VAEL: BRIDGING VARIATIONAL AUTOENCODERS AND PROBABILISTIC LOGIC PROGRAMMING

Anonymous authors
Paper under double-blind review

ABSTRACT

We present VAEL, a neuro-symbolic generative model integrating variational autoencoders (VAE) with the reasoning capabilities of probabilistic logic (L) programming. Besides standard latent subsymbolic variables, our model exploits a probabilistic logic program to define a further structured representation, which is used for logical reasoning. The entire process is end-to-end differentiable. Once trained, VAEL can solve new unseen generation tasks by (i) leveraging the previously acquired knowledge encoded in the neural component and (ii) exploiting new logical programs on the structured latent space. Our experiments provide support on the benefits of this neuro-symbolic integration both in terms of task generalization and data efficiency. To the best of our knowledge, this work is the first to propose a general-purpose end-to-end framework integrating probabilistic logic programming into a deep generative model.

1 INTRODUCTION

Neuro-symbolic learning has gained tremendous attention in the last few years (Besold et al., 2017; De Raedt et al., 2020; Kautz, 2020; Bengio & Marcus, 2020) as such integration has the potential of leading to a new era of intelligent solutions, enabling the integration of deep learning and reasoning strategies (e.g. logic-based or expert systems). While a lot of effort has been devoted to devising neuro-symbolic methods in the discriminative setting (Manhaeve et al., 2018; Yi et al., 2018; Minervini et al., 2020), less attention has been paid to the generative counterpart. An ideal generative neuro-symbolic framework should be able to encode the available small amount of training data into an expressive symbolic representation and to exploit complex forms of high level reasoning on such representation to generate new data samples. This is far from the actual state-of-the-art, where neuro-symbolic methods (Jiang & Ahn, 2020; Feinman & Lake, 2020; Gothoskar et al., 2021) have been mostly applied on generative tasks requiring only spatial-reasoning. As a motivation for this work, consider a task where a single image of multiple handwritten numbers is labeled with their sum. Suppose that we want to generate new images not only given their addition, but also given their multiplication, power, etc. Common generative approaches, like VAE-based models, have a strong connection between the latent representation and the label of the training task (i.e. the addition) (Kingma et al., 2014; Joy et al., 2021). Consequently, when considering new generation tasks that go beyond the simple addition, they have to be retrained on new data.

In this paper, we tackle the problem by providing a novel generative neuro-symbolic solution, named VAEL. In VAEL, the latent representation is not directly linked to the label of the task, but to a set of newly introduced symbols, i.e. logical expressions. Starting from these expressions, we use a *probabilistic logic program* to deduce the label. Importantly, the neural component only needs to learn a mapping from the raw data to this new symbolic representation. In this way, the model only weakly depends on the supervised information and can generalize to new generation tasks involving the same set of symbols. Moreover, the reasoning component offers a strong inductive bias, which enables a more data efficient learning.

2 THE VAEL MODEL

We propose a probabilistic graphical model which enables to unify VAEs with Probabilistic Logic Programming.

Specifically, we use ProbLog (De Raedt et al., 2007), (see Appendix A), which lifts logic programs to *probabilistic logic programs* through the introduction of probabilistic facts $p_i :: f_i$, i.e. logical fact that are true with probability p_i. A ProbLog program defines a probability distribution over possible worlds $P(w_F; p)$, where a possible world is an assignment of truth values to the probabilistic facts F and to the facts that can be derived from F using the logic program. The graphical model of VAEL is displayed in Figure 1, where black arrows refer to the generative model, whereas blue dashed arrows correspond to the inference counterpart. The model consists of four core variables. $x \in \mathbb{R}^{H \times W \times C}$ represents the image we want to generate, while $y \in \{0, 1\}^K$ represents a label, i.e. a symbolic information characterizing the image. The latent variable is split into a symbolic component $z_{sym} \in \mathbb{R}^N$ and a subsymbolic component $z \in \mathbb{R}^M$.

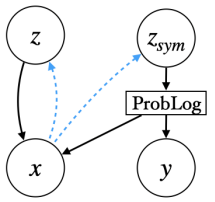

Figure 1: VAEL graphical model.

Generative model. The generative distribution of VAEL is factorized in the following way:

$$p_\theta(x, y, \mathbf{z}) = p(x|\mathbf{z})p(y|z_{sym})p(\mathbf{z}) \tag{1}$$

where $\mathbf{z} = [z_{sym}, z]$ and θ are the parameters of the generative model. $p(\mathbf{z})$ is a standard Gaussian distribution, while $p(y|z_{sym})$ is the success distribution of the label of the ProbLog program T. $p(x|\mathbf{z})$ is a Laplace distribution with mean value μ and identity covariance. Here, μ is a neural network decoder whose inputs are z and a possible ProbLog world $\omega_F \sim P(\omega_F; MLP(z_{sym}))$. Importantly, VAEL does not rely on a one-to-one mapping between y and z_{sym}, rather it exploits a probabilistic logic program to link them, viz. $p(y|z_{sym})$. Indeed, the probabilistic facts are used by the ProbLog program to compute the actual labels y and they can encode a more meaningful symbolic representation of the image than y. Additional technical details about the generative process and the graphical model are available in Appendix B.

Inference model. We amortise inference by using an approximate posterior distribution $q_\phi(\mathbf{z}|x, y)$ with parameters ϕ. Furthermore, we assume that $\mathbf{z}$ and y are conditionally independent given x, thus obtaining $q_\phi(\mathbf{z}|x, y) = q_\phi(\mathbf{z}|x)$[1]. This allows us to decouple the latent representation from the training task. Conversely, the other VAE frameworks do not exploit this assumption and have a latent representation that is dependent on the training task.

The overall VAEL model (including the inference and the generative components) is shown in Figure 2. We provide an example to showcase the inference and the generative parts of VAEL.

Example 1 *Consider a dataset composed of images of pairs of digits labelled with their sum, for example* $x =$  *labelled as* $y = 3$*. In Figure 2, we show the inference and generative components of VAEL on this task. First, the encoder (**left**) computes an approximated posterior of the latent variables* $\mathbf{z}$ *from the image* x*. The latent variables are split into two components: (i) a subsymbolic* z*, which is meant to encode subsymbolic properties of the image (e.g. writing style); and a symbolic* z_{sym}*, which is meant to encode the symbolic properties (i.e. the digits). A MLP is used to map the real variables* z_{sym} *into the probabilities* p_{ij} *of the facts in the following ProbLog program.*

```
p_10::digit(X,first,0); ...; p_19::digit(X,first,9).
p_20::digit(X,second,0); ...; p_29::digit(X,second,9).
addition(X,Y) :- digit(X,first, Z1), digit(X,second, Z2), Y is Z1 + Z2.
```

which states that an image `X` *is classified as* `Y` *if the* `first` *digit is classified as* `Z1`*, the second as* `Z2` *and they satisfy "*`Y` *is* $\texttt{Z1} + \texttt{Z2}$*". The program is used to compute the label* y *and a possible world, i.e. the identity of the two digits. Finally, a decoder (**right**) takes both the latent vector* z *and the possible world from ProbLog to reconstruct the image* $\tilde{x}$*.*

Objective function. The objective function of VAEL computes an evidence lower bound (ELBO) on the log likelihood of pair (x, y), namely:

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{REC}(\theta, \phi) + \mathcal{L}_Q(\theta, \phi) - \mathcal{D}_{\mathcal{KL}}[q_\phi(\mathbf{z}|x)||p(\mathbf{z})]] \tag{2}$$

where

$$\mathcal{L}_{REC}(\theta, \phi) = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|x)}[\log(p(x|\mathbf{z})], \quad \mathcal{L}_Q(\theta, \phi) = \mathbb{E}_{z_{sym} \sim q_\phi(z_{sym}|x))}[\log(p(y|z_{sym}))]].$$

[1]We use a Gaussian distribution with a mean parameterized by the encoder network and identity covariance.

The ELBO is used to train VAEL in an end-to-end differentiable manner, thanks to the Reparametrization Trick (Kingma & Welling, 2014) at the level of the encoder $q_\phi(\mathbf{z}|x)$ and the differentiability of the ProbLog inference, which is used to compute the success probability of a query and sample a world. In Appendix C, we report VAEL training algorithm (Algorithm 1) along with further details on the training procedure.

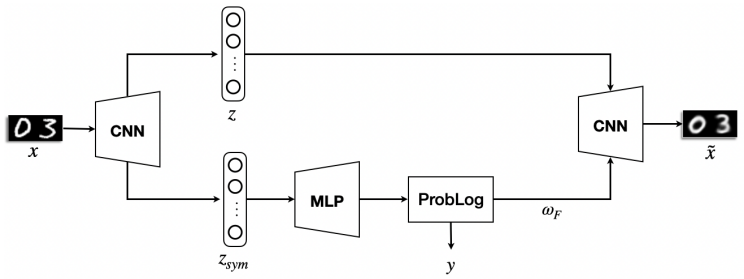

Figure 2: The VAEL model on the MNIST addition generative task.

3 EXPERIMENTS

We test our model on four tasks: 1. *classification*, i.e., predicting the correct label given the input image; 2. *joint generation*, i.e., generating both the image and the label; 3. *conditional generation*, i.e., generating the image given the label; 4. *out-of-task generalization*, i.e., generating the image given the label of a different task than the training one. In fact, once trained VAEL on a specific symbolic task (e.g., the addition of two digits), we can generalize to any novel task that involves reasoning with the same set of probabilistic facts by simply changing the ProbLog program accordingly (indeed, we can generalize to the multiplication of two integers).

When possible, we compare VAEL against the state-of-the-art CCVAE (Joy et al., 2021) We evaluate the models by relying on a *reconstruction loss* (m_{REC}) in terms of data log-likelihood and two accuracies, *predictive accuracy* (m_{CLASS}), i.e. the accuracy on the label y, and *generative accuracy* (m_{GEN}), i.e. the accuracy of a pre-trained MNIST classifier to identify the generated digits.

We created two different datasets to validate our approach. The first dataset, named *2digit MNIST*, contains 64.400 images of two digits taken from the MNIST dataset (LeCun et al., 1998). Each image is labelled with the sum of the two digits. The combinatorial nature of the dataset makes any task defined on it harder than its single-digit counterpart. The second dataset, referred to as *2level Mario*, consists of $6,720$ images of two consequent states of a 3×3 grid world where an agent can move by one single step (diagonals excluded). Each image in the *2level Mario* dataset is labelled with the move performed by the agent. Both datasets present a compositional scene that showcases the advantages of using a neuro-symbolic approach to logically reasoning upon the scene components. Moreover, the compositional nature allows us to design novel tasks that imply arbitrarily complex forms of reasoning among the scene elements.

Table 1: Reconstructive, predictive and generative ability of VAEL and CCVAE. We use repeated trials to evaluate both the models on a test set of $10K$ images for *2digit MNIST* dataset and 1344 images for *2level Mario* dataset.

	2digit MNIST			*2level Mario*		
	$m_{REC}(\downarrow)$	$m_{CLASS}(\uparrow)$	$m_{GEN}(\uparrow)$	$m_{REC}(\downarrow)$	$m_{CLASS}(\uparrow)$	$m_{GEN}(\uparrow)$
CCVAE	1549 ± 2	0.53 ± 0.01	0.51 ± 0.02	43461 ± 209	$\mathbf{1.00 \pm 0.00}$	0.00 ± 0.00
VAEL	$\mathbf{1542 \pm 3}$	$\mathbf{0.85 \pm 0.02}$	$\mathbf{0.79 \pm 0.04}$	$\mathbf{42734 \pm 246}$	$\mathbf{0.98 \pm 0.06}$	$\mathbf{0.81 \pm 0.30}$

Main results. As shown in Table 1, CCVAE and VAEL achieve comparable predictive accuracy in *2level Mario* dataset (**classification**). However, VAEL generalizes better than CCVAE in *2digit MNIST* dataset. The reason behind this performance gap is due to the fact that the

addition task is combinatorial in nature and CCVAE would require a larger number of training samples in order to solve it. We further investigate this aspect in the *Data Efficiency* experiment. Furthermore, VAEL outperforms CCVAE in terms of both reconstructive and generative ability (**joint-generation**). The difference in performance between the models lies in the fact that for each label there are many possible correct images. For example, in the *2level Mario* dataset, there are 6 possible pairs of agent's positions that correspond to the label `left`. Our probabilistic logic program explicitly encodes the digits value or the single agent's positions in its probabilistic facts, and uses the variable z_{sym} to compute their probabilities. On the contrary, CCVAE is not able to learn the proper mapping from the digits value or the agent's positions to the label, but it can learn to encode only the label in the latent space z_{sym}. We define several novel tasks to evaluate the task generative ability of VAEL (**out-of-task generalization**). For example, for *2digit MNIST* dataset, we introduce the *multiplication*, *subtraction* and *power* between two digits. As shown in Figure 3, VAEL is able to conditionally generate pairs of numbers consistent with the the result y of the corresponding mathematical operation between the first and second digit. To the best of our knowledge, such a level of task generalization cannot be achieved by any existing VAE framework. On the contrary, in VAEL, we can generalize by simply substituting the ProbLog program used for the training task with the program for the desired target task, without re-training the model. Additional results, included **conditional generation**, can be found in Appendix F.

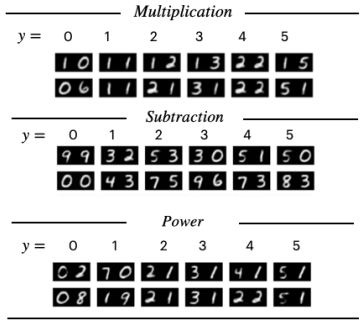

Figure 3: Examples of VAEL task generalization for *2digit MNIST* dataset.

Data Efficiency. Here, we want to verify whether the use of a logic-based prior helps the learning in contexts characterized by data scarcity. To this goal, we define different training splits of increasing size for the *addition* task of *2digit MNIST* dataset. In particular, the different splits range from 10 up to 100 images per pair of digits. The results (Figure 4) show that VAEL outperforms the baseline for all the tested sizes. In fact, with only 10 images per pair, VAEL already performs better than CCVAE trained with 100 images per pair. The reason behind this disparity is that the logic-based prior helps the neural model in properly structuring the latent representation, so that one part can easily focus on recognizing individual digits and the other on capturing the remaining information in the scene. Conversely, CCVAE needs to learn how to correctly model very different pairs that sum up to the same value. We further investigated the gap between CCVAE and VAEL in Appendix E.

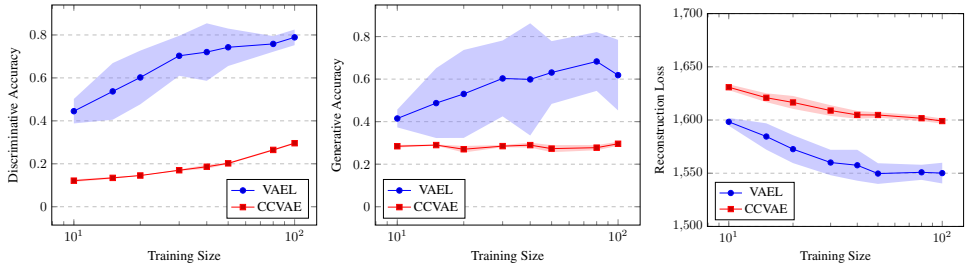

Figure 4: Discriminative, generative and reconstructive ability of VAEL (**blue**) and CCVAE (**red**) trained in contexts characterized by data scarcity. Both the models are evaluated on the same test set. The training size refers to the number of samples per pair of digits seen during the training.

4 Future Works

In the future, we plan to improve VAEL by i) investigating alternative and more scalable semantics for probabilistic programs (e.g. stochastic logic program Winters et al. (2022)), and ii) exploring different solutions to achieve out-of-distribution generalization. Moreover, we plan to apply VAEL to other settings, like structured object generation Liello et al. (2020), to showcase the flexibility and expressivity provided by the integration with a probabilistic logic program.

REFERENCES

Yoshua Bengio and Gary Marcus. Ai debate. https://montrealartificialintelligence.com/aidebate/, 2020.

Tarek R Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Daniel Lowd, Priscila Machado Vieira Lima, et al. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv preprint arXiv:1711.03902*, 2017.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pp. 2462–2467, 2007.

Luc De Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence. In *IJCAI*, 2020.

Reuben Feinman and Brenden M. Lake. Generating New Concepts with Hybrid Neuro-Symbolic Models. In *CogSci*, 2020.

Nishad Gothoskar, Marco Cusumano-Towner, Ben Zinberg, Matin Ghavamizadeh, Falk Pollok, Austin Garrett, Joshua B. Tenenbaum, Dan Gutfreund, and Vikash K. Mansinghka. 3DP3: 3D Scene Perception via Probabilistic Programming. In *NeurIPS*, 2021.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*, 2017.

Jindong Jiang and Sungjin Ahn. Generative Neurosymbolic Machines. In *NeurIPS*, 2020.

Tom Joy, Sebastian M. Schmon, Philip H. S. Torr, Siddharth Narayanaswamy, and Tom Rainforth. Capturing Label Characteristics in VAEs. In *ICLR*, 2021.

Henry Kautz. The third ai summer. https://roc-hci.com/announcements/the-third-ai-summer/, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014.

Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised Learning with Deep Generative Models. In *NeurIPS*, pp. 3581–3589, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324, 1998.

Luca Di Liello, Pierfrancesco Ardino, Jacopo Gobbi, Paolo Morettin, Stefano Teso, and Andrea Passerini. Efficient Generation of Structured Objects with Constrained Adversarial Networks. In *NeurIPS*, 2020.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural Probabilistic Logic Programming. In *NeurIPS*, pp. 3753–3763, 2018.

Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving. In *ICML*, 2020.

Thomas Winters, Giuseppe Marra, Robin Manhaeve, and Luc De Raedt. Deepstochlog: Neural stochastic logic programming. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9):10090–10100, Jun. 2022. doi: 10.1609/aaai.v36i9.21248. URL https://ojs.aaai.org/index.php/AAAI/article/view/21248.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *NeurIPS*, 2018.

A PROBABILISTIC LOGIC PROGRAMMING

A *logic program* is a set of *definite clauses*, i.e. expressions of the form $h \leftarrow b_1 \wedge ... \wedge b_n$, where h is the *head literal* or conclusion, while the b_i are *body literals* or conditions. Definite clauses can be seen as computational rules: IF all the body literals are true THEN the head literal is true. Definite clauses with no conditions ($n = 0$) are *facts*. In first-order logic programs, literals take the form $a(t_1, ..., t_m)$, with a a predicate of arity m and t_i are the terms, that is constants, variables or functors (i.e. functions of other terms). Grounding is the process of substituting all the variables in an atom or a clause with constants.

ProbLog De Raedt et al. (2007) lifts logic programs to *probabilistic logic programs* through the introduction of probabilistic facts. Whereas a fact in a logic program is deterministically true, a probabilistic fact is of the form $p_i :: f_i$ where f_i is a logical fact and p_i is a probability. In ProbLog, each ground instance of a probabilistic fact f_i corresponds to an *independent Boolean random variable* that is true with probability p_i and false with probability $1 - p_i$. Mutually exclusive facts can be defined through *annotated disjunctions* $p_0 :: f_0;\ ... ; p_n :: f_n.$ with $\sum_i p_i = 1$. Let us denote with $\mathcal{F}$ the set of all ground instances of probabilistic facts and with p their corresponding probabilities. Every subset $F \subseteq \mathcal{F}$ defines a *possible world* w_F obtained by adding to F all the atoms that can be derived from F using the logic program. The probability $P(w_F; p)$ of such a possible world w_F is given by the product of the probabilities of the truth values of the probabilistic facts; i.e:

$$P(w_F; p) = \prod_{f_i \in F} p_i \prod_{f_i \in \mathcal{F} \setminus F} (1 - p_i) \tag{3}$$

Two inference tasks on these probabilities are of interest for this paper.

Success: The probability of a query atom y, or formula, also called *success probability of* y, is the sum of the probabilities of all worlds where y is *True*, i.e.,

$$P(y; p) = \sum_{F \subseteq \mathcal{F} : w_F \models y} P(w_F; p) \tag{4}$$

Sample with evidence: Given a set of atoms or formulas E, the *evidence*, the probability of a world given evidence is:

$$P(w_F | E; p) = \frac{1}{Z} \begin{cases} P(w_F; p) & \text{if } w_F \models E \\ 0 & otherwise \end{cases} \tag{5}$$

where Z is a normalization constant. Sampling from this distribution provides only worlds that are coherent with the given evidence.

Example 2 (Addition of two digits) *Let us consider a setting where images contains two digits that can only be* 0 *or* 1*. Consider the following two logical predicates:* $\texttt{digit}(\texttt{img}, \texttt{I}, \texttt{Y})$ *states that a given image* `img` *has a certain digit* `Y` *in position* `I`*, while* $\texttt{add}(\texttt{img}, \texttt{z})$ *states that the digits in* `img` *sum to a certain value* z*.*

We can encode the digit addition task in the following program T*:*

```
p1::digit(img,1,0); p2::digit(img,1,1).
p3::digit(img,2,0); p4::digit(img,2,1).

add(img,Z) :- digit(img,1,Y1),
              digit(img,2,Y2),
              Z is Y1 + Y2.
```

In this program T*, the set of ground facts* $\mathcal{F}$ *is*

$$\{\texttt{digit}(\texttt{img}, 1, 0), \texttt{digit}(\texttt{img}, 1, 1), \texttt{digit}(\texttt{img}, 2, 0), \texttt{digit}(\texttt{img}, 2, 1)\}.$$

The set of probabilities p *is* $p = [p_1, p_2, p_3, p_4]$*. The ProbLog program* T *defines a probability distribution over the possible worlds and it is parameterized by p, i.e.* $P(\omega_F; p)$*. Then, we can ask ProbLog to compute the success probability of a query using Equation 4, e.g.* $P(\texttt{add}(\texttt{img}, 1))$*; or sample a possible world coherent with some evidence* $\texttt{add}(\texttt{img}, 2)$ *using Equation 5, e.g.* $\omega_F = \{\texttt{digit}(\texttt{img}, 1, 1), \texttt{digit}(\texttt{img}, 2, 1)\}$.

B ELBO DERIVATION

To derive the ELBO defined in (2) we start from the maximization of the log-likelihood of the input image x and the class y, namely

$$\log(p(x,y)) = \log\left(\int p(x,y|\mathbf{z})d\mathbf{z}\right). \tag{6}$$

Recalling the generative network factorization (1), we can write

$$\log(p(x,y)) = \log\left(\int p_\theta(x|z,z_{sym})p_\theta(y|z_{sym})p(z)p(z_{sym})dzdz_{sym}\right) \tag{7}$$

Then, by introducing the variational approximation $q_\phi(\mathbf{z}|x)$ to the intractable posterior $p_\theta(\mathbf{z}|x)$ and applying the factorization, we get

$$\log(p(x,y)) = \log\left(\int \frac{q_\phi(z|x)q_\phi(z_{sym}|x)}{q_\phi(z|x)q_\phi(z_{sym}|x)}p_\theta(x|z,z_{sym})p_\theta(y|z_{sym})p(z)p(z_{sym})dzdz_{sym}\right). \tag{8}$$

We now apply the *Jensen's inequality* to equation (8) and we obtain the lower bound for the log-likelihood of x and y given by

$$\int q_\phi(z|x)q_\phi(z_{sym}|x)\log\left(p_\theta(x|z,z_{sym})p_\theta(y|z_{sym})\frac{p(z)p(z_{sym})}{q_\phi(z|x)q_\phi(z_{sym}|x)}dzdz_{sym}\right). \tag{9}$$

Finally, by relying on the linearity of expectation and on logarithm properties, we can rewrite equation (9) as

$$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|x)}\left[\log(p_\theta(x|\mathbf{z}))\right] + \mathbb{E}_{z_{sym}\sim q_\phi(z_{sym}|x)}\left[\log(p_\theta(y|z_{sym}))\right] + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|x)}\left[\log\left(\frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|x)}\right)\right].$$

The last term is the negative Kullback-Leibler divergence between the variational approximation $q_\phi(\mathbf{z}|x)$ and the prior $p(\mathbf{z})$. This leads us to the ELBO of equation (2), that is

$$\begin{aligned}\log(p(x,y)) &\geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|x)}\left[\log(p_\theta(x|\mathbf{z}))\right] + \mathbb{E}_{z_{sym}\sim q_\phi(z_{sym}|x)}\left[\log(p_\theta(y|z_{sym}))\right] - \mathcal{D}_{\mathcal{KL}}[q_\phi(\mathbf{z}|x)||p(\mathbf{z})] \\ &:= \mathcal{L}(\theta,\phi).\end{aligned}$$

In VAEL graphical model (Figure 5), we omit ω_F since we exploit an equivalence relation between the probabilistic graphical models (PGMs) shown in Figure 5. Indeed, the objective for the PGM where ω_F is explicit is equivalent to the one reported in the paper. This is supported by the derivation of $\log p(x,y)$ (Eq. 10), which is equivalent to Eq. (2) in our paper, where the expectation over ω_F is estimated through Gumbel-Softmax.

$$\begin{aligned}\log p(x,y) &= \log\int_{z,z_{sym},\omega_F} q(z,z_{sym}|x)p(x|z,\omega_F)p(y|z_{sym})p(\omega_F|z_{sym},y)\frac{p(z,z_{sym})}{q(z,z_{sym}|x)} \\ &\geq \int_{z,z_{sym},\omega_F} q(z,z_{sym}|x)p(\omega_F|z_{sym},y)\log p(x|z,\omega_F)p(y|z_{sym})\frac{p(z,z_{sym})}{q(z,z_{sym}|x)} \\ &= \mathbb{E}_{z,z_{sym},\omega_F}[\log p(x|z,\omega_F)] + \mathbb{E}_{z_{sym}}[\log p(y|z_{sym})] - KL[q(z,z_{sym}|x)\|p(z,z_{sym})]\end{aligned} \tag{10}$$

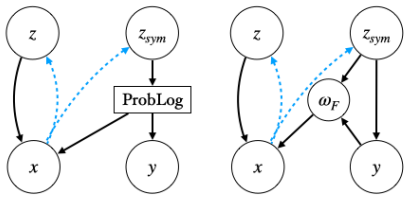

Figure 5: PGM with (left) and without (right) ProbLog box.

C ELBO ESTIMATION AND LEARNING

We estimate the ELBO and its gradients w.r.t. the model parameters using standard Monte Carlo estimates of expectations Kingma & Welling (2014). Since both $q_\phi(\mathbf{z}|x)$ and $p(\mathbf{z})$ are chosen to be Gaussian distributions, the Kullback-Leibler divergence in (2) can be integrated analytically by relying on its closed form. Thus, only the expected reconstruction and query errors $\mathcal{L}_{REC}(\theta,\phi)$ and $\mathcal{L}_Q(\theta,\phi)$ require estimation by sampling.
We can therefore define the ELBO estimator as

$$\mathcal{L}(\theta,\phi) \approx \tilde{\mathcal{L}}(\theta,\phi;\epsilon) = \tilde{\mathcal{L}}_{REC}(\theta,\phi;\epsilon) + \tilde{\mathcal{L}}_Q(\theta,\phi;\epsilon) - \mathcal{D}_{\mathcal{KL}}[q_\phi(\mathbf{z}|x)||p(\mathbf{z})]. \tag{11}$$

The estimators of $\mathcal{L}_{REC}$ and $\mathcal{L}_Q$ can be written as

$$\tilde{\mathcal{L}}_{REC}(\theta,\phi;\epsilon) = \frac{1}{N}\sum_{n=1}^{N}(\log(p_\theta(x|\hat{\mathbf{z}}^{(n)}))) \tag{12}$$

$$\tilde{\mathcal{L}}_Q(\theta,\phi;\epsilon) = \frac{1}{N}\sum_{n=1}^{N}(\log(p_\theta(y|\hat{z}^{(n)}_{sym}))) \tag{13}$$

where

$$\hat{\mathbf{z}}^{(n)} = \{\hat{z}^{(n)}, \hat{z}^{(n)}_{sym}\} := \mu(x) + \sigma(x)\epsilon^{(n)},$$
$$\epsilon^{(n)} \sim \mathcal{N}(0,1).$$

During the training, we aim at maximizing $\mathcal{L}(\theta,\phi)$ with respect to both the encoder and the decoder parameters, we therefore need to compute the gradient w.r.t. θ and ϕ. Since any sampling operation prevents back-propagation, we need to reparametrize the two sampled variables $\mathbf{z}$ and ω. Due to their nature, we use the well-known *Reparametrization Trick* Kingma & Welling (2014) for the Gaussian $\mathbf{z}$, while we exploit the *Categorical Reparametrization with Gumbel-Softmax* Jang et al. (2017) for the discrete variable ω corresponding to the sampled possible world.
In particular, by defining ω as the one-hot encoding of the possible worlds, we have

$$\hat{\omega}_i = \frac{\exp((\log\pi_i + \hat{g}_i)/\lambda}{\sum_{j=1}^{J}\exp((\log\pi_j + \hat{g}_j)/\lambda)}, \text{ with } \hat{g}_i \sim Gumbel(0,1) \tag{14}$$

where J is the number of possible worlds (e.g. all the possible pairs of digits), and π_i depends on z^i_{sym}, which is reparametrized with the Gaussian Reparametrization Trick. In Algorithm 1 we report VAEL training algorithm .

Algorithm 1: VAEL Training.

Data: Set of images $\mathcal{X}$
$\theta,\phi \leftarrow$ Initialization of paramters
repeat
  Forward Phase
  $x \leftarrow$ Training sample
  $\mathbf{z} = [z, z_{sym}] \sim q(\mathbf{z} \mid x)$
  $p = MLP(z_{sym})$
  $\omega_F \sim P(\omega_F; p)$
  $y \sim P(y; p)$
  $\tilde{x} \sim p(x|z,\omega_F)$
  Backward Phase
  $\mathbf{g} \leftarrow \nabla_{\theta,\phi}\mathcal{L}(\theta,\phi)$
  $\theta,\phi \leftarrow$ Update parameters using gradients $\mathbf{g}$
until *convergence of parameters* (θ,ϕ);

D IMPLEMENTATION DETAILS

D.1 VAEL

In Tables 2 and 3 we report the architectures of VAEL for *2digit MNIST* and *Mario* dataset. For both the datasets we performed a model selection by minimizing the objective function computed

on a validation set of $12,000$ samples for *2digit MNIST* and $2,016$ samples for *Mario*. In all the experiments we trained the model with Adam Kingma & Ba (2015). The explored hyper-parameters values are reported in Section D.4.

For *2digit MNIST*, the resulting best configuration is: latent space $z \in \mathbb{R}^M$, $z_{sym} \in \mathbb{R}^N$ with dimension $M = 8$ and $N = 15$; weights 0.1, 1×10^{-5} and 1.0 for the reconstruction, Kullback-Leibler and classification term of the ELBO respectively; learning rate 1×10^{-3}.

For *Mario*, we obtain: latent space $z \in \mathbb{R}^M$, $z_{sym} \in \mathbb{R}^N$ with dimension $M = 30$ and $N = 18$; weights 1×10^1, 1×10^1 and 1×10^4 for the reconstruction, Kullback-Leibler and classification term of the ELBO respectively; learning rate 1×10^{-4}.

Table 2: VAEL architectures for *2digit MNIST* dataset.

Encoder	**Decoder**
Input $28 \times 56 \times 1$ channel image	Input $\in \mathbb{R}^{M+20}$
$64 \times 1 \times 4 \times 4$ Conv2d stride 2 & ReLU	(M+20) $\times$ 256 Linear layer
$128 \times 64 \times 4 \times 4$ Conv2d stride 2 & ReLU	$256 \times 128 \times 5 \times 4$ ConvTranspose2d stride 2 & ReLU
$256 \times 128 \times 4 \times 4$ Conv2d stride 2 &ReLU	$128 \times 64 \times 4 \times 4$ ConvTranspose2d stride 2 & ReLU
256×2 (M+N) Linear layer	$1 \times 64 \times 4 \times 4$ ConvTranspose2d stride 2 & Sigmoid

MLP & ProbLog
Input $\in \mathbb{R}^N$
$N \times 20$ Linear layer & ReLU
20×20 Linear layer
ProbLog (IN dim: 20, OUT dim: 100)

Table 3: VAEL architectures for *Mario* dataset.

Encoder	**Decoder**
Input $200 \times 100 \times 3$ channel image	Input $\in \mathbb{R}^{M+9}$
$64 \times 3 \times 5 \times 5$ Conv2d stride 2 & SELU	(M+9) $\times$ 512 Linear layer
$128 \times 64 \times 5 \times 5$ Conv2d stride 2 & SELU	$512 \times 256 \times 5 \times 5$ ConvTranspose2d stride 2 & SELU
$256 \times 128 \times 5 \times 5$ Conv2d stride 2 & SELU	$256 \times 128 \times 5 \times 5$ ConvTranspose2d stride 2 & SELU
$512 \times 256 \times 5 \times 5$ Conv2d stride 2 & SELU	$128 \times 64 \times 5 \times 5$ ConvTranspose2d stride 2 & SELU
512×2 (M+9) Linear layer	$3 \times 64 \times 5 \times 5$ ConvTranspose2d stride 2 & Sigmoid

MLP & ProbLog
Input $\in \mathbb{R}^N$
$N \times 20$ Linear layer & ReLU
20×9 Linear layer
ProbLog (IN dim: 18, OUT dim: 24)

D.2 CCVAE

In the original paper Joy et al. (2021), there was a direct supervision on each single element of the latent space. To preserve the same type of supervision in our two digits addition task, where the supervision is on the sum and not directly on the single digits, we slightly modify the encoder and decoder mapping functions of CCVAE. By doing so, we ensure the correctness of the approach without changing the graphical model. The original encoder function learns from the input both the mean μ and the variance σ of the latent space distribution, while the decoder gets in input the latent representation $\mathbf{z} = \{z_{sym}, z\}$ (please refer to the original paper for more details Joy et al. (2021)). In our modified version, the encoder only learns the variance, while the mean is set to be equal to the image label $\mu = y$, and the decoder gets in input the label directly $\mathbf{z}^* := \{y, z\}$.

In Tables 4 and 5 we report the architectures of CCVAE for *2digit MNIST* and *Mario* dataset. For both the datasets we performed a model selection by minimizing the objective function computed on a validation set of $12,000$ samples for *2digit MNIST* and $2,016$ samples for *Mario*. In all the experiments we trained the model with Adam Kingma & Ba (2015). The explored hyper-parameters values are reported in Section D.4.

For *2digit MNIST*, the resulting best configuration is: latent space $z_{sym} \in \mathbb{R}^N$ with dimension equal to the number of classes $N = 19$ (due to the one-to-one mapping between z_{sym} and the label y); latent space $z \in \mathbb{R}^M$ with dimension $M = 8$, model objective reconstruction term with weight 0.05, while the other ELBO terms with unitary weights; learning rate 1×10^{-4}.

For *Mario*, we obtain: latent space $z_{sym} \in \mathbb{R}^N$ with dimension equal to the number of classes $N = 4$; latent space $z \in \mathbb{R}^M$ with dimension $M = 300$, model objective Kullback-Leibler term and classification term with weight 1×10^4 and 1×10^3 respectively, while the other ELBO terms with unitary weights; learning rate 1×10^{-4}.

Table 4: CCVAE architectures for *2digit MNIST* dataset.

Encoder	**Decoder**
Input $28 \times 56 \times 1$ channel image	Input $\in \mathbb{R}^{M+N}$
$64 \times 1 \times 4 \times 4$ Conv2d stride 2 & ReLU	(M+N) $\times$ 256 Linear layer
$128 \times 64 \times 4 \times 4$ Conv2d stride 2 & ReLU	$256 \times 128 \times 5 \times 4$ ConvTranspose2d stride 2 & ReLU
$256 \times 128 \times 4 \times 4$ Conv2d stride 2 &ReLU	$128 \times 64 \times 4 \times 4$ ConvTranspose2d stride 2 & ReLU
256×2 (M+N) Linear layer	$1 \times 64 \times 4 \times 4$ ConvTranspose2d stride 2 & Sigmoid

Table 5: CCVAE architectures for *Mario* dataset.

Encoder	**Decoder**
Input $200 \times 100 \times 3$ channel image	Input $\in \mathbb{R}^{M+N}$
$64 \times 3 \times 5 \times 5$ Conv2d stride 2 & SELU	(M+N) $\times$ 512 Linear layer
$128 \times 64 \times 5 \times 5$ Conv2d stride 2 & SELU	$512 \times 256 \times 5 \times 5$ ConvTranspose2d stride 2 & SELU
$256 \times 128 \times 5 \times 5$ Conv2d stride 2 &SELU	$256 \times 128 \times 5 \times 5$ ConvTranspose2d stride 2 & SELU
$512 \times 256 \times 5 \times 5$ Conv2d stride 2 &SELU	$128 \times 64 \times 5 \times 5$ ConvTranspose2d stride 2 & SELU
512×2 (M+N) Linear layer	$3 \times 64 \times 5 \times 5$ ConvTranspose2d stride 2 & Sigmoid

D.3 CLASSIFIERS

In Table 6 we report the architecture of the classifier used to measure the generative ability of VAEL and CCVAE for *2digit MNIST* dataset. We trained the classifier on $60,000$ MNIST images LeCun et al. (1998) for 15 epochs with SGD with a learning rate of 1×10^{-2} and a momentum of 0.5, achieving 0.97 accuracy on the test set.

Table 6

MNIST classifier (*2digit MNIST*)
Input $28 \times 28 \times 1$ channel image
Linear layer 784×128 & ReLU
Linear layer 128×64 & ReLU
Linear layer 64×10 & LogSoftmax

In Table 7 we report the architecture of the classifier used to measure the generative ability of VAEL and CCVAE for *Mario* dataset. We trained the classifier on $9,140$ single state images of *Mario* dataset for 10 epochs with Adam Kingma & Ba (2015) optimizer with a learning rate of 1×10^{-4}, achieving 1.0 accuracy on the test set.

D.4 OPTIMIZATION

Experiments are conducted on a single Nvidia GeForce 2080ti 11 GB. Training consumed $\sim 2GB$ for *2digit MNIST* dataset and $\sim 2.8GB$ for *Mario* dataset, taking around 1 hour and 15 minutes to complete 100 epochs for *2digit MNIST* and 1 hour and 30 minutes to complete 100 epochs for *Mario* dataset. As introduced in the previous sections, we performed a model selection based on ELBO minimization for both the model.

In the following bullet lists, lr refers to the learning rate, z, z_{sym} refer to the latent vectors dimensions, W_{REC}, W_{KL}, W_Q refer to the weights of $\mathcal{L}_{REC}, \mathcal{D}_{KL}, \mathcal{L}_Q$ terms of VAEL objective function, and

Table 7

MNIST classifier (*Mario*)
Input $100 \times 100 \times 3$ channels image
Conv layer $5 \times 5 \times 32$ & SELU
Conv layer $5 \times 5 \times 64$ & SELU
Conv layer $5 \times 5 \times 128$ & SELU
Linear layer 2048×9

$W_{REC}, W_{KL}, W_{q(y|z_{sym})}, W_{q(y|x)}$ refer to the corresponding terms of CCVAE objective function (please refer to the original paper for more details Joy et al. (2021)).

For *2digit MNIST* we explore the following values; we repeat the model training 5 times for each configuration.

- VAEL
  - $z \in \{8, 9, 10\}$
  - $z_{sym} \in \{15, 19\}$
  - $lr \in \{0.0001, 0.001\}$
  - $W_{REC} \in \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$
  - $W_{KL} \in \{0.00001, 0.0001, 0.001\}$
  - $W_Q \in \{1, 5\}$
- CCVAE
  - $z_{sym} \in \{8, 10, 15, 20, 30\}$
  - $lr \in \{0.00001, 0.0001\}$
  - $W_{KL} \in \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$
  - $W_{REC} \in \{0.01, 0.1, 1, 10, 100\}$
  - $W_{q(y|z_{sym})} \in \{0.01, 0.1, 1, 10, 100\}$
  - $W_{q(y|x)} \in \{0.01, 0.1, 1, 10, 100\}$

For *Mario* we explore the following values; we repeat the model training 5 times for each configuration.

- VAEL
  - $z \in \{20, 25, 30, 35, 40\}$
  - $z_{sym} \in \{18, 20\}$
  - $lr \in \{0.0001, 0.0005\}$
  - $W_{REC} \in \{1, 10\}$
  - $W_{KL} \in \{0.1, 1, 10\}$
  - $W_Q \in \{1, 100, 10000\}$
- CCVAE
  - $z_{sym} \in \{3, 4, 5, 10, 20, 30, 40, 50, 100, 200, 300, 400\}$
  - $lr \in \{0.0001, 0.0005\}$
  - $W_{KL} \in \{0.0, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000\}$
  - $W_{REC} \in \{1, 10\}$
  - $W_{q(y|z_{sym})} \in \{1, 10, 100\}$
  - $W_{q(y|x)} \in \{1, 10, 100, 1000\}$

E DATA EFFICIENCY: SIMPLIFIED SETTING

To further investigate the performance gap between CCVAE and VAEL in the *Data Efficiency* task described in Section 3, we run an identical experiment in a simplified dataset with only three possible

digits values: 0, 1 and 2. The goal is to train CCVAE on a much larger number of images per pair, which is impractical in the 10-digits setting, due to the combinatorial nature of the task. The dataset consists of $30,000$ images of two digits taken from the MNIST dataset LeCun et al. (1998). We use 80%, 10%, 10% splits for the train, validation and test sets, respectively. As for the 10-digits dataset, each image in the dataset has dimension 28×56 and is labelled with the sum of the two digits. In Figure 6 we compare VAEL and CCVAE discriminative, generative and reconstructive ability when varying the training size. In this simplified setting, CCVAE requires around $3,000$ samples per pair to reach the accuracy that VAEL achieves trained with only 10 samples per pair.

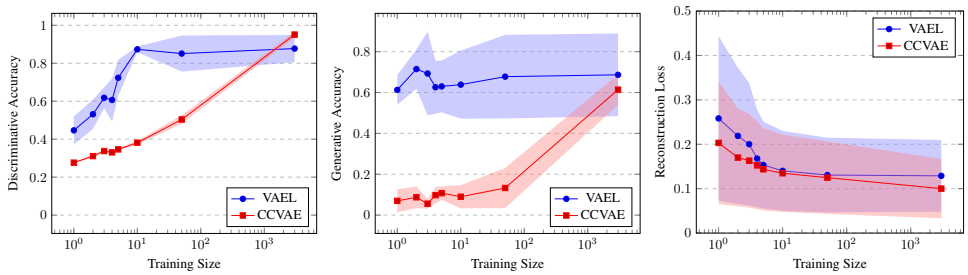

Figure 6: Discriminative, generative and reconstructive ability of VAEL (**blue**) and CCVAE (**red**) trained in contexts characterized by data scarcity. Both the models are evaluated on the same test set. The training size refers to the number of samples per pair of digits seen during the training.

F Additional Results

Here we report some additional results for the tasks described in Section 3.

Figures 7 and 8 show additional qualitative results for the *Conditional Image Generation* and *Task Generalization* experiments relative to *2digit MNIST* dataset.

In Figures 9 and 10, we report some additional examples of *Image Generation* and *Task Generalization* for *Mario* dataset. As it can be seen in Figure 10, VAEL is able to generate subsequent states consistent with the shortest path, whatever the agent's position in the initial state ($t = 0$). Moreover, the model generates states that are consistent with the initial one in terms of background.

Figure 11 shows some examples of image reconstruction for CCVAE. As it can be seen, CCVAE focuses only on reconstructing the background and discards the small portion of the image containing the agent, thus causing the disparity in the reconstructive and generative ability between VAEL and CCVAE (Table 1).

Figure 7: Conditional generation for CCVAE and VAEL for *2digit MNIST* dataset. In each column the generation is conditioned on a different sum y between the two digits.

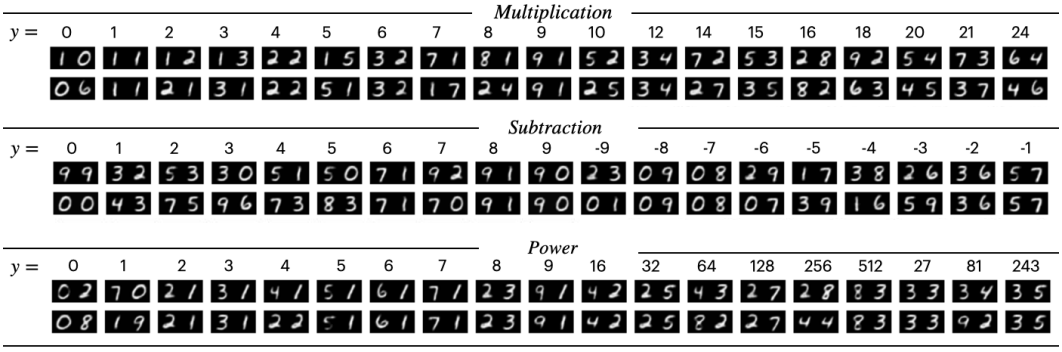

Figure 8: Examples of the generation ability of VAEL in 3 previously unseen tasks for *2digit MNIST* dataset. In each column the generation is conditioned on a different label y referring to the corresponding mathematical operation between the first and second digit.

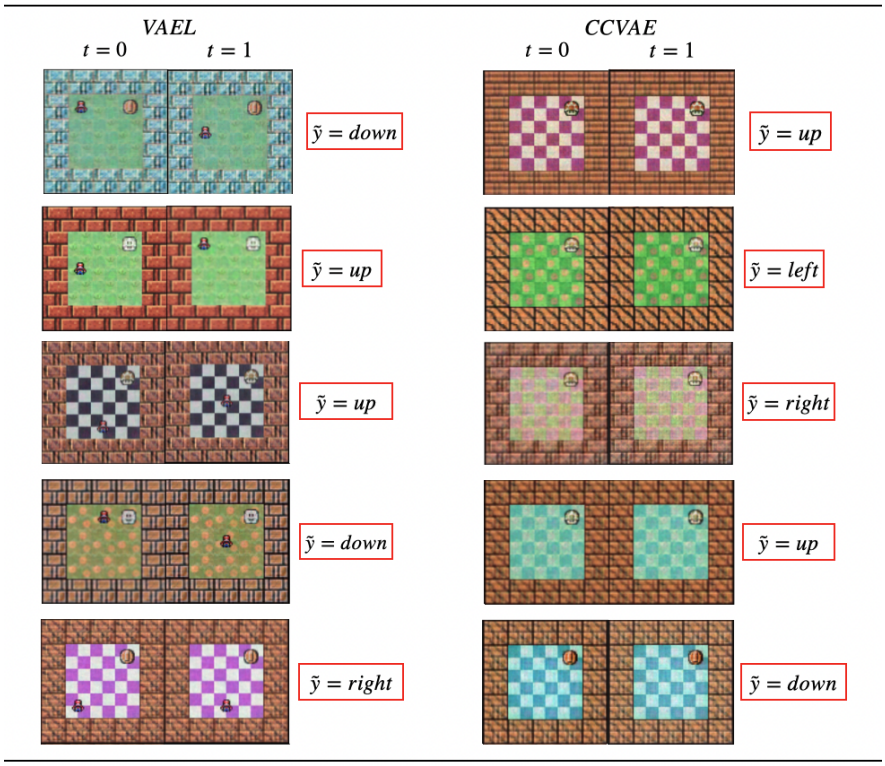

Figure 9: Examples of the generation ability of CCVAE and VAEL for *Mario* dataset.

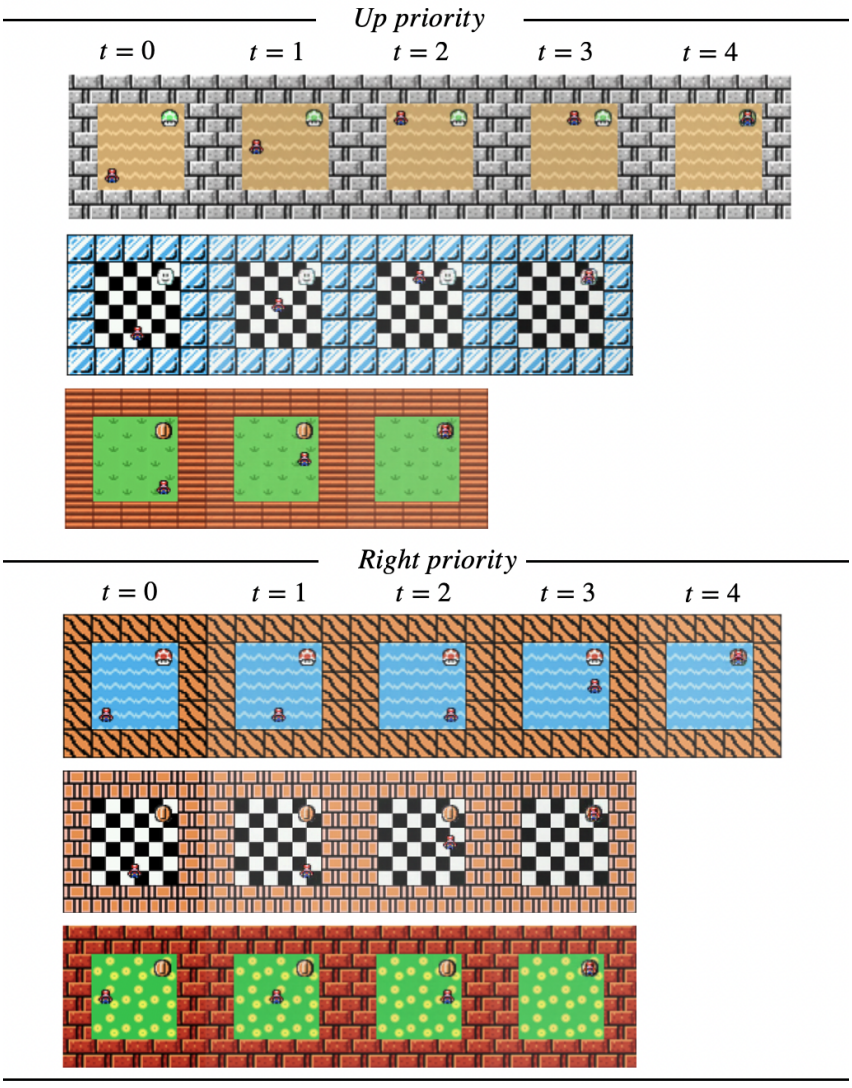

Figure 10: Examples of the generation ability of VAEL in previously unseen tasks for *Mario* dataset. In each row, VAEL generates a trajectory starting from the initial image (t = 0) and following the shortest path using an *up priority* or a *right priority*.

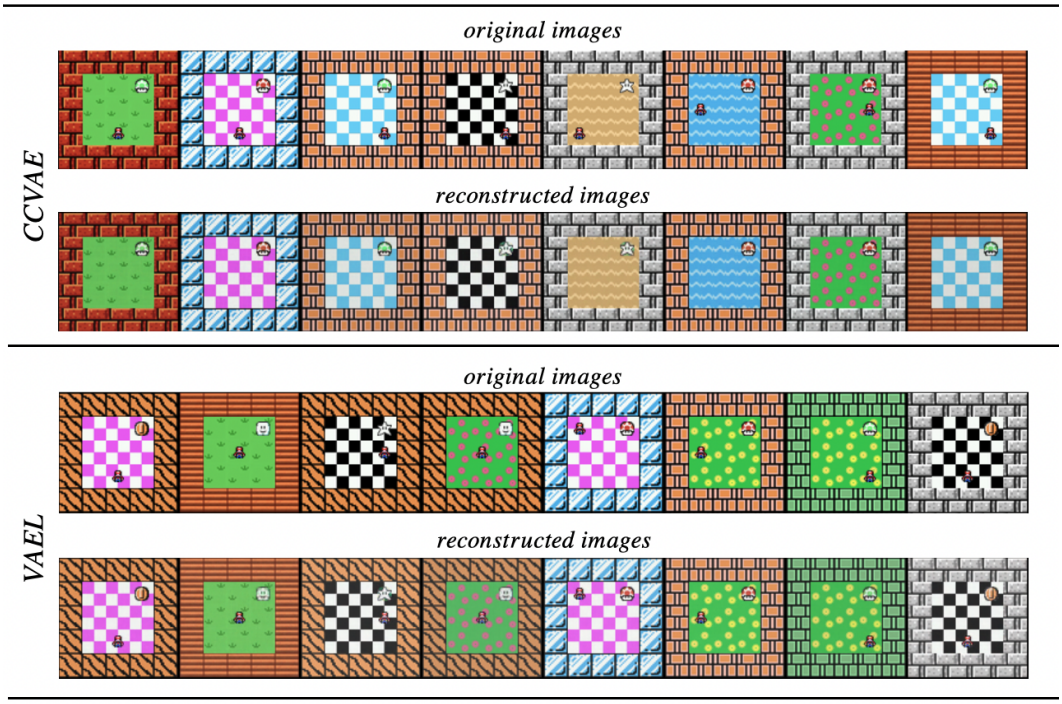

Figure 11: Examples of reconstructive ability of CCVAE and VAEL trained on *Mario* dataset.