina Farsiu, and Vahid Tarokh. Identifying latent stochastic differential equations. *IEEE Transactions on Signal Processing*, 70:89–104, 2021.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Jindong Jiang and Sungjin Ahn. Generative neurosymbolic machines. *Advances in Neural Information Processing Systems*, 33:12572–12582, 2020.

Jindong Jiang, Sepehr Janghorbani, Gerard De Melo, and Sungjin Ahn. Scalor: Generative world models with scalable object representations. *arXiv preprint arXiv:1910.02384*, 2019.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. *Advances in Neural Information Processing Systems*, 31, 2018.

James R Kubricht, Keith J Holyoak, and Hongjing Lu. Intuitive physics: Current research and controversies. *Trends in cognitive sciences*, 21(10):749–759, 2017.

Ananya Kumar, SM Eslami, Danilo J Rezende, Marta Garnelo, Fabio Viola, Edward Lockhart, and Murray Shanahan. Consistent generative query networks. *arXiv preprint arXiv:1807.02033*, 2018.

Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

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.

Juho Lee, Yoonho Lee, Jungtaek Kim, Eunho Yang, Sung Ju Hwang, and Yee Whye Teh. Bootstrapping neural processes. *Advances in neural information processing systems*, 33:6606–6615, 2020.

Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3870–3882. PMLR, 2020.

Zhixuan Lin, Yi-Fu Wu, Skand Peri, Bofeng Fu, Jindong Jiang, and Sungjin Ahn. Improving generative imagination in object-centric world models. In *International Conference on Machine Learning*, pp. 6140–6149. PMLR, 2020a.

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. *arXiv preprint arXiv:2001.02407*, 2020b.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

Christos Louizos, Xiahan Shi, Klamer Schutte, and Max Welling. The functional neural process. *Advances in Neural Information Processing Systems*, 32, 2019.

Alexander Norcliffe, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Liò. Neural ode processes. In *International Conference on Learning Representations*, 2021.

Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 33:16108–16118, 2020.

John C Raven and John Hugh Court. *Raven's progressive matrices and vocabulary scales*, volume 759. Oxford pyschologists Press Oxford, 1998.

Adam Santoro, Felix Hill, David Barrett, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning*, pp. 4477–4486, 2018.

Fan Shi, Bin Li, and Xiangyang Xue. Raven's progressive matrices completion with latent gaussian process priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9612–9620, 2021.

Gautam Singh, Jaesik Yoon, Youngsung Son, and Sungjin Ahn. Sequential neural processes. *Advances in Neural Information Processing Systems*, 32, 2019.

Nicki Skafte and Søren Hauberg. Explicit disentanglement of appearance and perspective in generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28: 3483–3491, 2015.

Xander Steenbrugge, Sam Leroux, Tim Verbelen, and Bart Dhoedt. Improving generalization for abstract reasoning tasks using disentangled feature representations. *arXiv preprint arXiv:1811.04784*, 2018.

Prudencio Tossou, Basile Dura, Francois Laviolette, Mario Marchand, and Alexandre Lacoste. Adaptive deep kernel learning. *arXiv preprint arXiv:1905.12131*, 2019.

Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

Tailin Wu, John Peurifoy, Isaac L Chuang, and Max Tegmark. Meta-learning autoencoders for few-shot prediction. *arXiv preprint arXiv:1807.09912*, 2018.

Yuhuai Wu, Honghua Dong, Roger Grosse, and Jimmy Ba. The scattering compositional learner: Discovering objects, attributes, relationships in analogical reasoning. *arXiv preprint arXiv:2007.04212*, 2020.

Tian Ye, Xiaolong Wang, James Davidson, and Abhinav Gupta. Interpretable intuitive physics model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 87–102, 2018.

A APPENDIX

In Supplementary Materials, (1) we first provide the details of ELBO, then (2) introduce the datasets, model architecture, hyperparameters, and computational resources adopted in our experiments, and finally (3) provide additional experimental results. In particular, we provide additional results of *editing* and *manipulating* latent random functions, which validate our motivation and contribution.

B DETAILS OF ELBO

B.1 ELBO OF CONDITIONAL LATENT VARIABLE MODELS

$$
\begin{aligned}
\log p\left(\boldsymbol{y}_T|\boldsymbol{y}_S,\boldsymbol{x}\right) &= \int_{\boldsymbol{h}} q\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)\log p\left(\boldsymbol{y}_T|\boldsymbol{y}_S,\boldsymbol{x}\right)d\boldsymbol{h} \\
&= \int_{\boldsymbol{h}} q\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)\log \frac{p\left(\boldsymbol{h},\boldsymbol{y}_T|\boldsymbol{y}_S,\boldsymbol{x}\right)}{p\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)}d\boldsymbol{h} \\
&= \int_{\boldsymbol{h}} q\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)\log \frac{p\left(\boldsymbol{h},\boldsymbol{y}_T|\boldsymbol{y}_S,\boldsymbol{x}\right)}{q\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)}d\boldsymbol{h} + \int_{\boldsymbol{h}} q\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)\log \frac{q\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)}{p\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)}d\boldsymbol{h} \\
&\geq \int_{\boldsymbol{h}} q\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)\log \frac{p\left(\boldsymbol{h},\boldsymbol{y}_T|\boldsymbol{y}_S,\boldsymbol{x}\right)}{q\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)}d\boldsymbol{h} \\
&= \mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log p\left(\boldsymbol{y}_T|\boldsymbol{h}\right)\right] - \mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{q\left(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}\right)}{p\left(\boldsymbol{h}|\boldsymbol{y}_S,\boldsymbol{x}\right)}\right] = \mathcal{L}.
\end{aligned}
\tag{11}
$$

B.2 PRIOR AND POSTERIOR FACTORIZATION

$$
\begin{aligned}
p\left(\boldsymbol{y}_T|\boldsymbol{h}\right) &= \prod_{t\in T} p\left(\boldsymbol{y}_t|\boldsymbol{z}_t\right) \\
p(\boldsymbol{h}|\boldsymbol{y}_S,\boldsymbol{x}) &= \prod_{c\in C} p\left(f^c,\boldsymbol{z}_S^c,\boldsymbol{z}_T^c|\boldsymbol{y}_S,\boldsymbol{x}\right) \\
&= \prod_{c\in C} p\left(\boldsymbol{z}_T^c|f^c,\boldsymbol{z}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_T\right)p\left(f^c|\boldsymbol{z}_S^c,\boldsymbol{x}_S\right)p\left(\boldsymbol{z}_S^c|\boldsymbol{y}_S\right) \\
&= \prod_{c\in C}\left(p\left(f^c|\boldsymbol{z}_S^c,\boldsymbol{x}_S\right)\prod_{t\in T} p\left(\boldsymbol{z}_t^c|f^c,\boldsymbol{z}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_t\right)\prod_{s\in S} p\left(\boldsymbol{z}_s^c|\boldsymbol{y}_s\right)\right)
\end{aligned}
\tag{12}
$$

$$
\begin{aligned}
q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x}) &= \prod_{c\in C} q(f^c,\boldsymbol{z}_S^c,\boldsymbol{z}_T^c|\boldsymbol{y},\boldsymbol{x}) \\
&= \prod_{c\in C} q(f^c|\boldsymbol{z}^c,\boldsymbol{x})q(\boldsymbol{z}_S^c|\boldsymbol{y}_S)q(\boldsymbol{z}_T^c|\boldsymbol{y}_T) \\
&= \prod_{c\in C}\left(q(f^c|\boldsymbol{z}^c,\boldsymbol{x})\prod_{s\in S} q(\boldsymbol{z}_s^c|\boldsymbol{y}_s)\prod_{t\in T} q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)\right)
\end{aligned}
\tag{13}
$$

B.3 ELBO OF CLAP

$$
\begin{aligned}
\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\big[\log \prod_{t\in T} p\left(\boldsymbol{y}_t|\boldsymbol{z}_t\right)\big] \\
&\quad - \mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{\prod_{c\in C}\left(q(f^c|\boldsymbol{z}^c,\boldsymbol{x})\prod_{s\in S} q(\boldsymbol{z}_s^c|\boldsymbol{y}_s)\prod_{t\in T} q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)\right)}{\prod_{c\in C}\left(\prod_{t\in T} p\left(\boldsymbol{z}_t^c|f^c,\boldsymbol{z}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_t\right) p\left(f^c|\boldsymbol{z}_S^c,\boldsymbol{x}_S\right)\prod_{s\in S} p\left(\boldsymbol{z}_s^c|\boldsymbol{y}_s\right)\right)}\right] \\
&= \sum_{t\in T}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\big[\log p\left(\boldsymbol{y}_t|\boldsymbol{z}_t\right)\big] - \sum_{c\in C}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{\prod_{t\in T} q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)}{\prod_{t\in T} p\left(\boldsymbol{z}_t^c|f^c,\boldsymbol{z}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_t\right)}\right] \\
&\quad - \sum_{c\in C}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{q(f^c|\boldsymbol{z}^c,\boldsymbol{x})}{p\left(f^c|\boldsymbol{z}_S^c,\boldsymbol{x}_S\right)}\right] - \sum_{c\in C}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{\prod_{s\in S} q(\boldsymbol{z}_s^c|\boldsymbol{y}_s)}{\prod_{s\in S} p\left(\boldsymbol{z}_s^c|\boldsymbol{y}_s\right)}\right] \\
&= \underbrace{\sum_{t\in T}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\big[\log p\left(\boldsymbol{y}_t|\boldsymbol{z}_t\right)\big]}_{\text{Reconstruction term } \mathcal{L}_r} - \underbrace{\sum_{c\in C}\sum_{t\in T}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)}{p\left(\boldsymbol{z}_t^c|f^c,\boldsymbol{z}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_t\right)}\right]}_{\text{Target regularizer } \mathcal{L}_t} \\
&\quad - \underbrace{\sum_{c\in C}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{q(f^c|\boldsymbol{z}^c,\boldsymbol{x})}{p\left(f^c|\boldsymbol{z}_S^c,\boldsymbol{x}_S\right)}\right]}_{\text{Function regularizer } \mathcal{L}_f} - \underbrace{\sum_{c\in C}\sum_{s\in S}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}_s^c|\boldsymbol{y}_s)}{p\left(\boldsymbol{z}_s^c|\boldsymbol{y}_s\right)}\right]}_{\text{Context regularizer } \mathcal{L}_s}
\end{aligned}
\tag{14}
$$

B.3.1 RECONSTRUCTION TERM

$$
\begin{aligned}
\mathcal{L}_r &= \sum_{t\in T}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\big[\log p\left(\boldsymbol{y}_t|\boldsymbol{z}_t\right)\big] = \sum_{t\in T}\mathbb{E}_{q(\boldsymbol{z}_t|\boldsymbol{y}_t)}\big[\log p\left(\boldsymbol{y}_t|\boldsymbol{z}_t\right)\big] \\
&\approx \sum_{t\in T}\log p\left(\boldsymbol{y}_t|\tilde{\boldsymbol{z}}_t\right), \quad \text{where } \tilde{\boldsymbol{z}}_t \sim q(\boldsymbol{z}_t|\boldsymbol{y}_t)
\end{aligned}
\tag{15}
$$

B.3.2 TARGET REGULARIZER

$$
\begin{aligned}
\mathcal{L}_t &= \sum_{c\in C}\sum_{t\in T}\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)}{p\left(\boldsymbol{z}_t^c|f^c,\boldsymbol{z}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_t\right)}\right] \\
&= \sum_{c\in C}\sum_{t\in T}\mathbb{E}_{q(\boldsymbol{z}_S^c|\boldsymbol{y}_S)}\left[\mathbb{E}_{q(\boldsymbol{z}_T^c|\boldsymbol{y}_T)}\left[\mathbb{E}_{q(f^c|\boldsymbol{z}^c,\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)}{p\left(\boldsymbol{z}_t^c|f^c,\boldsymbol{z}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_t\right)}\right]\right]\right].
\end{aligned}
\tag{16}
$$

Because of the function consistency in samples, function posteriors $q(f^c|\tilde{\boldsymbol{z}}^c,\boldsymbol{x})$ derived from arbitrary sampled $\tilde{\boldsymbol{z}}^c \sim q(\boldsymbol{z}^c|\boldsymbol{y})$ need to be consistent. So we let f^c condition on only the observations $\boldsymbol{x}$ and $\boldsymbol{y}$ to simplify the posterior distribution, that is, we replace $q(f^c|\tilde{\boldsymbol{z}}^c,\boldsymbol{x})$ with $q(f^c|\boldsymbol{y},\boldsymbol{x})$:

$$
\begin{aligned}
\mathcal{L}_t &\approx \sum_{c\in C}\sum_{t\in T}\mathbb{E}_{q(\boldsymbol{z}_S^c|\boldsymbol{y}_S)}\left[\mathbb{E}_{q(\boldsymbol{z}_T^c|\boldsymbol{y}_T)}\left[\mathbb{E}_{q(f^c|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)}{p\left(\boldsymbol{z}_t^c|f^c,\boldsymbol{z}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_t\right)}\right]\right]\right] \\
&= \sum_{c\in C}\sum_{t\in T}\mathbb{E}_{q(\boldsymbol{z}_S^c|\boldsymbol{y}_S)}\Big[\mathbb{E}_{q(f^c|\boldsymbol{y},\boldsymbol{x})}\Big[\mathrm{KL}\big(q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)\|p\left(\boldsymbol{z}_t^c|f^c,\boldsymbol{z}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_t\right)\big)\Big]\Big].
\end{aligned}
\tag{17}
$$

In this way, we convert the computation of the log-likelihoods on $q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)$ and $p(\boldsymbol{z}_t^c|f^c,\boldsymbol{z}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_t)$ to the KL divergences between them. Then a Monte Carlo estimator can be used to approximate $\mathcal{L}_t$ where $\tilde{\boldsymbol{z}}_S^c$ is sampled by $\tilde{\boldsymbol{z}}_S^c \sim q(\boldsymbol{z}_S^c|\boldsymbol{y}_S)$ to compute the outer expectation $\mathbb{E}_{q(\boldsymbol{z}_S^c|\boldsymbol{y}_S)}[*]$; and for inner expectation $\mathbb{E}_{q(f^c|\boldsymbol{y},\boldsymbol{x})}[*]$, because $q(f^c|\boldsymbol{y},\boldsymbol{x}) = \int q(f^c|\boldsymbol{z}^c,\boldsymbol{x})q(\boldsymbol{z}^c|\boldsymbol{x})d\boldsymbol{z}^c$, we are able to first sample $\tilde{\boldsymbol{z}}_T^c \sim q(\boldsymbol{z}_T^c|\boldsymbol{y}_T)$ and obtain $\tilde{f}^c \sim q(f^c|\tilde{\boldsymbol{z}}_S^c,\tilde{\boldsymbol{z}}_T^c,\boldsymbol{x})$. By means of $\tilde{f}^c$ and $\tilde{\boldsymbol{z}}^c$ sampled from the posterior, we have $\mathcal{L}_t \approx \mathrm{KL}(q(\boldsymbol{z}_t^c|\boldsymbol{y}_t)\|p(\boldsymbol{z}_t^c|\tilde{f}^c,\tilde{\boldsymbol{z}}_S^c,\boldsymbol{x}_S,\boldsymbol{x}_t))$.

B.3.3 FUNCTION REGULARIZER

$$\begin{aligned}\mathcal{L}_f &= \sum_{c\in C} \mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{q(f^c|\boldsymbol{z}^c,\boldsymbol{x})}{p\left(f^c|\boldsymbol{z}^c_S,\boldsymbol{x}_S\right)}\right] \\ &= \sum_{c\in C} \mathbb{E}_{q(\boldsymbol{z}^c_S|\boldsymbol{y}_S)}\left[\mathbb{E}_{q(\boldsymbol{z}^c_T|\boldsymbol{y}_T)}\left[\mathbb{E}_{q(f^c|\boldsymbol{z}^c,\boldsymbol{x})}\left[\log \frac{q(f^c|\boldsymbol{z}^c,\boldsymbol{x})}{p\left(f^c|\boldsymbol{z}^c_S,\boldsymbol{x}_S\right)}\right]\right]\right] \\ &= \sum_{c\in C} \mathbb{E}_{q(\boldsymbol{z}^c|\boldsymbol{y})}\Big[\mathrm{KL}\big(q(f^c|\boldsymbol{z}^c,\boldsymbol{x})\|p\left(f^c|\boldsymbol{z}^c_S,\boldsymbol{x}_S\right)\big)\Big].\end{aligned} \tag{18}$$

To estimate $\mathcal{L}_f$ in the same way as the target regularizer, $\tilde{\boldsymbol{z}}^c$ is sampled by $\tilde{\boldsymbol{z}}^c \sim q(\boldsymbol{z}^c|\boldsymbol{y})$ and we have $\mathcal{L}_f \approx \mathrm{KL}(q(f^c|\tilde{\boldsymbol{z}}^c,\boldsymbol{x})\|p\left(f^c|\tilde{\boldsymbol{z}}^c_S,\boldsymbol{x}_S\right))$.

B.3.4 CONTEXT REGULARIZER

$$\mathcal{L}_s = \sum_{c\in C}\sum_{s\in S} \mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}^c_s|\boldsymbol{y}_s)}{p\left(\boldsymbol{z}^c_s|\boldsymbol{y}_s\right)}\right] = \sum_{c\in C}\sum_{s\in S} \mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{y},\boldsymbol{x})}\big[\log 1\big] = 0. \tag{19}$$

B.4 TOTAL CORRELATION

Let $\mathcal{I} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_K\}$ be the set with images from all samples in the dataset, and $\tilde{\boldsymbol{z}}_k \sim q(\boldsymbol{z}_k|\boldsymbol{y}_k)$ are the sampled latent variables for each image. To apply the Total Correlation (TC) of β-TCVAE (Chen et al., 2018) to CLAP, we make a tiny modification that the aggregated posterior $\bar{q}(\boldsymbol{z}) = \sum_{k=1}^{K} q(\boldsymbol{z}_k|\boldsymbol{y}_k)p(\boldsymbol{y}_k)$ is decomposed along with concepts instead of the dimension of latent variables. According to β-TCVAE, there is

$$\mathcal{R}_{TC} = \mathrm{KL}\left(\bar{q}(\boldsymbol{z})\Big\|\prod_{c\in C}\bar{q}(\boldsymbol{z}^c)\right) = \mathbb{E}_{\bar{q}(\boldsymbol{z})}\big[\log \bar{q}(\boldsymbol{z})\big] - \sum_{c\in C}\mathbb{E}_{\bar{q}(\boldsymbol{z})}\big[\log \bar{q}(\boldsymbol{z}^c)\big]. \tag{20}$$

where β-TCVAE provides Minibatch Weighted Sampling (MWS) to approximate Equation 20 by a minibatch $\{\boldsymbol{y}_m|m \in M\} \subseteq \mathcal{I}$ of the dataset:

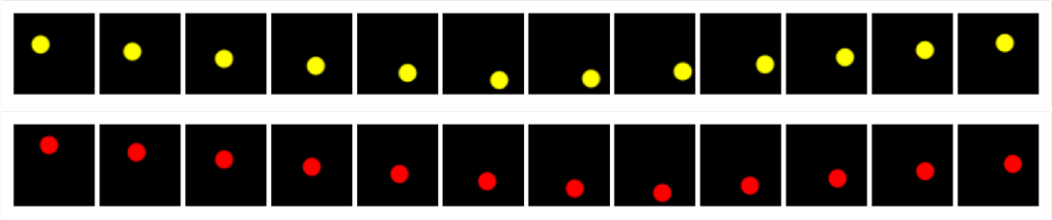
$$\begin{aligned}\mathbb{E}_{\bar{q}(\boldsymbol{z})}\big[\log \bar{q}(\boldsymbol{z})\big] &\approx \frac{1}{|M|}\sum_{i\in M}\left[\log \sum_{j\in M} q\left(\tilde{\boldsymbol{z}}_i|\boldsymbol{y}_j\right) - \log(K|M|)\right] \\ \mathbb{E}_{\bar{q}(\boldsymbol{z})}\big[\log \bar{q}(\boldsymbol{z}^c)\big] &\approx \frac{1}{|M|}\sum_{i\in M}\left[\log \sum_{j\in M} q\left(\tilde{\boldsymbol{z}}^c_i|\boldsymbol{y}_j\right) - \log(K|M|)\right].\end{aligned} \tag{21}$$

C DATASETS AND EXPERIMENTAL SETUP

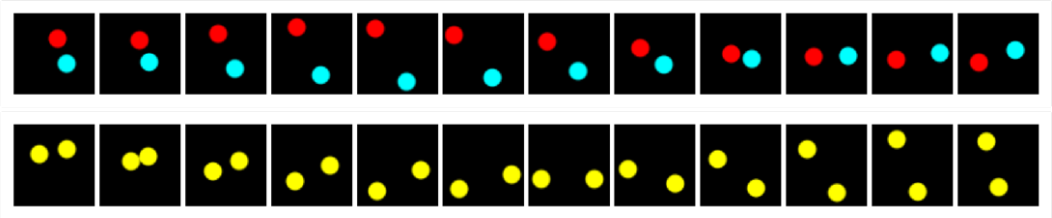
C.1 DETAILS OF DATASETS

In this paper, three types of datasets are adopted: BoBa, CRPM, and MPI3D. Figure 8 displays two samples for each instance of the datasets, and Table 2 describes the train, validation, and test configurations of the instances.

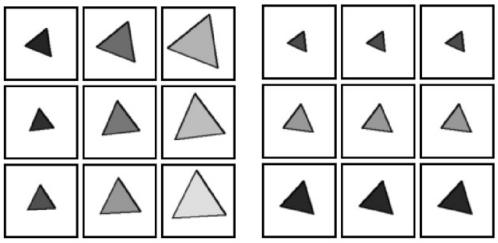
Bouncing Ball (BoBa) BoBa contains many videos of bouncing balls. Depending on the number of balls in the scene, the dataset provides two instances BoBa-1 and BoBa-2. In the videos of Figure 8a and 8b, the motion of balls follows the law of physical collisions; and the appearance of balls (color, size, amount, etc.) is constant over time. The models need to capture the probability space on functions $f : \mathbb{R} \mapsto \mathbb{R}^{3\times 64\times 64}$ that map timestamps to video frames. Referring to Table 2, we provide 10,000 videos of bouncing balls for training, 1,000 for validation and hyperparameter selection, and 2,000 to test the intuitive physics of the models. In the training and validation phase, we randomly select 1 to 4 frames from video frames as the target images to predict; in the testing phase, we provide two configurations (referred to as Test-4 and Test-6) to evaluate the model's performance when there are 4 and 6 target images in videos.



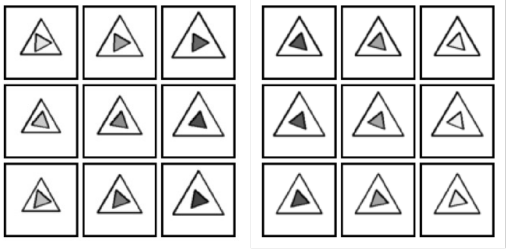
(a) BoBa-1



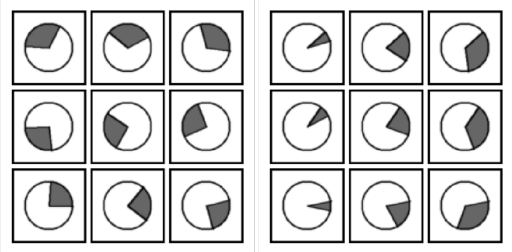
(b) BoBa-2



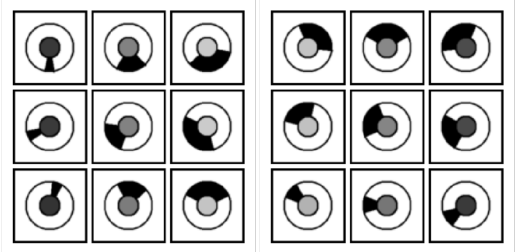
(c) CRPM-T



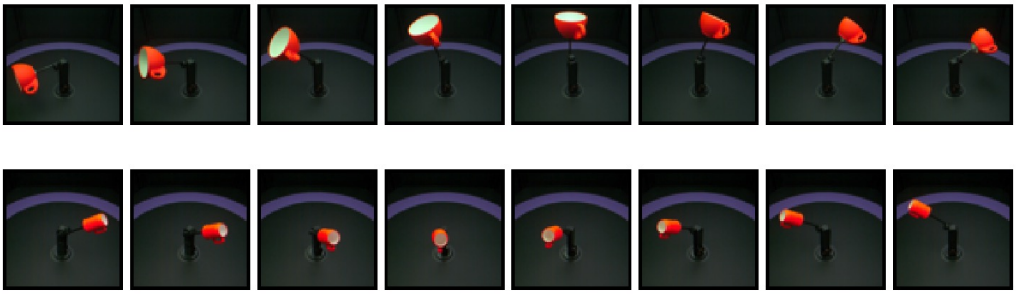
(d) CRPM-DT



(e) CRPM-C



(f) CRPM-DC



(g) MPI3D

Figure 8: Examples from the different instances of datasets. BoBa includes two instances (a) BoBa-1 and (b) BoBa-2; CRPM includes four instances (c) CRPM-T, (d) CRPM-DT, (e) CRPM-C, and (f) CRPM-DC; (g) we only use one instance of MPI3D.

Table 2: Details of BoBa, CRPM, and MPI3D dataset. Row 1: dataset names. Row 2: splits of datasets where the number k in the suffix of Test-k denotes the number of target images. Row 3: number of samples in each split. Row 4: number of images in each sample. Row 5: number of target images in a sample. Row 6: image size represented by Channel $\times$ Height $\times$ Width.

Dataset	BoBa-1				BoBa-2			
Split	Train	Valid	Test-4	Test-6	Train	Valid	Test-4	Test-6
Samples	10000	1000	2000	2000	10000	1000	2000	2000
Images	12				12			
Targets	$1 \sim 4$	$1 \sim 4$	4	6	$1 \sim 4$	$1 \sim 4$	4	6
Image Size	$3 \times 64 \times 64$				$3 \times 64 \times 64$			
Dataset	CRPM-T				CRPM-C			
Split	Train	Valid	Test-2	Test-3	Train	Valid	Test-2	Test-3
Samples	10000	1000	2000	2000	10000	1000	2000	2000
Images	9				9			
Targets	$1 \sim 2$	$1 \sim 2$	2	3	$1 \sim 2$	$1 \sim 2$	2	3
Image Size	64×64				64×64			
Dataset	CRPM-DT				CRPM-DC			
Split	Train	Valid	Test-2	Test-3	Train	Valid	Test-2	Test-3
Samples	10000	1000	2000	2000	10000	1000	2000	2000
Images	9				9			
Targets	$1 \sim 2$	$1 \sim 2$	2	3	$1 \sim 2$	$1 \sim 2$	2	3
Image Size	64×64				64×64			

Dataset	MPI3D					
Split	Train	Valid	Test-4	Test-6	Test-10	Test-20
Samples	16127	2305	4608	4608	4608	4608
Images	8				40	
Targets	$1 \sim 4$	$1 \sim 4$	4	6	10	20
Image Size	$3 \times 64 \times 64$					

Table 3: A detailed description of $\boldsymbol{x}_n$ on each dataset.

Dataset	Description
CRPM	2D index of the image in matrix
BoBa	timestamps of the frame
MPI3D	azimuths or altitudes of the center object

Continuous Raven's Progressive Matrix (CRPM) CRPM consists of 3×3 image matrices where the images contain one or two centered triangles or circles. CRPM provides four instances with different image types: CRPM-T, CRPM-DT, CRPM-C, and CRPM-DC. Images in a matrix follow attribute-specific changing rules (e.g., rules in the size, grayscale, and rotation of the triangle). The first sample in Figure 8c shows that, for images within the row, the sizes of triangles increase progressively, the grayscales change progressively from dark to light, and the rotations keep constant. To predict the missing target images in the matrix, the models need to learn the probability space on functions $f : \{-1, 0, 1\}^2 \mapsto \mathbb{R}^{64\times 64}$ that map 2D grid coordinates to grid images. Table 2 shows that we provide 10,000 image matrices for training, 1,000 for tuning the model hyperparameters, and 2,000 to test the abstract visual reasoning ability of models. In the training and validation

Table 4: Hyperparameters of CLAP-NP.

Dataset	$\|C\|$	β_t	β_f	β_{TC}	σ_z	d_g
BoBa-1	3	100	100	1000	0.03	32
BoBa-2	6	400	100	1500	0.03	64
CRPM-T	3	400	100	5000	0.1	64
CRPM-DT	3	200	50	5000	0.1	64
CRPM-C	2	100	100	5000	0.1	64
CRPM-DC	3	300	300	10000	0.1	64
MPI3D	7	50	50	400	0.03	64

process, we randomly select 1 or 2 images as target images; in the testing process, we use Test-2 and Test-3 to evaluate the performance of the models to predict 2 and 3 target images.

MPI3D MPI3D (Gondal et al., 2019) dataset contains a series of single-object scenes, each with 40 different scene images. There are two underlying rules to change the object in a scene: change the altitude of the object (sample 1 in Figure 8g); change the azimuth of the object (sample 2 in Figure 8g). The other attributes (e.g., object color, camera height, etc.) do not change within scene images. It is essential for the models to grasp different changing patterns for target image prediction, and the key is to describe the distribution over functions $f : \mathbb{R} \mapsto \mathbb{R}^{3\times64\times64}$ that map object altitudes or azimuths to scene images. As Table 2 shows, we provide 16127 scenes for training, 2305 for tuning the hyperparameters, and 4608 for testing. In terms of training and validation, we first randomly select 8 images from 40 scene images to represent the scene, and then randomly select 1 to 4 images from the 8 images again as the target images, leaving the remaining images as the context. For testing, we supply Test-4, Test-6, Test-10, and Test-20, where Test-4 and Test-6 evaluate the performance of the models to predict 4 and 6 target images out of 8 images; Test-10 and Test-20 evaluate the performance to predict 10 and 20 target images out of 40 images. One can fetch MPI3D from the repository [1] under the Creative Commons Attribution 4.0 International License.

C.2 Model Architecture and Hyperparameters

CLAP-NP In this subsection, we will first describe the architecture of the encoder, decoder, concept-specific function parsers, and concept-specific target predictors network in CLAP-NP. Then we will list hyperparameters in the training phase.

- **Encoder**: for all datasets, CLAP-NP uses the same convolutional blocks to downsample high-dimensional images into parameters of concepts, which can be described as
  - 4×4 Conv, stride 2, padding 1, 32 BatchNorm, ReLU
  - 4×4 Conv, stride 2, padding 1, 32 BatchNorm, ReLU
  - 4×4 Conv, stride 2, padding 1, 64 BatchNorm, ReLU
  - 4×4 Conv, stride 2, padding 1, 128 BatchNorm, ReLU
  - 4×4 Conv, 256 BatchNorm, ReLU
  - 1×1 Conv, 512
  - ReshapeBlock, 512
  - Fully Connected, $|C|$

  The ReshapeBlock flattens the output tensor of size $512 \times 1 \times 1$ from the last convolutional layer to the vector of size 512. The output of the encoder is the mean of $|C|$ visual concepts.
- **Decoder**: in BoBa and CRPM datasets, CLAP-NP uses several deconvolutional layers to up-sample concepts to images, which is
  - ReshapeBlock, $|C| \times 1 \times 1$
  - 1×1 Deconv, 128 BatchNorm, ReLU
  - 4×4 Deconv, 64 BatchNorm2d, ReLU
  - 4×4 Deconv, stride 2, padding 1, 32 BatchNorm, ReLU

[1] https://github.com/rr-learning/disentanglement_dataset

Table 5: Hyperparameters of NP.

Dataset	d_r	d_g
BoBa-1	3	1024
BoBa-2	6	1024
CRPM-T	3	1024
CRPM-DT	3	1024
CRPM-C	2	512
CRPM-DC	3	512
MPI3D	7	1024

  - 4×4 Deconv, stride 2, padding 1, 32 BatchNorm, ReLU
  - 4×4 Deconv, stride 2, padding 1, 32 BatchNorm, ReLU
  - 4×4 Deconv, stride 2, padding 1, N_{Channel} Sigmoid

  where N_{Channel} is the number of image channels and Sigmoid is the activation function used to generate pixel values ranging in $(0, 1)$.

- **Function Parser**: CLAP-NP provides identical but independent function parsers for each concept. Each function parser consists of a multi-layer perception (MLP) $\mathcal{T}_s$ to encode concepts, a MLP $\mathcal{T}_i$ to encode function inputs, a MLP $\mathcal{T}_a$ to aggregate context points, and a MLP $\mathcal{T}_f$ to predict the mean and standard deviation of the global latent variable. The architecture of $\mathcal{T}_s$ and $\mathcal{T}_i$ is
  - Fully Connected, 32 ReLU
  - Fully Connected, 16

  By concatenating the encoded concepts and the encoded function inputs, we get the context function points. The architecture of $\mathcal{T}_a$ is
  - Fully Connected, 256 ReLU
  - Fully Connected, 256 ReLU
  - Fully Connected, 128

  Finally, the architecture of $\mathcal{T}_f$ is
  - Fully Connected, 256 ReLU
  - Fully Connected, $2d_g$

  where d_g is the size of the global latent variable $\boldsymbol{g}^c$. The outputs of $\mathcal{T}_f$ consist of the mean and standard deviation of the latent variable $\boldsymbol{g}^c$.

- **Target Predictor**: the concept-specific target predictor $\mathcal{T}_p$ is a MLP that maps $\boldsymbol{g}^c$ and the encoded function input into the target concept, whose architecture is
  - Fully Connected, 256 ReLU
  - Fully Connected, 256 ReLU
  - Fully Connected, 1

To generate more complex scene images in MPI3D, CLAP-NP uses a deeper decoder where the hidden size is [512, 256, 128, 64, 32]. And we use $\mathcal{T}_f$ with the hidden size [128, 128] and output size 64, $\mathcal{T}_f$ with the hidden size [64], and $\mathcal{T}_p$ with the hidden size [128, 128]. Hyperparameters for CLAP-NP on different datasets are shown in Table 4. And in all datasets, we set learning rate as 3×10^{-4}, batch size as 512, and $\sigma_x = 0.1$. For MPI3D, we anneal β_t and β_f in the first 400 epochs. After each training epoch, we use the validation set to compute the evidence lower bound (ELBO) of the current model and save the model with the largest ELBO as the trained model. For all datasets, CLAP-NP uses the Adam (Kingma & Ba, 2015) optimizer to update parameters.

Neural Process (NP) Different from (Garnelo et al., 2018b), to apply NP in high-dimensional images, we add an encoder and decoder which are the same as those in CLAP-NP to convert high-dimensional images into low-dimensional representations. And the representations are combined with the function inputs to form the context by an aggregator $\mathcal{T}_a$. Then the function parser $\mathcal{T}_f$ obtains the mean and standard deviation of the global latent variable. Finally, the decoder generates

Table 6: Learning rate and epoch in GP to adjust hyperparameters.

Dataset	learning rate	epoch
BoBa-1	0.05	200
BoBa-2	0.05	200
CRPM-T	0.05	200
CRPM-DT	0.05	100
CRPM-C	0.01	400
CRPM-DC	0.05	100
MPI3D	0.05	10

target images from the target function inputs and the global latent variable. The architectures of $\mathcal{T}_a$ and $\mathcal{T}_f$ are

- **Aggregator**
  - Fully Connected, 512 ReLU
  - Fully Connected, 512 ReLU
  - Fully Connected, 512
- **Function Parser**
  - Fully Connected, 512 ReLU
  - Fully Connected, 512 ReLU
  - Fully Connected, $2d_g$

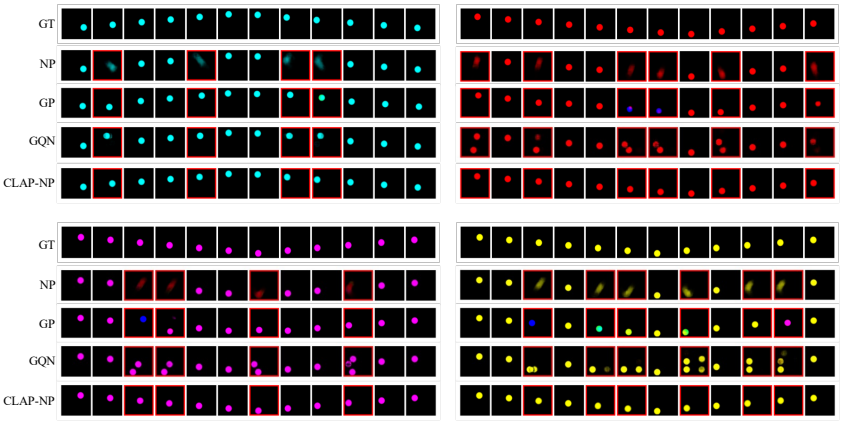
Hyperparameters for NP on different datasets are shown in Table 5 where d_r is the representation size of the encoder. And for all datasets, NP adopts the learning rate 1×10^{-4} batch size 512, and $\sigma_x = 0.1$. For all datasets, NP uses the Adam optimizer for parameter updates.

Generative Query Network (GQN) To implement GQN (Eslami et al., 2018), we use a PyTorch implementation from the repository [2] with the following changes to the default configuration: set learning rate to 0.0005, batch size to 256, the representation type to pool, the number of iterations to 4, and share the ConvLSTM core among iterations.

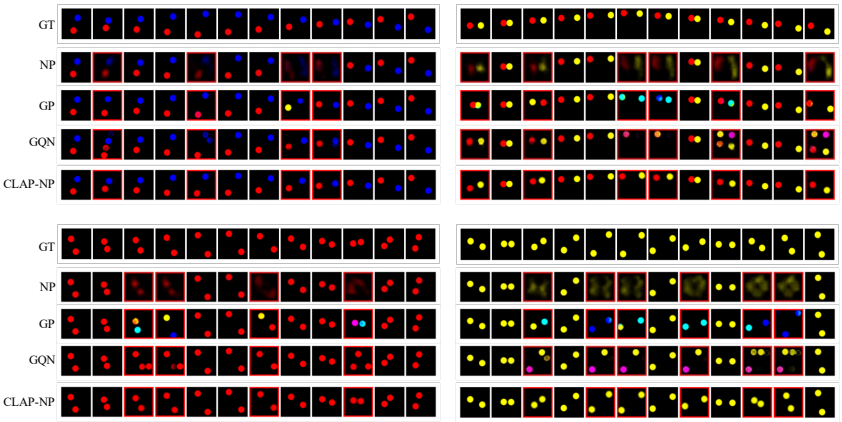
Gaussian Process (GP) We use a pretrained autoencoder to reduce the dimensionality of images, by which GP needs to capture the changing patterns in the latent space and thus can handle image-valued functions. We adopt the same encoder and decoder as in CLAP-NP and NP for a fair comparison. The mean function of GP is set to a constant function. By comparing the RBF kernel, periodic kernel, and deep kernel, we finally chose the deep kernel as the kernel function that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma^2 \exp(-\|\mathcal{T}_k(\boldsymbol{x}_i) - \mathcal{T}_k(\boldsymbol{x}_j)\|/2\ell^2)$ (Wilson et al., 2016). The architecture of the neural network $\mathcal{T}_k$ in the deep kernel is a multilayer perceptron with hidden sizes [1000, 1000, 500, 50], output size 2, and ReLU activation functions, which is the same as in DKL (Wilson et al., 2016). The hyperparameters of GP are tuned on each sample through the log-likelihood of context points and the RMSprop optimizer. We give the learning rate and epoch to adjust hyperparameters in Table 6. To model the correlations between dimensions, we use multi-task Gaussian Process prediction (Bonilla et al., 2007).

C.3 Computational Resource

We train our model and baselines on the server with Intel(R) Xeon(R) Gold 6133 CPUs, 24GB NVIDIA GeForce RTX 3090 GPUs, 512GB RAM, and Ubuntu 18.04 OS. All models are implemented with the PyTorch (Paszke et al., 2019) framework. To avoid the randomness in dataset splitting, we fix the random seed to 0.



(a) BoBa-1



(b) BoBa-2

Figure 9: Prediction results on BoBa. The images with red boxes are prediction results of models.

Table 7: MSE scores on CRPM-T and CRPM-C.

Model	CRPM-T (η = 1∼2/9)		CRPM-C (η = 1∼2/9)	
	η = 2/9	η = 3/9	η = 2/9	η = 3/9
NP	125.8 ± 4.9	204.6 ± 5.9	112.8 ± 6.9	186.8 ± 12.6
GP	416.4 ± 44.2	611.0 ± 44.9	286.0 ± 26.5	432.1 ± 27.4
GQN	235.2 ± 4.7	493.4 ± 7.2	246.5 ± 5.2	484.6 ± 8.2
CLAP-NP	**41.3 ± 1.8**	**89.1 ± 13.7**	**52.4 ± 11.3**	**136.6 ± 15.8**

D Additional Experimental Results

D.1 Intuitive Physics

In Figure 9 we display additional prediction results on BoBa-1 and BoBa-2. For each instance, there are samples that correspond to Test-4 (left) and Test-6 (right), where CLAP-NP outperforms other baselines. NP generates blurred target images on both BoBa-1 and BoBa-2 datasets. It indicates that NP has difficulty modeling changing patterns on bouncing balls. GQN can produce clear images but may generate non-existent balls and lose existing balls. In the first sample of BoBa-1, the predictions of GQN deviate from the ground truths significantly in the position of balls. Overall. CLAP-NP performs well in modeling scene consistency and predicting motion trajectory.

D.2 Abstract Visual Reasoning

With the additional experimental results in Figure 10, we can intuitively see that CLAP-NP has a better understanding of the changing rules than baselines. For the second sample of each dataset, a row of images is removed to interpret the abstract visual reasoning ability of the model. As samples in Figure 10 shows, when we remove a row of images, the answer is not unique: the predictions are correct as long as their sizes increase progressively and their colors and rotations keep constant. CLAP-NP represents such randomness in predictions by means of the probabilistic generative process, making it possible to generate different correct answers. In terms of the prediction quality, although generating blurred images, NP has the basic reasoning ability about the progressive rule of the outer triangle size in Figure 10b. GQN generates clear images, however, the generated images probably deviate from the underlying changing rules on matrices. Table 7 and Table 11 illustrate the outperforming results of CLAP-NP in abstract visual reasoning by quantitative experiments.

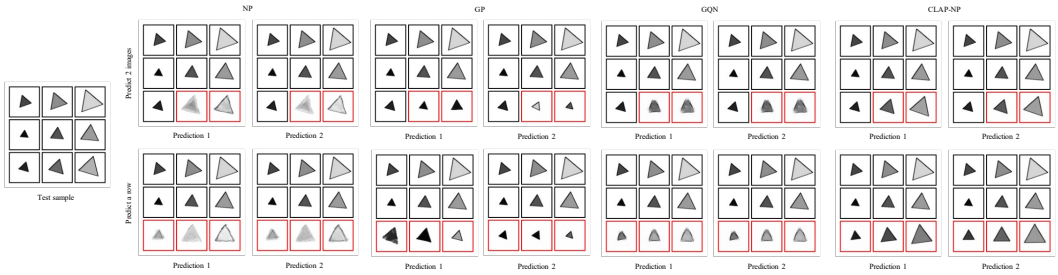
D.3 Scene Representation

Figure 12 shows the prediction results within 8 scene images. Both GQN and CLAP-NP generate clear prediction results when the number of target images is 4; if the number of target images is increased to 6, the prediction results of GQN have significant deviations, while CLAP-NP maintains the accuracy of predictions. NP generates ambiguous results in all experiments. Figure 13 shows a more complicated situation: we provide 40 scene images and set 20 of them as target images. In this case, GQN and NP can hardly generate clear foreground objects; while CLAP-NP produces relatively accurate predictions with only a little decrease in generative quality. This experiment aims to test the generalization of the laws learned by the models, and the results above illustrate that NP can hardly represent scenes with functions, GQN has difficulty generalizing the scene representation ability to different configurations, and the laws learned by CLAP-NP can be generalized to novel scenes with more available images.

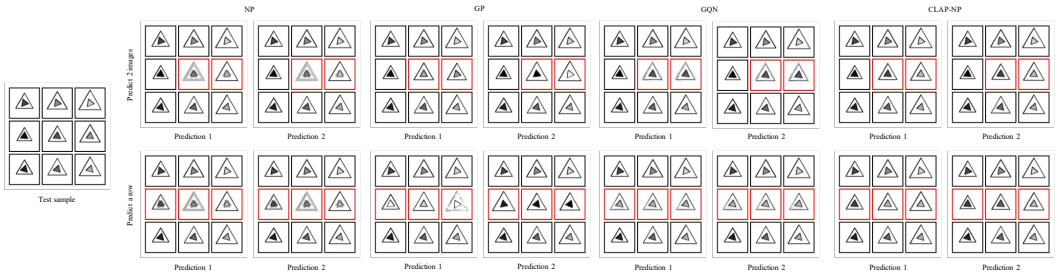
D.4 Concept Decomposition

Concept decomposition is the foundation for CLAP to understand concept-specific laws. In this experiment, we traverse each concept in the latent space and visualize the concepts through a decoder to illustrate the LRF learned for each concept. First of all, we decompose a batch of images into

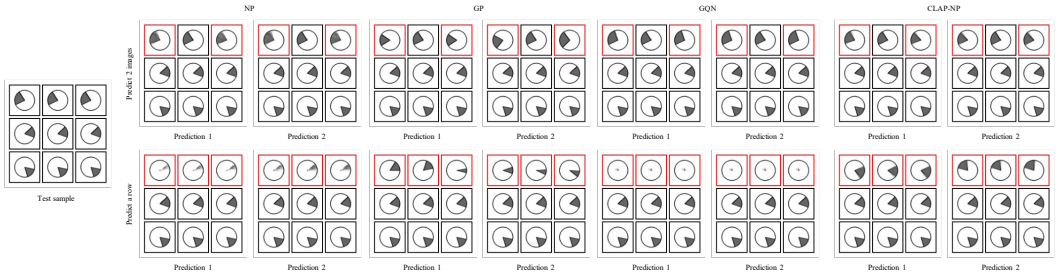
[2]https://github.com/iShohei220/torch-gqn



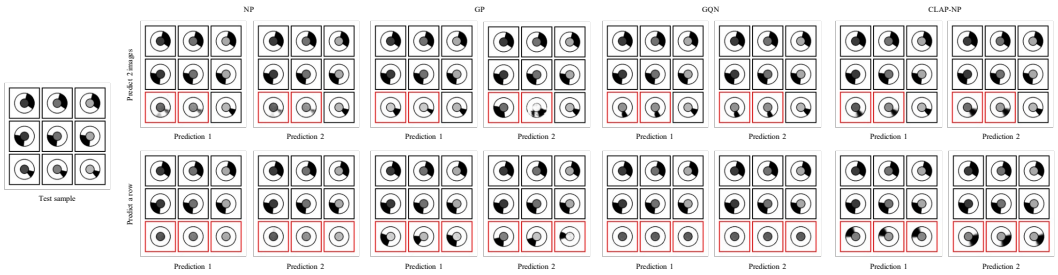
(a) CRPM-T



(b) CRPM-DT



(c) CRPM-C



(d) CRPM-DC

Figure 10: Predictions on CRPM. For each sample, we show two predictions sampled from models to test the model's understanding of attribute-specific rules.

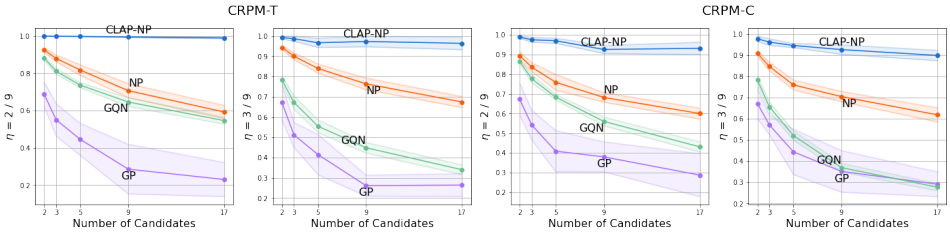


Figure 11: SA scores on CRPM-T and CRPM-C.

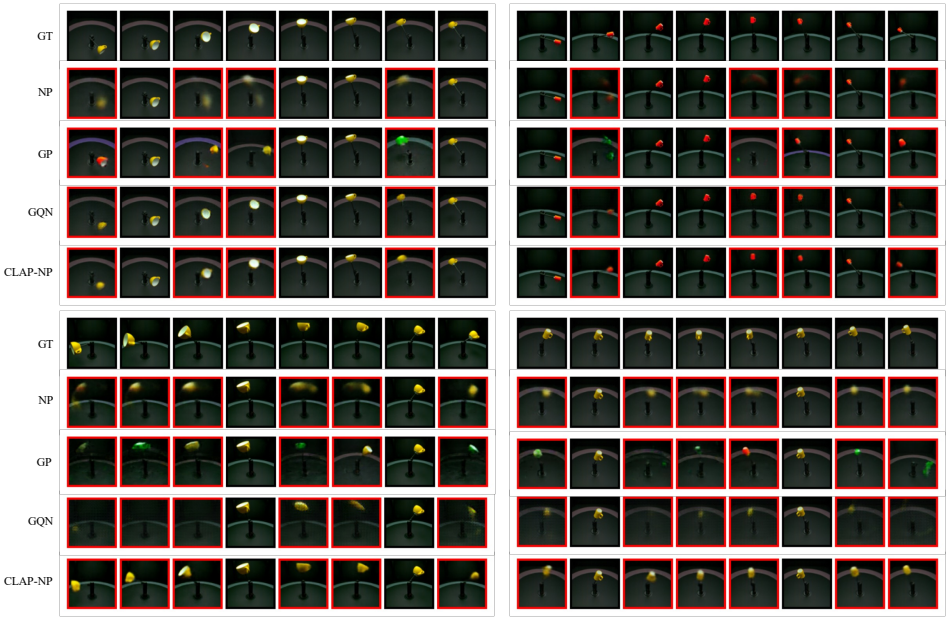


Figure 12: Target prediction with 8 scene images on MPI3D

concepts by the encoder to estimate the range of concepts in the latent space. To traverse one concept, we fix the other concepts and linearly interpolate it from the minimum value to the maximum value to generate a sequence of interpolation results, which are decoded into images for visualization. Each row of Figure 14 represents the latent traversal results of one concept. In BoBa-1, CLAP-NP learns LRFs on concepts of color, horizontal position, and vertical position in an unsupervised manner. This is similar to the way we understand the motion of balls: the color keeps constant over time; the horizontal and vertical positions conform to the physical laws in their respective directions. For CRPM, CLAP-NP correctly parses images into concepts that correspond to the attribute-specific rules in matrices. Concept decomposition in real environments is a challenge for models. For MPI3D, CLAP-NP parses the LRFs on the object axis (Dimension 1, 5, 6, 7) and other static scene attributes (Dimension 2, 3, 4).

D.5 Latent Random Function Composition

By means of concept decomposition, CLAP-NP can parse the latent random function of each concept to represent concept-specific laws, by which the law of a sample can be modified in terms of concepts. Figure 15 and 16 display the way to edit the laws. We compose the concept-specific LRFs of existing samples to generate samples that have novel changing patterns. For example, in the third sample of Figure 15a, we generate a sample with the progressive law of size, constant law of rotation, and constant law of color from three existing samples. The results on four instances of CRPM indicate that CLAP-NP understands the changing patterns on matrices in an interpretable and manipulable way, which embodies the compositionality of laws.

D.6 Latent Random Function Exchange

Another way to edit laws is to exchange LRFs of some concepts between samples. In BoBa, we swap the law of collisions and appearance between samples, and the results are shown in Figure 17. According to the concept decomposition results in Figure 14, we exchange the LRFs on Dimension 1 (Dimensions 2 and 3) to modify the law of collisions (the law of appearance) in BoBa-1. And we exchange the LRFs on Dimensions 1 and 6 (Dimensions 2, 3, 4, and 5) to modify the law of collisions (the law of appearance) in BoBa-2. To exchange laws in MPI3D, as shown in Figure 18, we swap LRFs on Dimensions 1, 5, 6, and 7 to change the law of the object axis. By modifying laws on example 1 of Figure 18, the changing pattern of the first sample becomes the horizontal rotation, and the second sample inherits the law of vertical rotation from the first sample, which

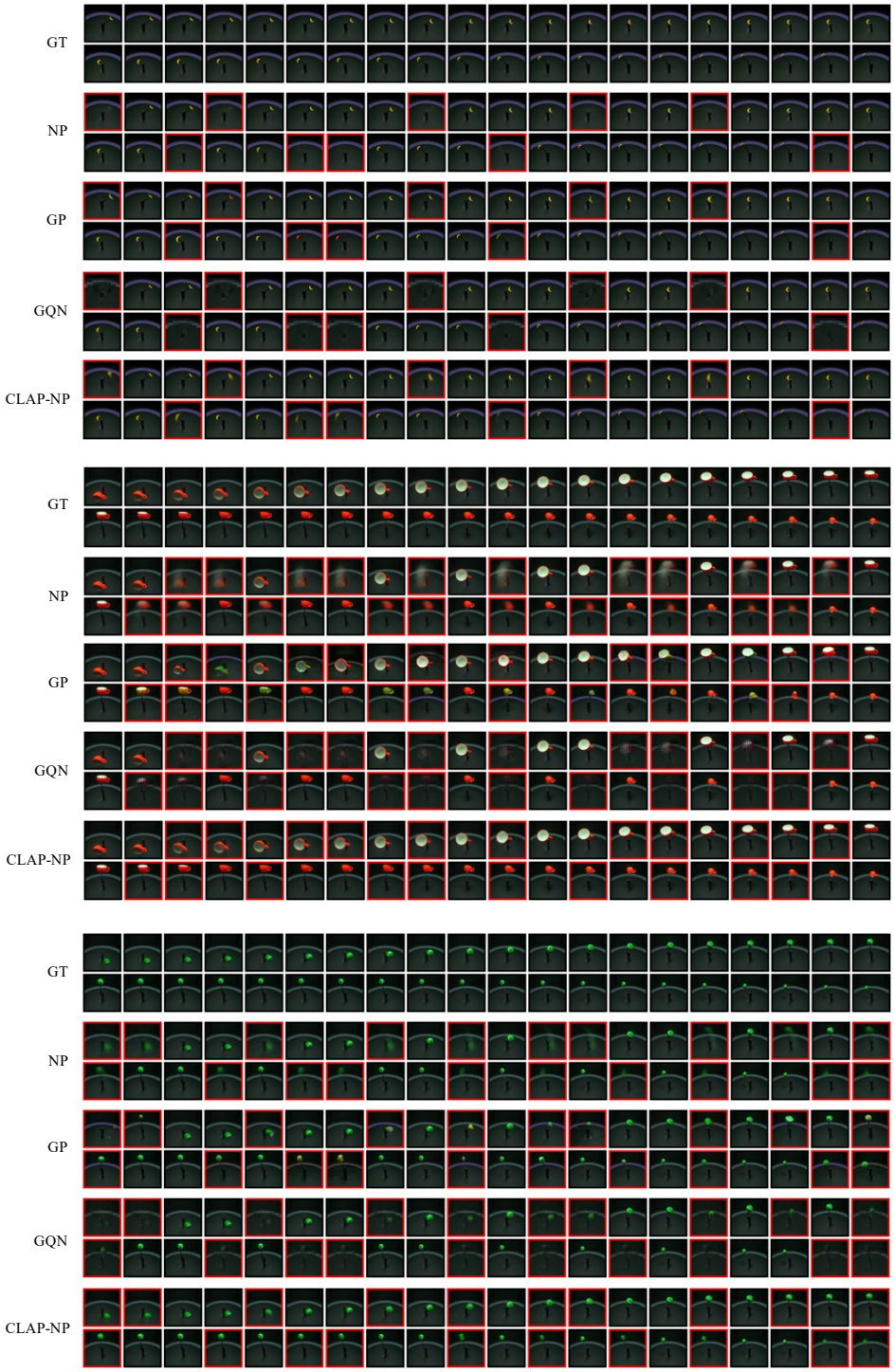


Figure 13: Target prediction with 40 scene images on MPI3D

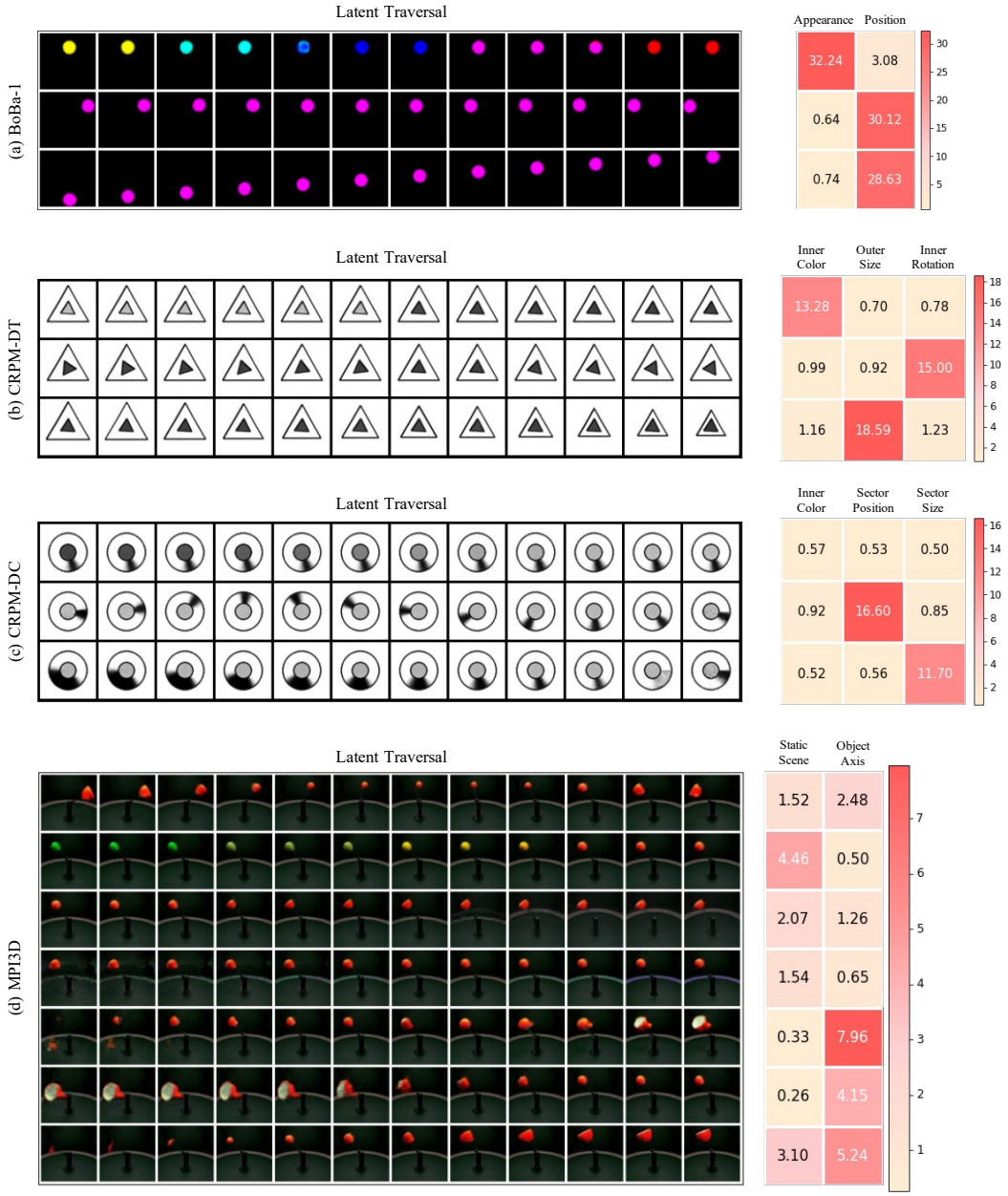
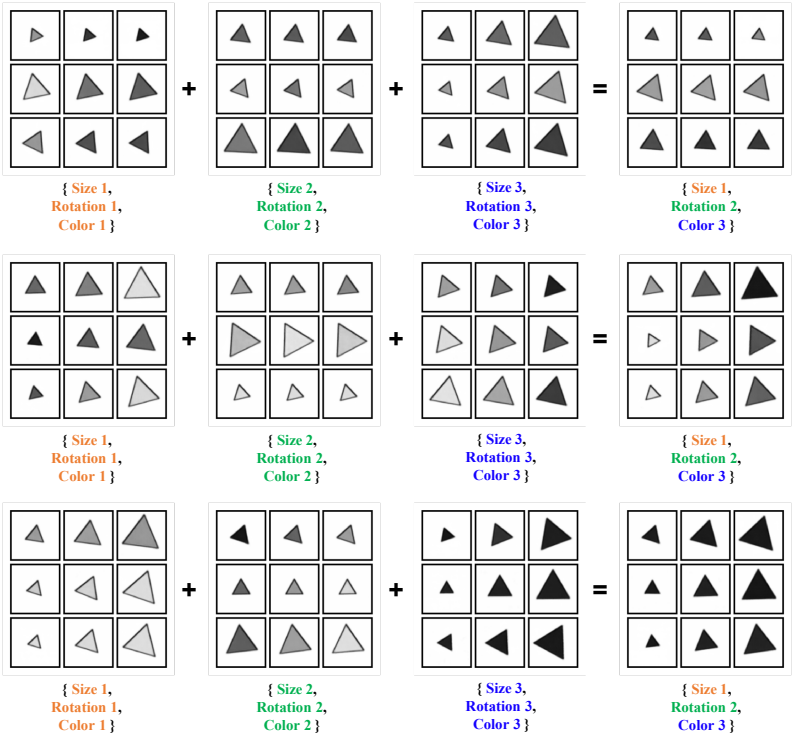


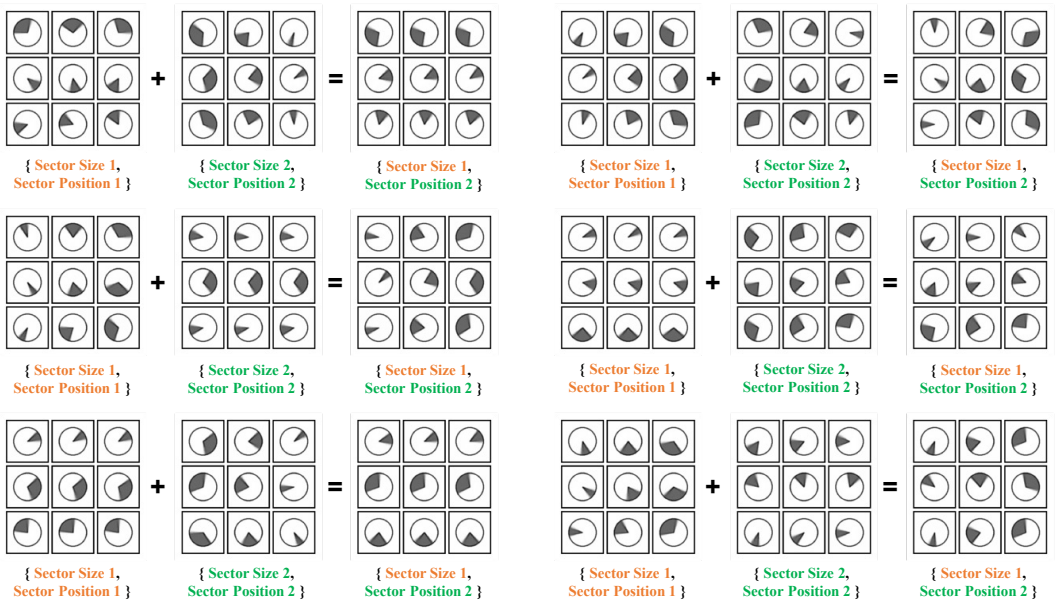
Figure 14: The latent traversal results and variance declines for BoBa-1, CRPM-DT, CRPM-DC, and MPI3D.



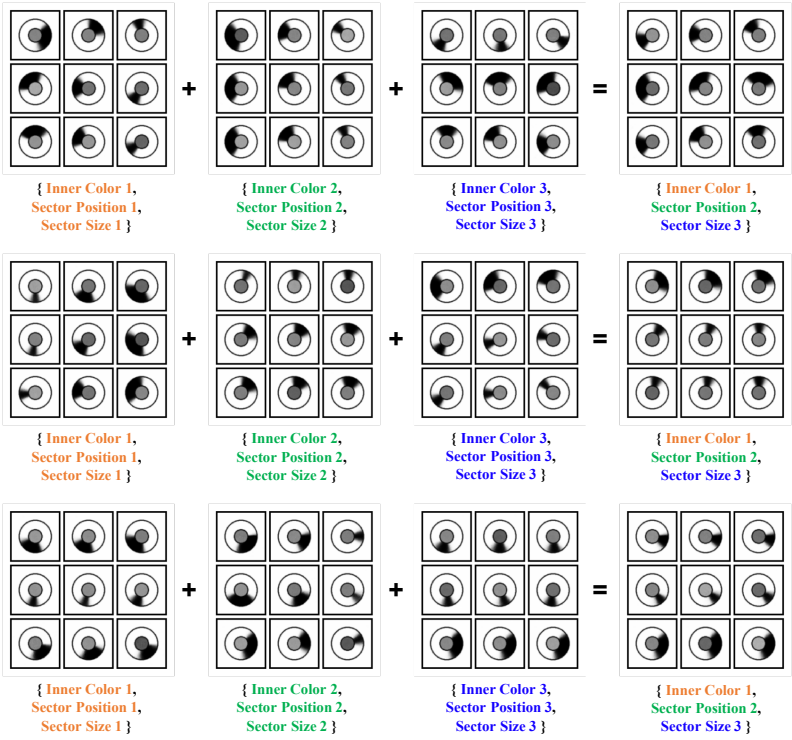
(a) CRPM-T

(b) CRPM-DT

Figure 15: Law Composition on CRPM-T and CRPM-DT.

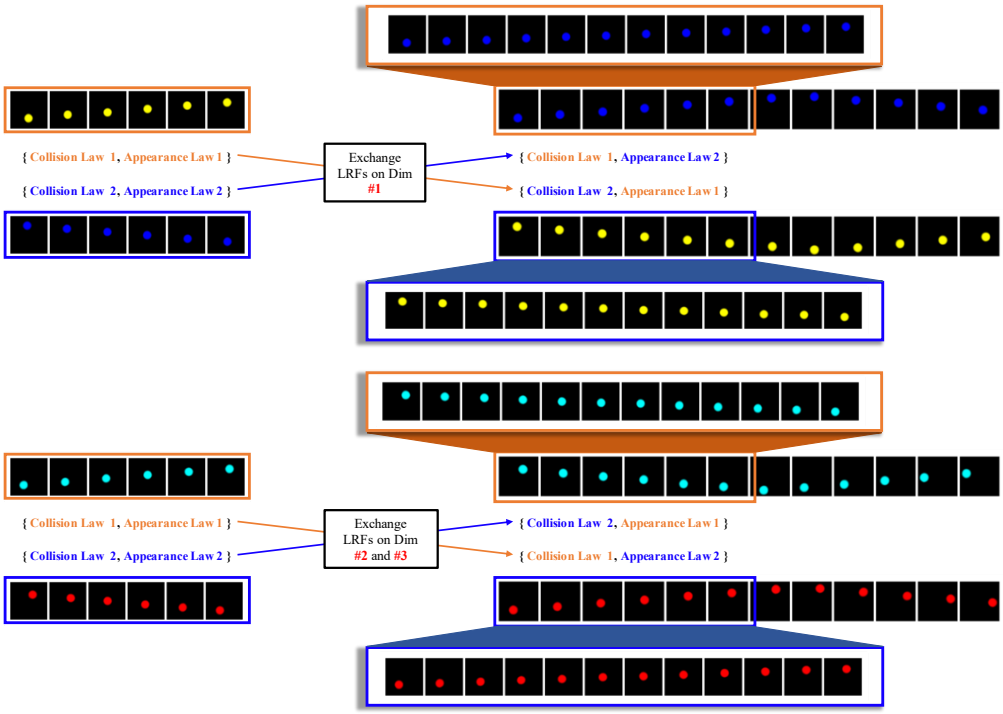


(a) CRPM-C

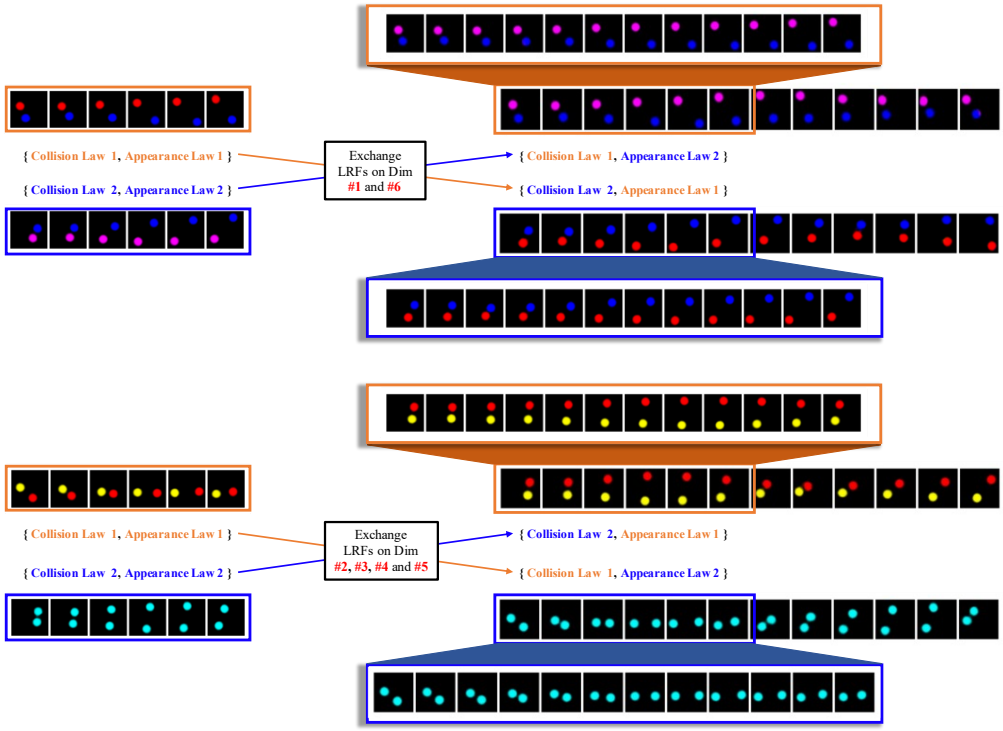


(b) CRPM-DC

Figure 16: Law Composition on CRPM-C and CRPM-DC.



(a) BoBa-1



(b) BoBa-2

Figure 17: Law Exchange on BoBa-1 and BoBa-2.

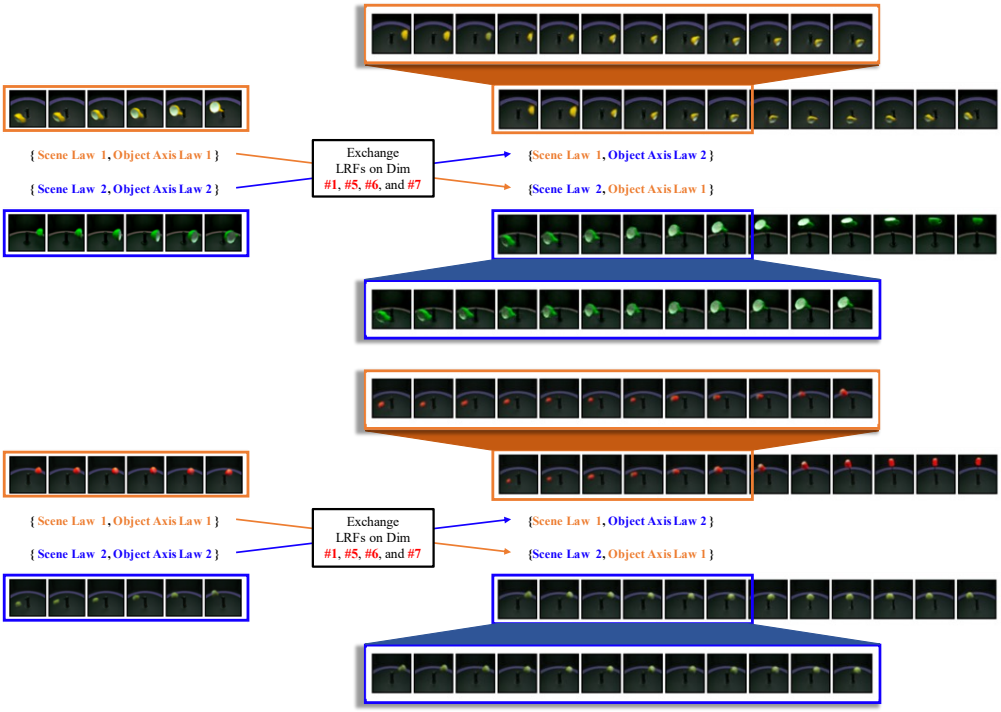


Figure 18: Law Exchange on MPI3D.

illustrates CLAP-NP's ability to generate new samples. The experiments on law exchange evidence CLAP-NP's interpretability and manipulability from another perspective.

D.7 Generalization on Unseen Concepts

We extend BoBa-2 to generate four datasets with novel laws: **Novel-colors** (BoBa-2-NC), **Novel-shape** (BoBa-2-NS), **Without-ball-collisions** (BoBa-2-WBO), and **With-gravity** (BoBa-2-WG). We draw balls on BoBa-2-NC with unseen colors and BoBa-2-NS with squares without changing the laws of positions. For BoBa-2-WBO, we disable collisions between balls and reserve collisions between balls and borders; for BoBa-2-WG, we add vertical gravity to scenes. After training on BoBa-2, CLAP-NP is directly tested by four datasets to evaluate whether the compositionality of laws helps generalization to unseen concepts. The experimental results in Figure 19 show that CLAP-NP can still correctly predict the object positions on datasets with novel colors and shapes. And the color and shape of objects deviate from those of real ones because the encoder and decoder are trained on BoBa-2, whose latent space does not include unseen colors or shapes. The same situation can be observed on BoBa-2-WBO and BoBa-2-WG. In the scene with unseen physical laws, CLAP-NP learns the correct law of appearance constancy but predicts incorrect positions of balls. To better illustrate the generalization ability in scenes with novel laws, we quantitatively evaluate CLAP-NP and baseline models by MSE ad SA scores on the four datasets. Table 8 shows MSE scores and SA scores with different numbers of candidates. CLAP-NP achieves the best MSE and SA scores in most situations, illustrating CLAP-NP's generalization ability in scenes with novel laws.

D.8 Number of Latent Random Functions

This experiment explores the influence of setting too large or too small numbers of latent random functions (LRFs) on CLAP-NP. Figure 20 shows the latent traversal results on MPI3D, BoBa-2, and CRPM-DT. If we set too few LRFs, CLAP-NP learns mixed laws among dimensions instead of the expected compositional laws. Moreover, too few LRFs can influence CLAP-NP's generation ability (e.g., set only two LRFs for BoBa-2). Setting too many LRFs has no significant influence on CLAP-NP's performance because there will be redundant dimensions in CLAP-NP that do not encode any

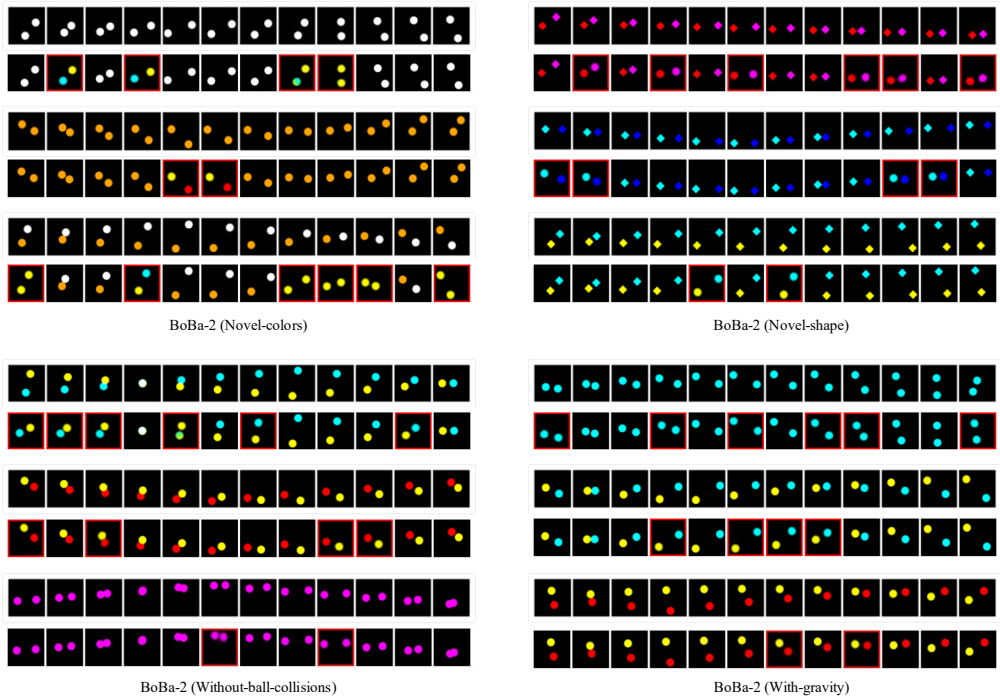


Figure 19: Generalize CLAP-NP to novel colors, shapes, and physical laws on BoBa-2 dataset.

Table 8: MSE and SA scores on BoBa-2 with novel laws. The models are tested by $\eta = 4/12$, and SA-k indicates the SA scores with k candidates.

Metric	Model	BoBa-2-NC	BoBa-2-NS	BoBa-2-WBO	BoBa-2-WG
MSE	NP	2047.6 ± 10.9	990.2 ± 14.1	1743.1 ± 15.7	1525.5 ± 12.9
	GP	2671.9 ± 75.9	1907.4 ± 74.2	2068.2 ± 124.9	2019.9 ± 85.0
	GQN	2350.8 ± 55.4	1459.7 ± 24.6	2365.8 ± 38.3	2092.9 ± 23.5
	CLAP-NP	**1611.2 ± 71.1**	**979.0 ± 72.5**	**1420.3 ± 60.4**	**1322.4 ± 58.9**
SA-2	NP	0.984 ± 0.004	0.989 ± 0.003	0.925 ± 0.010	0.956 ± 0.008
	GP	0.991 ± 0.005	0.986 ± 0.009	0.974 ± 0.011	0.985 ± 0.012
	GQN	0.952 ± 0.009	0.955 ± 0.010	0.878 ± 0.015	0.891 ± 0.012
	CLAP-NP	**0.995 ± 0.004**	**0.999 ± 0.001**	**0.997 ± 0.002**	**0.998 ± 0.001**
SA-3	NP	0.972 ± 0.005	0.981 ± 0.004	0.876 ± 0.018	0.924 ± 0.005
	GP	0.986 ± 0.013	0.970 ± 0.021	0.940 ± 0.035	0.972 ± 0.019
	GQN	0.920 ± 0.007	0.926 ± 0.012	0.821 ± 0.009	0.820 ± 0.015
	CLAP-NP	**0.990 ± 0.006**	**0.998 ± 0.002**	**0.995 ± 0.005**	**0.998 ± 0.002**
SA-5	NP	0.949 ± 0.008	0.969 ± 0.005	0.796 ± 0.012	0.877 ± 0.010
	GP	**0.982** ± 0.012	0.962 ± 0.021	0.907 ± 0.049	0.968 ± 0.018
	GQN	0.884 ± 0.013	0.887 ± 0.010	0.734 ± 0.028	0.741 ± 0.013
	CLAP-NP	0.978 ± 0.016	**0.997 ± 0.003**	**0.991 ± 0.006**	**0.995 ± 0.004**
SA-9	NP	0.918 ± 0.017	0.945 ± 0.009	0.705 ± 0.021	0.823 ± 0.020
	GP	0.954 ± 0.031	0.948 ± 0.021	0.879 ± 0.038	0.946 ± 0.028
	GQN	0.828 ± 0.018	0.842 ± 0.015	0.627 ± 0.026	0.643 ± 0.028
	CLAP-NP	**0.975 ± 0.010**	**0.992 ± 0.005**	**0.982 ± 0.008**	**0.991 ± 0.004**
SA-17	NP	0.884 ± 0.015	0.917 ± 0.011	0.584 ± 0.026	0.752 ± 0.019
	GP	0.938 ± 0.044	0.916 ± 0.045	0.826 ± 0.030	0.926 ± 0.034
	GQN	0.781 ± 0.023	0.782 ± 0.026	0.549 ± 0.014	0.558 ± 0.016
	CLAP-NP	**0.965 ± 0.014**	**0.991 ± 0.006**	**0.972 ± 0.007**	**0.982 ± 0.013**

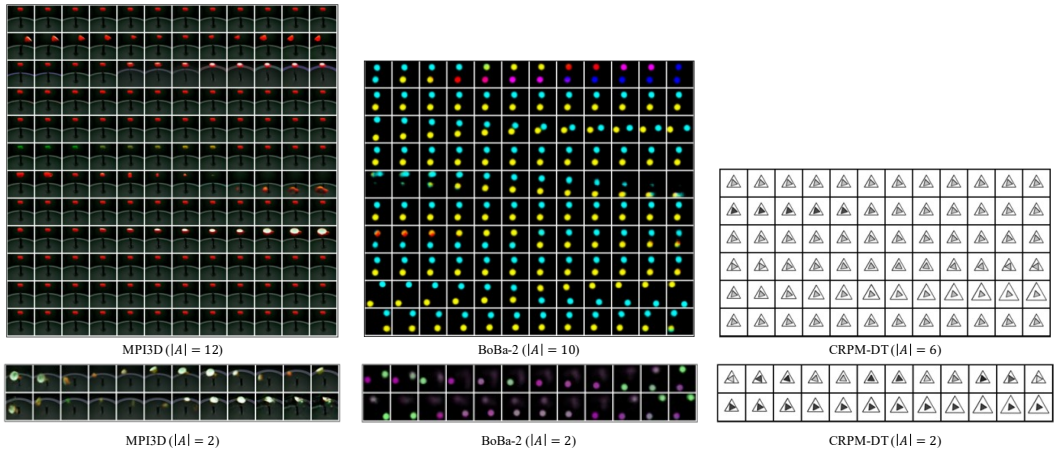


Figure 20: Latent traversal results on MPI3D, BoBa-2, and CRPM-DT with too many (top) and too few (bottom) latent random functions.

Table 9: MSE scores on CRPM-DT, BoBa-2, and MPI3D where CLAP-NP predicts target images by the rollout and one-shot strategy.

Dataset	CRPM-DT (η = 6/9)	BoBa-2 (η = 6/12)	MPI3D (η = 20/40)
Rollout	362.5 $\pm$ 3.5	**4651.8** $\pm$ **15.7**	**228.2** $\pm$ **0.6**
One-Shot	**361.9** $\pm$ **2.3**	4778.5 $\pm$ 15.8	237.3 $\pm$ 1.2

information (e.g., dimensions 1, 3, and 6 on CRPM-DT). However, due to the independent modeling of concept-specific LRFs, setting a large number of LRFs will reduce the computational speed of CLAP-NP and increase the number of model parameters.

D.9 Prediction Strategy

CLAP uses the one-shot strategy to predict all target images at one time. However, the rollout strategy can be another choice when predicting the following target images through a few context images. For example, in the test configuration $\eta = 20/25$, we first predict the 6th to 10th images from the given first five images, then combine them (1st to 10th images) to predict the following five images, and repeat this process until predicting all target images. Table 9 shows the MSE scores on CRPM-DT, BoBa-2, and MPI3D where CLAP-NP predicts targets with the rollout and one-shot strategy, respectively. On BoBa-2 and MPI3D, the rollout strategy slightly improves the prediction accuracy. If a target image is far away from all the context images, the prediction is likely to have high uncertainty. In this experiment, we give the first few images as the contexts and let the model predict the following target images. The rollout strategy only predicts a few target images at close positions each time, while the one-shot strategy has to predict all the target images at one time. Therefore, the rollout strategy is more likely to reduce uncertainty and improve prediction accuracy.

D.10 Failure Cases

In this experiment, we display some failure cases. For BoBa, most failure cases occur when there are continuous target images (1st sample of BoBa in Figure 21) or too many target images (2nd sample of BoBa in Figure 21). For CRPM, CLAP can predict target images with diversity when we remove the third row from a matrix, but it sometimes generates target images that break the rules. For example, the color of images keeps invariant in the first two rows, but CLAP generates images in the third row with changing grayscales (samples of CRPM in Figure 21). For MPI3D, when the object size has an obvious change (1st sample of MPI3D in Figure 21) or the centric object is tiny (2nd sample of MPI3D in Figure 21), the predictions will deviate or become unclear.

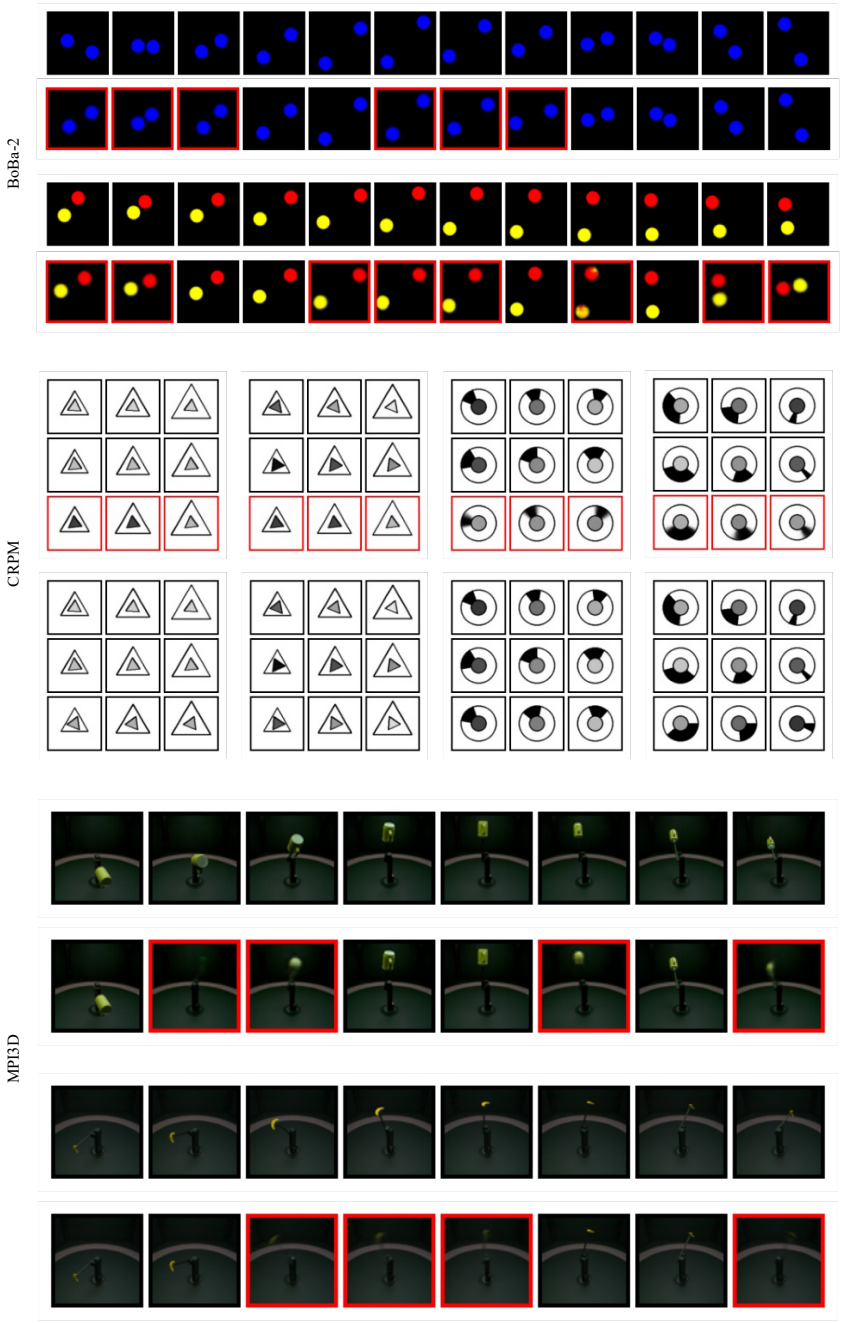


Figure 21: Failure cases on datasets.

E LIMITATIONS

We conclude our limitations in two aspects. (1) **Complexity of datasets.** Because compositional law parsing is an underexplored task, we first validate the effectiveness of SVCL on datasets with relatively clear and simple rules to avoid the influence of unknown confounders in complicated datasets. We believe that the compositionality of laws also exists in more complex scenarios (e.g., learning physical laws in realistic scenes). Furthermore, some vision tasks may benefit from compositional law parsing. For example, we can perform a controllable video generation process based on law modification or make more interpretable predictions by analyzing dominant laws in videos. Discovering such compositional law parsing ability in more complex situations can be a valuable topic in future works. (2) **Setting the number of LRFs.** CLAP places an upper bound on the number of LRFs. For scenes with multiple complex rules, we can empirically set an appropriate upper bound or directly put a large upper bound for the number of LRFs. However, a large bound will linearly increase the number of model parameters, and the redundant concepts will waste computing resources, decreasing computational efficiency. Therefore, exploring a mechanism in the future to dynamically adjust the number of functions will be meaningful. When applying CLAP to more complex scenes, we may introduce more inductive biases for better concept decomposition (e.g., use a more task-specific encoder or decoder). We design CLAP as an unsupervised model independent of task-specific annotations from tasks, which means that it can be extended to take advantage of these annotations. Thus we can find ways to integrate CLAP with additional supervision information (e.g., supervise SVCL with the labels of changing factors for each scene) to help concept learning.

REFERENCES

Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction. *Advances in neural information processing systems*, 20, 2007.

Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2610–2620, 2018.

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.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. *Advances in Neural Information Processing Systems*, 32, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.