State Representations in Robotics: Identifying Relevant Factors of Variation using Weak Supervision

Constantinos Chamzas[*,†] chamzas@rice.edu
Martina Lippi[*,‡] martina.lippi@uniroma3.it
Michael C. Welle[*,§] mwelle@kth.se
Anastasiia Varava[§] varava@kth.se
Alessandro Marino[¶] al.marino@unicas.it
Lydia E. Kavraki[†] kavraki@rice.edu
Danica Kragic[§] dani@kth.se

Abstract

Representation learning allows planning actions directly from raw observations. Variational Autoencoders (VAEs) and their modifications are often used to learn latent state representations from high-dimensional observations such as images of the scene. This approach uses the similarity between observations in the space of images as a proxy for estimating similarity between the underlying states of the system. We argue that, despite some successful implementations, this approach is not applicable in the general case where observations contain task-irrelevant factors of variation. We compare different methods to learn latent representations for a box stacking task and show that models with weak supervision such as Siamese networks with a simple contrastive loss produce more useful representations than traditionally used autoencoders for the final downstream manipulation task.

1 Introduction and Related Work

Representation learning aims to infer abstract features that describe data. State representation learning [1] can be seen as a special case of representation learning, where the features are low-dimensional encodings characterizing states of a dynamical system. These features change over time as a result of the actions of an agent, such as in robotics and reinforcement learning scenarios. Low-dimensional representations help to overcome the curse of dimensionality and make it easier to control the system through actions by reducing the search space for planning or policy search.

In robotics and reinforcement learning, the idea of learning low-dimensional state representations from high-dimensional observations (such as images) in an unsupervised manner recently gained significant attention. For this, latent space models, such as (variational) autoencoders (VAE), are commonly used. For complex robotic manipulation, this approach is considered particularly promising since analytic approaches can be computationally expensive and complex. In particular, variational autoencoders have been successfully employed to generate a latent space that was subsequently used to perform robotic tasks.The authors of [2] use an autoencoder to project a high-dimensional state to a low dimensional latent space in which planning is performed. This latent plan is later decoded back to the original space through the decoder. Similarly, [3] use a variational autoencoder to generate 3D end-effector poses for grasping.

Despite some successful applications, such approaches are based on an implicit assumption that similar observations that are close in the image space correspond to similar states of the system, as the objective of autoencoders is to optimize the reconstruction loss. While this assumption is

[*]Contributed equally, listed in alphabetical order.
[†]Rice University
[‡]University of Roma Tre
[§] KTH Royal Institute of Technology
[¶] University of Cassino and Southern Lazio

NeurIPS 2020 3rd Robot Learning Workshop: Grounding Machine Learning Development in the Real World.

reasonable in some settings, we argue that in general, very different images can correspond to the same underlying state when other factors of variation, that are not relevant for the task, are present in the observations. Examples of these factors are changes in the lighting conditions, camera position, and occlusions in the scene.

To mitigate this effect, it is crucial to identify the *task-relevant* factors of variation. While some steps in this direction have been made, for instance, through considering independently controllable factors of variation [4] ensuring that only chosen factors of variation are being changed in the process of data collection, designing policies for that remains a challenge: for instance, if a robot is operating in a cluttered environment, moving one object can induce complex interactions with the surrounding objects; furthermore, if the data is being collected on real hardware, irrelevant external factors can be difficult to control and, ideally, need to be ignored.

In [5], this problem is addressed by incorporating weak supervision into the training dataset: the authors execute a small number of single actions on a robot and collect the observations before and after performing an action. This information is used to identify observations corresponding to different states: a pair of observations is considered different if there is an action performed between them. Furthermore, the recorded actions are used to build a roadmap in the latent space, allowing to capture the global dynamics of the system with relatively few observations, in contrast to [6] and [7], where a significant amount of training data is required to learn dynamics directly from observations.

In the present work, we investigate how this sparse information can be used to learn useful state representations. We consider a dataset where different task-irrelevant factors are present implying that changes in images are only weakly correlated with changes in the states. This is achieved by using different viewpoints of the same scene and randomizing the color of irrelevant parts of the scene as well as the illumination. Subsequently, we evaluate state space representations produced by three different models on a downstream manipulation task.

We show that *(i)* a significant performance improvement is achieved when weak supervision is included in the loss function and *(ii)*, when task-irrelevant factors of variation are present in the dataset, VAE-based encodings are significantly worse than models that leverage task-related information about state similarity. Furthermore, we show that a Siamese network with a much simpler architecture produces better representations and is significantly easier to train than VAE-based models.

2 Problem Description

We consider a robotic manipulation task for which a latent representation is built from images. As an example, the task of stacking boxes in a simulated environment similar to [5] is considered.

Box stacking task: The task is to find a feasible plan given a start and a goal image. To perform the task, it is desirable that the resulting latent structure is easy to exploit by simple clustering and graph building algorithms. We consider the arrangement of boxes in the scenes as underlying states, thus leading to exactly 12 possible states of the system. In this task, the underlying state of the system which we aim to capture with the latent representation, does not change if, for instance, different viewpoints of the camera, different background and table color, as well as different levels of illumination are used to record the scene. For more information about the task and the stacking rules see Appendix 5.4.

Datasets: Three different viewpoints, as shown in Fig. 1, are considered to capture the scene. such as lighting conditions or different viewpoints, have changed. The four datasets are built as follows: *i)* $\mathcal{D}_f$ where both the observations I_1 and I_2 in each triplet are taken from view front, *ii)* $\mathcal{D}_r$ where the observations are only taken from view right, *iii)* $\mathcal{D}_l$ where the observations are only taken from view left, and *iv)* $\mathcal{D}_m$ where we enforce that the views for I_1 and I_2 are different. The datasets are thus defined in such a way to allow the investigation of the effects of task-irrelevant factors in the observations. We use the simulation environment Unity [8] for generating the datasets. For each dataset, we use 2500 data samples for training, with 1598 action-pairs ($a = 1$), and we generate 2500 more data samples as holdout set for evaluation.

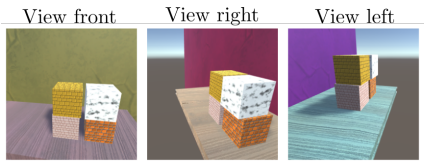

Figure 1: Examples of observations from three different viewpoints used for building the datasets.

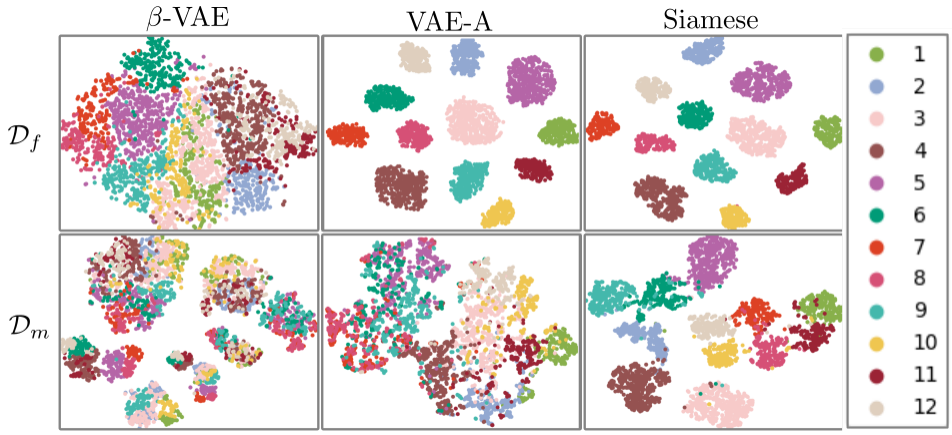

Figure 2: t-SNE plots for 2000 encoded observations form all four models latent space using datasets $\mathcal{D}_f$ (top row) and $\mathcal{D}_m$ (bottom row).

Models: We tested three different models each learning a 12 dimensional latent space as the underlying state: *i) β-VAE:* standard β-VAE [9] with reconstruction loss and KL-divergence term encouraging a compact latent space representation. *ii) VAE-A:* β-VAE combined with a contrastive loss, called action loss in [5], that encourages the encoding of actions pairs at a certain distance while minimising the distance between no-action pairs. *iii) Siamese:* a Siamese network with a pairwise contrastive loss [10] which structures the latent space such that it minimizes the pairwise euclidean distance between similar pairs and increases it between dissimilar pairs. As similar pairs the no-action ($a = 0$) pairs are considered while for dissimilar the action pairs were used ($a = 1$). The considered models thus allow to compare the results when the latent representation is obtained by heavily using the reconstruction loss (β-VAE) and when it is obtained using a combination of reconstruction and contrastive losses (VAE-A) or using only a contrastive loss (Siamese) that does not have any reconstructive term. For more details on the architecture, and loss functions, as well as the chosen hyperparameters refer to the Appendix 5.1. However, note that a much simpler architecture is used for the Siamese network compared to the VAE ones.

Connecting clusters with actions: We adopt the Latent Space Roadmap (LSR) method from [5] to build a graph structure in the latent space that can be used for planning purposes. The basic idea is to perform clustering in the latent space and connect the clusters in the case a transition between them is allowed. For this, we build a graph where each node is associated with a cluster and each edge with a possible transition. Given this graph, it is possible to find the shortest paths on it that lead from the encoded start state to the encoded goal state. In the case of perfect state space representation, the number of clusters should be equal to the number of possible states of the system while the number of edges should be equal to the number of possible transitions between states. In this work, we use HDBSCAN [11] for clustering that only takes the minimum cluster size as input parameter. For more details about the building of the Latent Space Roadmap see [5], and Appendix 5.2. Note that the objective of the planning process is to produce plans of *actions* that lead from start to goal state. To this aim, no decoded images are needed and the plans can be found through the LSR in the latent space generated by any model (either based on VAE or Siamese networks).

3 Experiments

In this section we evaluate the latent representations obtained with the different models on the different datasets. We consider the following criteria:

1. Number of clusters, which should ideally be equal to the number of possible system states (12 in our stacking scenario);
2. Homogeneity of clusters, with range $[0, 1]$, defined as the average ratio of samples associated with the same class compared to the total number of samples in each cluster;
3. Number of edges built between clusters adopting the graph building method from [5]. The optimal number of edges is 24;
4. Correctness of the edges, with range $[0, 1]$, defined as the number of edges that fulfill the stacking rules divided by the total number of edges;

5. Performance of planning 1000 paths between random novel start and goal observations. More specifically, we evaluate the percentage that all found paths are correct and the percentage if at least one path is correct.

Models	Dataset $\mathcal{D}_f$						Dataset $\mathcal{D}_m$					
	Clust. num.	Clust. hom.	Edges num.	Edge corr.	Paths scores		Clust. num.	Clust. hom.	Edges num.	Edge corr.	Paths scores	
					% all	% any					% all	% any
β-VAE	715	0.98	617	0.95	8.05	8.3	757	**0.95**	644	**0.93**	1.0	1.0
VAE-A	**12**	**1.0**	**24**	**1.0**	**100.0**	**100.0**	87	0.86	91	0.85	17.89	20.3
Siamese	**12**	**1.0**	25	0.96	94.21	98.1	**25**	**0.95**	**64**	0.88	**49.47**	**52.6**

Table 1: Results for the four models (β-VAE, VAE-A and Siamese) on $\mathcal{D}_f$ and $\mathcal{D}_m$. We report the number of clusters and their homogeneity, the number of edges and their correctness score, and the planning performance in terms of percentage when all found paths are correct and percentage if at least one path is correct. Best results in bold.

Table 1 reports the experimental results obtained with datasets $\mathcal{D}_f$ (on the left) and $\mathcal{D}_m$ (on the right). Note that dataset $\mathcal{D}_f$ fulfills the general assumption that changes in the observations significantly correlate with changes in the system states since it only uses the frontal view in Fig. 1. This is not verified instead in dataset $\mathcal{D}_m$ where different viewpoints are considered. The results show that, for dataset $\mathcal{D}_f$, the β-VAE leads to a very high number of clusters (715 nodes) indicating that the latent space structure is very fragmented. Based on these clusters, a high number of edges is built and poor planning performance of $\approx 8\%$ is achieved for the correctness of all found paths. Perfect performance is then achieved by the VAE-A model, which reaches the optimal number of clusters and edges as well as a perfect score for homogeneity and edges correctness. This result is in line with our expectations as, for this dataset, the VAE-A combines the best of two worlds: it makes use of the correlation between changes in observations and in the respective underlying states with the reconstruction loss, while also incorporating the weak knowledge of actions between states with the contrastive loss. This however comes at the cost of two additional hyperparameters that need to be tuned. while not perfect, very good results regarding clusters, edges , and planning are also achieved by the Siamese models.

For dataset $\mathcal{D}_m$, we observe that a significant decrease of performance is obtained with all the models, proving its higher representation complexity with respect to $\mathcal{D}_f$. More specifically, the β-VAE fails completely in all the evaluation criteria reaching 1% for the planning performance. Better structuring is achieved by the informed VAE-A which gives $\approx 18\%$ for the correctness of all the paths. However, the large number of clusters with low homogeneity (0.86) prevents better planing performance. Finally, the Siamese network reports the best overall performance as it builds the fewest number of clusters and edges as well as achieves the best performance of $\approx 49\%$ for the correctness of all paths. The results thus confirm that the absence of a reconstruction loss can improve the performance when the changes in the system observations are not necessarily correlated with changes in the underlying states.

Fig. 2 shows a qualitative visualization of the structure of the latent spaces using t-SNE [12]. The figure reports the t-SNE plots applied to 2500 representations obtained using β-VAE, VAE-A and Siamese on datasets $\mathcal{D}_f$ (top row) and $\mathcal{D}_m$ (bottom row). The figure thus confirms the previous findings showing that the structure generated by the models using the weak supervision is clearly superior to the basic β-VAE for which very fragmented clusters are obtained. It is also evident that the VAE-A does not succeed in structuring the latent representation as much as the Siamese model. The results associated with datasets $\mathcal{D}_r$ and $\mathcal{D}_l$ are reported in the Appendix 5.3.

4 Conclusion

In this work, we argue that autoencoder-based state representations are generally not ideal for robotics, as they fail to capture the true similarity between states when irrelevant factors of variation are present in the observations. We train three models that are based on a VAE and a Siamese network given a set of observations where visually different images correspond to the same state, and demonstrate that a weak supervision signal in the form of a relatively small set of action-separated observations significantly improves the quality of the generated latent state space. We argue that training a variational autoencoder to extract state representations is more computationally expensive and may lead to worse state representations with respect to the overall downstream task.

References

[1] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

[2] Brian Ichter and Marco Pavone. Robot motion planning in learned latent spaces. *IEEE Robot. Autom. Letters*, 4(3):2407–2414, 2019.

[3] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation, 2019.

[4] Valentin Thomas, Emmanuel Bengio, William Fedus, Jules Pondard, Philippe Beaudoin, Hugo Larochelle, Joelle Pineau, Doina Precup, and Yoshua Bengio. Disentangling the independently controllable factors of variation by interacting with the world. *arXiv preprint arXiv:1802.09484*, 2018.

[5] Martina Lippi, Petra Poklukar, Michael C Welle, Anastasiia Varava, Hang Yin, Alessandro Marino, and Danica Kragic. Latent space roadmap for visual action planning of deformable and rigid object manipulation. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2020.

[6] A Srinivas, A Jabri, P Abbeel, S Levine, and C Finn. Universal planning networks. In *Int. Conf. on Machine Learning*, 2018.

[7] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *IEEE Int. Conf. Robot. Autom.*

[8] John K Haas. A history of the unity game engine. 2014.

[9] 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. 2016.

[10] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 1735–1742, 2006.

[11] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017.

[12] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9:2579–2605, 2008.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[14] Davide Chicco. Siamese neural networks: An overview. *Artificial Neural Networks*, pages 73–94, 2020.

5 Appendix

5.1 Loss Functions and Architectures

In this section we describe the architectural details, the loss functions, as well as the hyperparameter for all models used. The input dimension for all models is a 256x256x3 image and the latent space for all models is 12-dimensional.

β-VAE: The standard β-VAE loss function is adopted:

$$\mathcal{L}_{vae}(x) = E_{z\sim q(z|x)}[\log p(x|z)] + \beta \cdot D_{KL}\left(q(z|x)||p(z)\right) \tag{1}$$

where the first part is the reconstruction loss and the second part the KL-divergence loss. As we feed in triplets, we combine the two individual loss functions $\mathcal{L}_{vae-\beta}(x_i)$ and $\mathcal{L}_{vae-\beta}(x_j)$ to a single loss as follows:

$$\mathcal{L}_{vae}(x_i, x_j) = \frac{1}{2}(\mathcal{L}_{vae}(x_i) + \mathcal{L}_{vae}(x_j)) \tag{2}$$

The implementation of encoder and decoder of the β-VAE [9] is realized with a two layer deep ResNet architecture [13] with a depth of two per block. We train the model for 500 epochs with a scheduling for beta from 0 to 1.5 and a batch size of 64.

VAE-A: As in [5], an additional contrastive loss, called action loss, to the standard β-VAE loss is introduced in VAE-A, thus resulting in the following overall loss:

$$\mathcal{L}_{vae-a}(x_i, x_j, a) = \frac{1}{2}(\mathcal{L}_{vae}(x_i) + \mathcal{L}_{vae}(x_j)) + \gamma \mathcal{L}_{action}(x_i, x_j, a) \tag{3}$$

with $\mathcal{L}_{action}(x_i, x_j)$ defined as

$$\mathcal{L}_{action}(x_i, x_j, a) = \begin{cases} \max(0, d_m - ||z_i - z_j||_1) & \text{if } a = 1 \\ ||z_i - z_j||_1 & \text{if } a = 0 \end{cases} \tag{4}$$

where a indicates if an action took place between I_1 and I_2 ($a = 1$) or not ($a = 0$), while d_m is a hyperparameter denoting the minimum distance that is encouraged for action pairs.

The same ResNet architecture of the β-VAE is adopted for the implementation of the models VAE-A. Note that in this case also the parameter γ needs to be tuned. For the training, we start with the scheduling for $\gamma = 40$ after 50 epoch warmup and increase it up to $\gamma = 100$. The minimum distance d_m is determined (as in [5]) by measuring the average action pair distance obtained from the β-VAE. In detail, the d_m was set to 13.3, 12.9, 14.5 and 14.7 when using datasets $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$ and $\mathcal{D}_m$, respectively.

Siamese:

This model uses the contrastive pairwise loss function [10] which is similar to eq. 4:

$$\mathcal{L}_{margin}(x_i, x_j, a) = \frac{1}{2} \begin{cases} \max(0, m - \|(z_i - z_j\|)^2 & \text{if } a = 1 \\ \|(z_i - z_j\|^2 & \text{if } a = 0 \end{cases} \tag{5}$$

where $m = 0.5$ is the margin parameter. This loss function requires pairs of examples that are either similar or dissimilar. The Siamese network [14] architecture is comprised of two identical encoder networks. Each encoder has the following latent encoding architecture:

$$\begin{aligned} x_1 &= MaxPool(x, 2 \times 2) \\ x_2 &= Conv(x_1, 4 \times 4, relu) \\ x_3 &= Conv(x_2, 4 \times 4, relu) \\ x_4 &= MaxPool(x_3, 7 \times 7) \\ z &= FC(x_4, 12, relu) \end{aligned}$$

5.2 Latent Space Roadmap

The Latent Space Roadmap (LSR), introduced in [5] for both rigid and deformable object manipulation, is a graph-based structure that allows to perform planning in the latent space. The LSR is built in three steps: *i)* a reference graph is constructed preserving the action pairs as connected nodes, *ii)*

the clustering is performed and a representative point for each cluster is chosen as its centroid point, *iii)* the edges of the nodes connected in the reference graph that belong to different clusters are used to build the edges of the LSR. In this work, we use the clustering method HDBSCAN [11] for step *ii)*. HDBSCAN is a hierarchical clustering method that tries to find clusters of different density by optimizing separations of clusters with a notion of persistence for each cluster. The main advantage of HDBSCAN over the ϵ-clustering method used in [5] is that it requires only the hyperparameter of the minimum samples per cluster to be set. For our evaluation, this parameter was set to 5.

For the evaluation of the paths, we determine the representative state of a cluster as the state that holds the majority in it. When analyzing the edges, we check if the move between the two representative states of the cluster is valid considering the stacking rules (detailed in Appendix 5.4). When analyzing the planning, we encode the start and goal observations with the model of interest and find the respective closest nodes in the LSR as start and goal nodes of the path. Given the start and goal nodes, all shortest paths are then extracted and their validity is checked to calculate the path scores.

5.3 Additional Experimental Results

In this section, we present the experimental analysis regarding datasets $\mathcal{D}_r$ and $\mathcal{D}_l$. Table 2 summarizes the respective results when the three different models are used. For dataset $\mathcal{D}_r$, we observe that all model representations result in a very high number of clusters, hinting at the increased difficulty of the viewpoint with respect to the frontal one. Lowest planning performance is obtained with β-VAE while similar performance of $\approx 48\%$ for the correctness of all paths is reached by VAE-A and Siamese models. The good performance despite the high number of clusters can be explained by the high homogeneity of the found clusters. This implies that the resulting LSR is composed of a high number of nodes, with really few samples in each, that allow to move in the latent space. This highlights the two ways in which solutions to the problem of planning in latent space can be improved: either by building a beneficial embedding (focus of this work) such that planning with it is easy or by building a roadmap that can perform well even with a suboptimal embedding (focus of [5]). Concerning the analysis of the results on dataset $\mathcal{D}_l$, it shows a significant lower number of clusters for all models with the VAE-A and siamese models again achiveing similar performance.

Models	Dataset $\mathcal{D}_r$						Dataset $\mathcal{D}_l$					
	Clust. num.	Clust. hom.	Edges num.	Edge corr.	Paths scores % all	Paths scores % any	Clust. num.	Clust. hom.	Edges num.	Edge corr.	Paths scores % all	Paths scores % any
β-VAE	740	0.96	**646**	0.95	5.95	6.4	41	0.91	44	0.7	10.1	10.1
VAE-A	**734**	**0.99**	687	0.98	**48.73**	49.2	**10**	**0.94**	**19**	**1.0**	**76.3**	76.3
Siamese	778	**0.99**	763	**0.99**	48.21	**49.6**	24	**0.94**	65	0.82	68.87	**77.5**

Table 2: Results for the four models (β-VAE, VAE-A, Siamese) on datasets $\mathcal{D}_r$ and $\mathcal{D}_l$. We report the number of clusters and their homogeneity, the number of edges and their correctness score, and the planning performance in terms of percentage when all found paths are correct and percentage if at least one path is correct. Best results in bold.

The t-SNE plots for $\mathcal{D}_r$ (top row) and $\mathcal{D}_l$ (bottom row) are finally shown in Fig. 3. We observe that that the β-VAE does not manage to cluster the same states together as efficiently as the other models. The VAE-A and Siamese are able to approximate the true number of 12 clusters much more effectively.

5.4 Box Stacking Setup and Underlying States

Box stacking setup: The box stacking task abides by the same rules presented in [5]. In detail, the boxes can only be placed in a predetermined 3×3 grid where each cell can only be occupied by one box at the time and the following rules need to be respected: *i)* a box can only be moved if there is no other box on top of it, *ii)* a box can only be placed on the ground or on top of another box (but never outside the grid), *iii)* no boxes can be added or completely removed from the grid (there are always 4 boxes in play). When generating the datasets, we introduce a planar position noise of $\approx 17\%$ to define the position of a box in a cell as well as we vary the lighting conditions, the color of the background wall and the color of the table for each observation. This guarantees that all the observations in the dataset differ from each other.

Underlying states: Given the box stacking rules, we can determine all the possible underlying states of the system. In particular, each state is given by a possible grid configuration specifying for

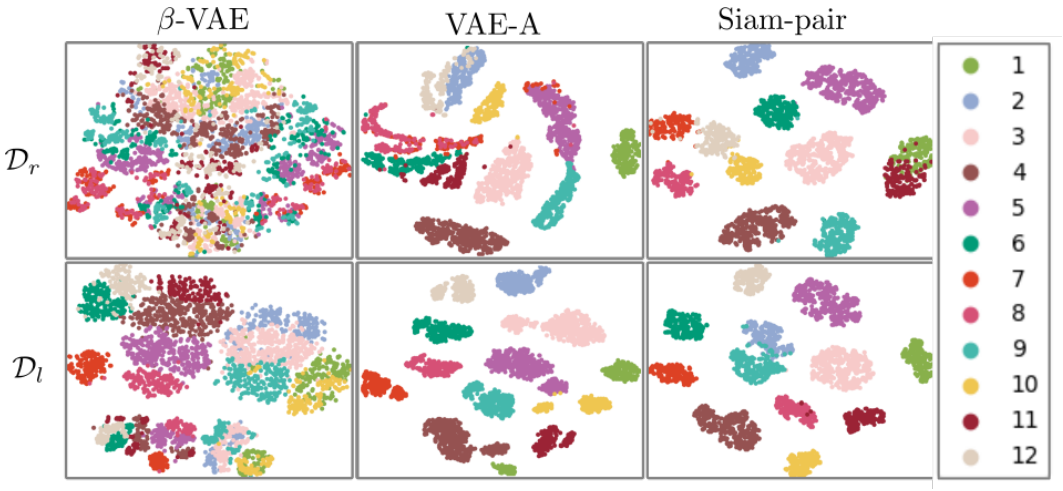

Figure 3: t-SNE plots for 2500 encoded observations from all three models using datasets $\mathcal{D}_r$ and $\mathcal{D}_l$.

every cell whether it is occupied by a box or not. Given the grid size and stacking rules, there are exactly 12 different box placements, i.e. grid configurations. Fig. 4 shows examples for these 12 distinct states.

Figure 4: Example observations of the 12 different possible states of the system. Note that the color of the boxes does not matter, i.e. boxes are interchangeable.