A ELBO derivation

To derive the ELBO defined in (5) 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 (4), 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 (5), 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 2c), we omit ω_F since we exploit an equivalence relation between the probabilistic graphical models (PGMs) shown in Figure 9. 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. (5) in our paper, where the expectation over ω_F is estimated through Gumbel-Softmax.

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

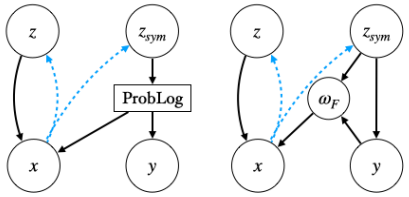


Figure 9: PGM with (left) and without (right) ProbLog box.

B ELBO estimation and Learning

We estimate the ELBO and its gradients w.r.t. the model parameters using standard Monte Carlo estimates of expectations [36]. Since both $q_\phi(\mathbf{z}|x)$ and $p(\mathbf{z})$ are chosen to be Gaussian distributions, the Kullback-Leibler divergence in (5) 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

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

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

where

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

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* [36] for the Gaussian $\mathbf{z}$, while we exploit the *Categorical Reparametrization with Gumbel-Softmax* [27] 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_{sym}^i, 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);

C Additional supervision for MNIST Task Generalization

During the training on *2digit MNIST* dataset, the model may learn a mapping between symbol and meaning that is logically correct, but different from the desired one. Indeed, the two symbols 1 and

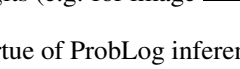
2 used for the left and right positions, respectively, of a handwritten digit in an image are just an assumption. However, VAEL may switch the pairs $(3, 2)$ and $(2, 3)$, since they both sum up to 5. This would prevent VAEL from generalizing to tasks involving non-commutative operations (i.e. *subtraction* and *power*).

To solve this issue, we simply introduce additional supervision on the digits of very few images (1 image per pair of digits, i.e. 100 images in total) to guide the model toward the desired symbols interpretation. This has to be intended just as an agreement between the model and the human. To include this supervision in the training procedure, we add a regularizer term to the ELBO defined in (5), namely

$$\mathcal{L}_{\mathcal{SUP}}(\theta, \phi) := \mathcal{L}(\theta, \phi) + \mathcal{L}_{digits}(\theta, \phi) \tag{15}$$

where

$$\mathcal{L}_{digits}(\theta, \phi) = \mathbb{E}_{z_{sym} \sim q_\phi(z_{sym}|x)}[log(p_\theta(y_{digits}|z_{sym})]]. \tag{16}$$

In equation (16), y_{digits} refers to the labels over the digits (e.g. for image  we have $y_{digits} = [0, 1]$).
Such a digit-level supervision can be easily done by virtue of ProbLog inference, that allows us to retrieve the predicted label of each digit in an image by relying on the query over the digits values.

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 [35]. 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
$256 \times 2(M + N)$ 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)

D.2 CCVAE

In the original paper [31], 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

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
$512 \times 2(M+9)$ 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)

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 [31]). 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 [35]. 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
$256 \times 2(M+N)$ 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
$512 \times 2(M+N)$ 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 [40] 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 [35] optimizer with a learning rate of 1×10^{-4}, achieving 1.0 accuracy on the test set.

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

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 $W_{REC}, W_{KL}, W_{q(y|z_{sym})}, W_{q(y|x)}$ refer to the corresponding terms of CCVAE objective function (please refer to the original paper for more details [31]).

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 Additional Results

Here we report some additional results for the tasks described in Section 4.

Figures 10 and 11 show additional qualitative results for the *Conditional Image Generation* and *Task Generalization* experiments relative to *2digit MNIST* dataset.

In Figures 12 and 13, we report some additional examples of *Image Generation* and *Task Generalization* for *Mario* dataset. As it can be seen in Figure 13, 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 14 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).

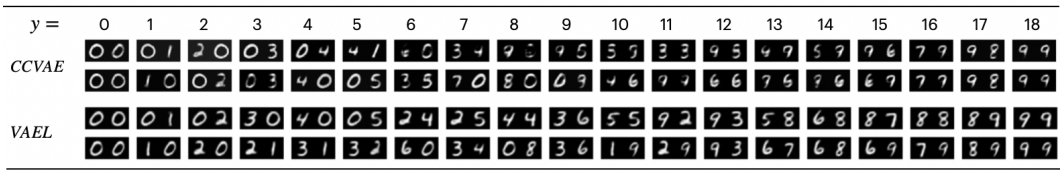


Figure 10: Conditional generation for CCVAE and VAEL for *2digit MNIST* dataset. In each column the generation is conditioned on a different sum y between the two digits.

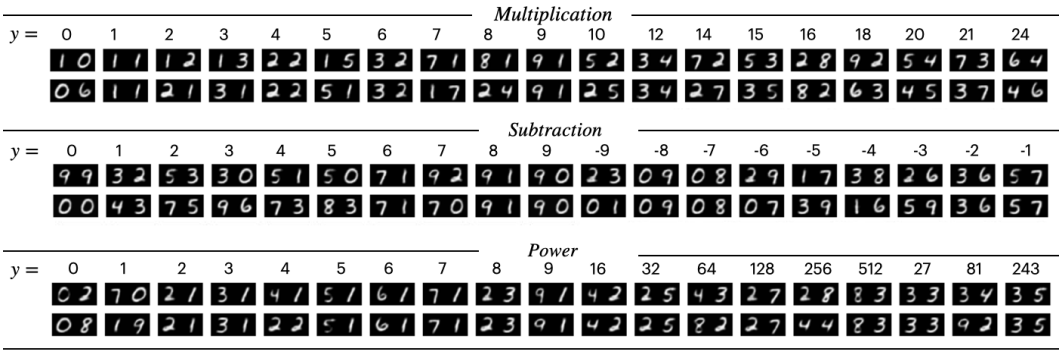


Figure 11: Examples of the generation ability of VAEL in 3 previously unseen tasks for *2digit MNIST* dataset. In each column the generation is conditioned on a different label y referring to the corresponding mathematical operation between the first and second digit.

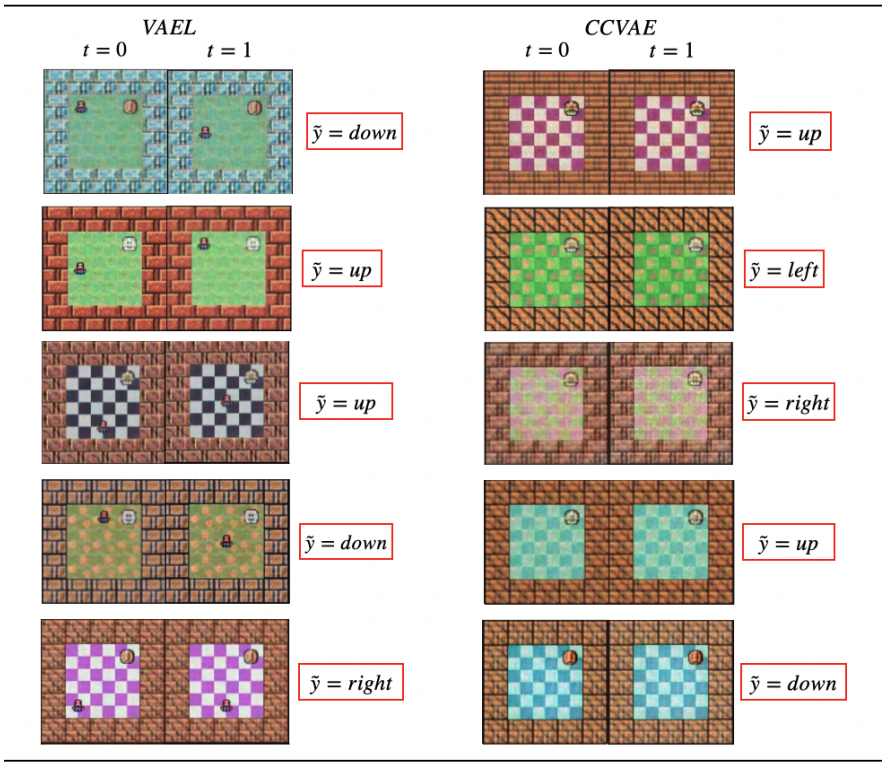


Figure 12: Examples of the generation ability of CCVAE and VAEL for *Mario* dataset.

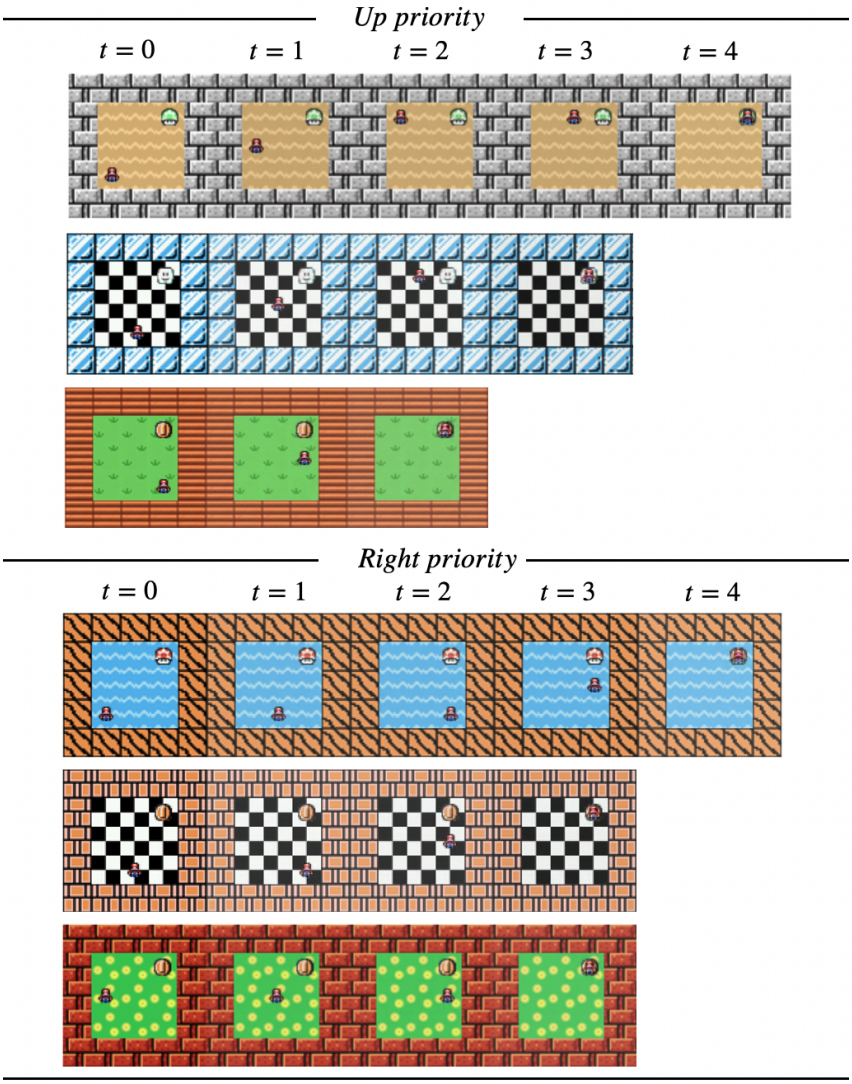


Figure 13: Examples of the generation ability of VAEL in previously unseen tasks for *Mario* dataset. In each row, VAEL generates a trajectory starting from the initial image (t = 0) and following the shortest path using an *up priority* or a *right priority*.

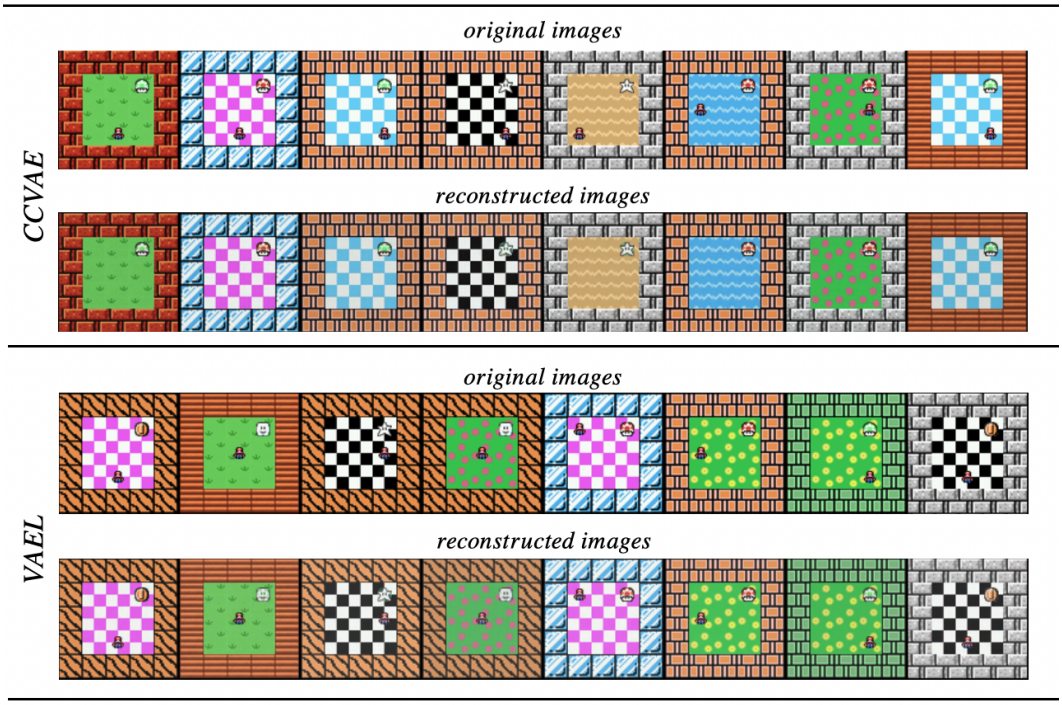


Figure 14: Examples of reconstructive ability of CCVAE and VAEL trained on *Mario* dataset.

F Data Efficiency: simplified setting

To further investigate the performance gap between CCVAE and VAEL in the *Data Efficiency* task 4, 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 [40]. 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 15 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.

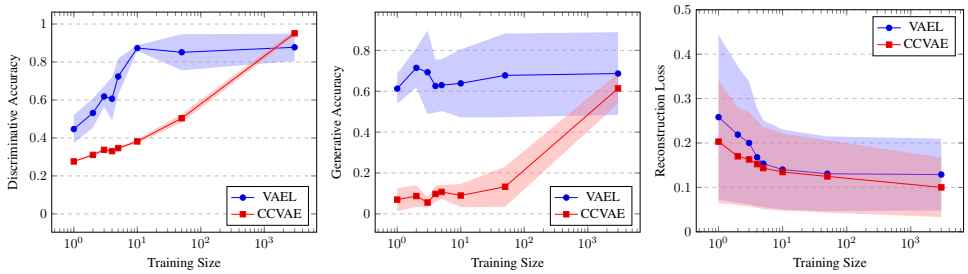


Figure 15: Discriminative, generative and reconstructive ability of VAEL (**blue**) and CCVAE (**red**) trained in contexts characterized by data scarcity. Both the models are evaluated on the same test set. The training size refers to the number of samples per pair of digits seen during the training.