Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Final Report

Author:
Andy Wang

Supervisor:
Dr. Antoine Cully

Submitted in partial fulfillment of the requirements for the MSc degree in Artificial Intelligence of Imperial College London

September 2020

Abstract

The Quality-Diversity family consists of optimisation algorithms that create a large collection of solutions for a task at hand. Each solution attempts to solve the task objective in a different but effective way. The cultivation of a collection of well-performing solutions is particularly attractive to the learning of robot behaviours. A large diversity in the collection can allow the robot to overcome challenges that would render a single optimal solution ineffective. In attempting to develop a diverse set of solutions, we are interested in the robot behaviours they generate. A similarity in behaviour is quantified by the distance between the solutions' Behavioural Descriptors. The AURORA algorithm produces each descriptor automatically by applying dimensionality reduction techniques to the robot's sensory data collected while executing a solution. However, these generated descriptors do not accurately capture the behaviour when the data points are distorted by events occurring independently of the robot's actions. In this work, we introduce a Variational Autoencoder architecture that replaces the dimensionality reduction technique used in AURORA. We perform experiments using two different types of observations of the robot's environment as the sensory data. The results show that the AURORA algorithm is strongly susceptible to the stochastic occurrence of irrelevant elements in the observations. The application of our algorithm generates solutions that approximate the diversity produced by AURORA in a noise-free environment, and maintain this level of diversity at all evaluated environment noise levels.

Contents

Chapter 1

Introduction

This project is titled *Unsupervised Reinforcement Learning in Robotics* and focuses on adapting the AURORA learning algorithm [1] to environments in which an occurrence of irrelevant elements in the observations of a robot's behaviour can impair the effectiveness of the algorithm in its current form.

1.1 Motivation

The AURORA algorithm is a Quality-Diversity (QD) algorithm [2, 3] used to generate an archive [4, 5, 6, 7] of robot controllers. QD optimisation algorithms search for a large collection of solutions, rather than a single optimal solution as commonly performed in gradient-based optimisation for instance. In every iteration of a QD algorithm, a number of solutions are selected as parent solutions from which new solutions are generated through mutation and cross-over operations. In choosing which solutions perform well and should be added to the archive, we can consider their quality and their diversity. While the quality can be rather task-specific, the goal of generating a diverse set of behaviours is largely task-agnostic. To determine the diversity of solutions, we want to know how similar they are to other discovered solutions. We achieve this by calculating the distance of the Behavioural Descriptor (BD) of a given solution to the descriptors of other known solutions [7]. These descriptors should capture information on the behaviour generated by the robot's execution of a given solution. For instance, in the case of a gripper mounted on a planar robot arm with N degrees of freedom, a solution might be located in the N-dimensional parameter space, dictating an angle of rotation at each joint of the arm. An appropriate BD however, might be the coordinates of the location of the gripper and therefore situated in the two-dimensional Euclidean space.

A given pair or set of solutions might thus be quite dissimilar in the solution space, but yield very similar behaviours, for instance due to joint redundancies in a robot arm. The appropriate definition of the Behavioural Descriptor is therefore vitally important to the generation of a high-performing and diverse set of solutions [3]. In simple environments as in the robot arm example given above, the manual definition of a descriptor is possible and indeed a common approach. However, the definition

quickly grows more challenging and less feasible in the complexity of the environment.

The AURORA algorithm performs an automatic definition of the Behavioural Descriptor. It uses dimensionality reduction techniques such as deep Autoencoders to generate a low-dimensional descriptor from a high-dimensional environment observation. To train the Autoencoder, the AURORA algorithm uses only the observations that are generated by the solutions found by the algorithm. An illustration of the algorithm is presented in Fig. 1.1.

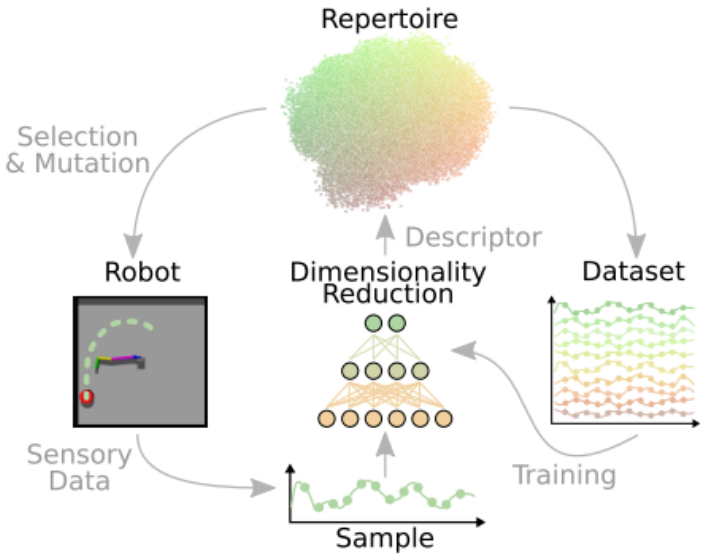


Figure 1.1: AURORA algorithm. Illustration taken from [1].

It is therefore critical for the algorithm to be able to distinguish which elements of a given environment observation are consequences of the robot's actions and should be captured in the Behavioural Descriptor. In this context, this project explores methods with which such source discrimination can be performed in an automatic and unsupervised way as part of the AURORA framework. We propose the use of a Variational Autoencoder to simultaneously perform the source discrimination and behaviour description.

1.2 Contribution

The goal of developing a large and diverse, yet high-performing collection of solutions makes QD a promising approach for learning robot behaviours. A robot equipped with such a behavioural repertoire [4] becomes more adept at reacting to changes in circumstances such as mechanical damages [8] which would render a single "optimal" solution unusable. However, prior knowledge of the robot design, the task or the environment is often still required to define the BD and other aspects of the QD algorithm. Further changes in the environment or task would necessitate

reconfiguration and reprogramming by engineers. This severely limits the applicability of these algorithms and their deployability to all but few potential use cases.

The elimination of this dependency on prior information is the defining feature of the AURORA algorithm. By design, the algorithm is robust to changing the environment. However, in its current form it is strongly susceptible to changes within a given environment when these occur independently of the robot's actions. This limits its applicability to all but few real world environments in which noise and stochasticity is the norm. At present, the benefits of the AURORA algorithm can thus hardly be realised. Our work forms a novel contribution to the QD literature. Yet more importantly, by enhancing the AURORA algorithm with a robustness to noise, we provide a further important stepping stone on its path to achieving true autonomy in smart robots.

Chapter 2

Background

We divide the relevant background in the literature into four sections. We first discuss the literature on Quality-Diversity, AURORA and similar algorithms in Chapter 2.1. Second, the background on Variational Autoencoders (VAEs) which we discuss in Chapter 2.2. In Chapter 2.3, we introduce some of the tools at our disposal to influence the shape of the latent space created by our VAE. In Chapter 2.4, we review some of the works in the Machine Learning literature that propose solutions to mitigate the adverse impact of label noise on the training of a Neural Network.

2.1 Quality-Diversity and AURORA

Optimisation algorithms are at the core of computational learning and have become the backbone of most Machine Learning techniques. These algorithms typically aim to produce a single high performing solution for a particular task at hand. However, there exist a multitude of advantages to generating and preserving a diverse set of solutions. For instance, the availability of a large number of different solutions allows us to overcome challenges arising due to differences between reality and the simulated environment in which a solution was determined to be optimal [9]. For instance, Cully et al. demonstrate the practicality of maintaining a set of diverse solutions in adapting a robot system to unexpected mechanical failures [8]. To this end, various approaches in the Evolutionary Computation literature have explored techniques to create a diverse set of high performing solutions which have recently been summarised as the family of Quality-Diversity algorithms [2, 3].

In [7], Lehman et al. introduce the Novelty Search algorithm, a non performance-based search for solutions that differ from solutions found previously. Computationally, this difference is computed as a novelty score. To calculate this score, solutions are first assigned a Behavioural Descriptor. The score is then computed as the average distance to the k nearest solutions in a novelty archive. If the novelty score of a solution exceeds a pre-defined threshold, it is added to this archive. In [7], the algorithm is shown to be successful in generating a diverse set of solutions. However, the authors note that this diversity is partly achieved through the creation of almost arbitrary and therefore non-functional solutions. In their follow-up work, they address

this issue with the Novelty Search with Local Competition (NSLC) algorithm [6]. In NSLC, solutions are additionally given a local quality score according to a specified quality criterion. The algorithm then applies simultaneous optimisation techniques to generate solutions that are both different from other existing solutions and of higher quality than similar existing solutions. In [6], the authors successfully apply this algorithm to create a diverse population of virtual creatures of different physical morphologies with the ability to walk.

In [4], Cully et al. introduce Behavioural Repertoire Evolution, a variant of NSLC that instead focuses on generating a diverse set of behaviours for a single robot. Additionally, Behavioural Repertoire Evolution now considers the novelty archive itself as the collection of solutions. In NSLC, the archive is used only to evaluate the novelty of newly generated solutions. Various further work in the QD literature have introduced different ways of storing and evolving solutions. Notably, MAP-Elites [10], a grid-based storage system, have emerged as the most relevant alternative to the concept of an archive-based storage. As this project does not utilise MAP-Elites, we do not discuss them further but refer to a detailed comparison of the two storage systems in [5].

An important contributing factor to the successful application of these algorithms lies in the appropriate definition of the Behavioural Descriptor of generated solutions. Several works in the literature have recently attempted the task of automating the behaviour description. However, in most cases prior knowledge of the task is integrated into the design of this automation.

Meyerson et al. [11] present a method that creates a Behavioural Descriptor which serves as a generic underlying descriptor for a large variety of tasks. This underlying descriptor is then refined further in an automatic fashion to produce final task-specific descriptors. While producing good results, this approach requires the injection of prior information in creating the underlying generic BD.

Cully et al. [12] propose a hierarchical behavioural repertoire which combines several archives in a hierarchical fashion to generate complex robot behaviour. The authors apply a convolutional deep Autoencoder to automatically create the BDs for the last hierarchical layer of the repertoire. In this work, the prior knowledge of the task - a robot arm writing digits - was used to select the MNIST dataset of handwritten digits to pre-train the Autoencoder. The network is then used to produce latent representations that are utilised as descriptors.

The AURORA [1] algorithm also applies dimensionality reduction in the observation space to automatically generate the BDs. However, unlike the works mentioned previously, no prior knowledge of the task is injected into the design of the algorithm. The dataset used to train the dimensionality reduction technique consists solely of the environment interactions generated by the robot's execution of solutions found by the algorithm. The training process is therefore fully integrated into the algorithm and executed online. In the paper, Principal Component Analysis and a deep Autoencoder are evaluated as the potential dimensionality reduction techniques.

Giuseppe et al. [13] propose a similar algorithm to AURORA. The approach in this work differs in its use of RGB image data as the environment observations, as opposed to ground truth internal state information in the AURORA paper.

2.2 Variational Autoencoder

A deep Autoencoder (AE) [14] consists of two Neural Networks, an Encoder and a Decoder network. The Encoder takes the input data x and produces a latent representation z. This latent representation is then given to the Decoder as input, from which it produces an output $\hat{x}$. The Autoencoder is trained end-to-end to minimise its reconstruction loss, the differences between the inputs and the outputs.

We are replacing the AE in the AURORA algorithm with a Variational Autoencoder [15]. A VAE is a deep generative model. Similar to the Autoencoder, it consists of two Neural Networks, an Encoder and a Decoder. However, in a VAE, both networks are probabilistic and produce the mean and variance of a distribution. Most commonly, and in the case of our application, the two distributions are modelled as normal distributions. The Encoder network models the parameters of the distribution of the latent representation conditioned on the input data. If we parameterise the Encoder network by φ, we can sample the latent variable z from the Encoder network which we denote as $q_\varphi(z|x)$. The Decoder network models the parameters of the distribution of the output $\hat{x}$ conditioned on the latent representations. If we parameterise the Decoder network by θ, we can denote this network as $p_\theta(\hat{x}|z)$.

Due to the intractibility of the direct calculation of the data log-likelihood $p_\theta(x)$, we instead maximise the evidence lower bound (ELBO) [16] to train the VAE. In calculating this lower bound, we use both probabilistic networks. The ELBO is derived as:

$$ELBO(x) = E_{q_\varphi(z|x)}[\log p_\theta(x|z) - \beta * \mathrm{KL}[q_\varphi(z|x)||p_\theta(z)]. \tag{2.1}$$

To achieve dimensionality reduction, a bottleneck is introduced by mapping the input into a lower dimensional latent space. This results in a loss of information. The reconstruction log-likelihood $\log p_\theta(x|z)$ measures the reconstruction performance as the log-likelihood of the model producing the input x given its latent representation z. For a data sample x_i, $\log p_\theta(x_i|z_i)$ is given as,

$$\log p_\theta(x_i|z_i) = \log \mathcal{N}(\mu_i, \sigma_i^2) = -\frac{(x_i - \mu_i)^2}{2\sigma_i^2} - \frac{1}{2}\log 2\pi\sigma_i^2, \tag{2.2}$$

where μ_i and σ_i^2 are the mean and variance produced by the Decoder network. To avoid instability issues for small values of the variance, we interpret the Decoder output as the logarithm of the variance instead.
More commonly, the Decoder variance term is ignored altogether. The resulting reconstruction log-likelihood we seek to maximise is then given as,

$$\log p_\theta(x_i|z_i) = -(x_i - \mu_i)^2. \tag{2.3}$$

In the second term of the ELBO, $q_\varphi(z|x)$ is the distribution parameterised by the Encoder network, from which the latent variable z can be sampled. In practice, we utilise the reparameterisation trick to perform the sampling [15]. This allows us to estimate the gradient using stochastic gradient descent. $p_\theta(z)$ is the prior distribution which we model to be a standard normal distribution. $KL[q_\varphi(z|x)||p_\theta(z)]$ is the Kullback–Leibler divergence between the two probability distributions, given as

$$\text{KL}[q_\varphi(z_i|x_i)||p_\theta(z_i)] = \frac{1}{2}(1 + \log(\sigma_i^2) - \mu_i^2 + \sigma_i^2), \tag{2.4}$$

where μ_i and σ_i^2 are the mean and variance produced by the Encoder network.
Finally, the scalar coefficient β [17] balances the reconstruction and the KL divergence terms in the ELBO.

Traditionally, a VAE is therefore also trained to reconstruct the input data; we desire the output $\hat{x}$ to be as similar as possible to x. Conceptually, the mere use of a VAE instead of an AE would therefore not change the AURORA algorithm substantially. To achieve the desired noise discrimination, we will instead be using the robot's actions as the input data, with the goal to construct the associated environment observations. This design allows for any infrequently occurring random observation elements to effectively be excluded from the representation as they are uncorrelated with the robot's actions. Simultaneously, we preserve the elements that reflect environment changes that are correlated with the robot's actions. As a result, accurate BDs can be generated despite the presence of noise in a stochastic environment.

A similar application of a VAE was presented by Zambelli et al. [18], who use a multi-modal VAE trained with measurements from different robot sensor modalities from two subsequent time-steps. This allows the model to learn to reconstruct missing sensor modalities at any given time-step, and to predict sensor readings of the next time-step given the current time-step's data.

The choice of a VAE over a standard Autoencoder additionally offers the advantage of its probabilistic framework. We discuss this in further detail in Chapter 3. Furthermore, the use of a VAE allows us to influence the shape of the produced latent space.

2.3 Shaping the Latent Space

By increasing the value of β in Eq. 2.1 we can influence the Encoder's arrangement of latent representations in the low dimensional space. Typically, we can observe two major consequences of using a standard Gaussian distribution as our prior $p_\theta(z)$.
First, a disentanglement of the latent features [17] due to the prior distribution's diagonal covariance matrix. When a learned representation is disentangled, changes across a single generative factor of the data trigger changes in the representation

along a single latent dimension. Bengio et al. find that learning such a representation of the data can be desirable in many tasks [19]. In the context of our application, a disentanglement allows us to understand which underlying generative factors of the data the model learns to include in the Behavioural Descriptor. This enhances the AURORA algorithm with some additional explainability and interpretability, two attributes that have gained prominence in the Explainable AI literature in recent years [20, 21, 22]. Additionally, a disentanglement can result in an increased isometry of the VAE's encoding of observations. This can render the novelty evaluation more effective as it is distance-based in the descriptor space.
Second, the zero mean and the unity of the standard Gaussian's covariance matrix imposes pressure on the Encoder to target a specific size and shape in its population of the latent space.

Applying this pressure is especially important in our proposed architecture. In the original AURORA algorithm, observations serve as the input to the dimensionality reduction algorithm. Consequently, a given observed behaviour will be assigned a certain descriptor regardless of the specifics of the robot's actions that trigger this observed behaviour. However, in our proposed architecture the robot's actions serve as the inputs to the Encoder. It is therefore possible for two robot controllers to be mapped to different descriptors, even if the produced observation constructions match. This reduces the accuracy of our BDs. By influencing the shape and the size of the occupied latent space, we can attempt to restrict the freedom the Encoder has to generate different descriptors for the same behaviour. In addition to an evaluation of the effects of the KL divergence criterion, we also explore the use of a Stochastic Neighbour Embedding (SNE) [23] and t-Distributed Stochastic Neighbour Embedding (t-SNE) [24] criterion to achieve this objective.

SNE is a dimensionality reduction technique that aims to preserve the neighbourhoods of data samples x_i situated in a high-dimensional observation space when creating their low-dimensional representations z_i. Euclidean distances between pairs of data points are used in both spaces to calculate conditional probabilities of a point i picking a point j as its neighbour. In SNE, these probabilities $p_{j|i}$ and $q_{j|i}$ are calculated under a Gaussian distribution centered at point x_i and z_i in the high and low-dimensional space respectively. Eq. 2.5 gives the calculation for the conditional probability $p_{j|i}$:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)} \tag{2.5}$$

These conditional probabilities represent pair-wise similarities. They are larger for nearby data points, and smaller for distant data point pairs. Since we are only interested in pairs, we define $p_{i|i}$ to equal 0.
As the density of data points in the high-dimensional space is likely non-uniform, using the same variance σ^2 for all data points would prove suboptimal. In dense regions, a smaller variance is usually more appropriate. The variance σ_i^2 of each of these Gaussian distributions is therefore determined by the density of the data

around the distribution mean. The authors use binary search for each data point to find the value of the variance that produces a probability distribution P_i over all other data points x_j, such that the perplexity of this induced probability distribution satisfies

$$\text{Perp}(P_i) = 2^{\text{H}(P_i)},$$

where the value of $\text{Perp}(P_i)$ is chosen by the user and $\text{H}(P_i)$ denotes the Shannon entropy of P_i measured in bits,

$$\text{H}(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

We model the low-dimensional similarities $q_{j|i}$ in the same way but with a constant $\sigma = \frac{1}{\sqrt{2}}$. The resulting conditional probability term is presented in Eq. 2.6.

$$q_{j|i} = \frac{\exp(-||z_i - z_j||^2)}{\sum_{k \neq i} \exp(-||z_i - z_k||^2)} \tag{2.6}$$

Similarly, we set $q_{i|i}$ equal to 0. The SNE algorithm then aims to find representations z_i that minimise the mismatch between these two probability distributions P_i and Q_i in order to preserve the pair-wise similarities present in the high-dimensional space, when constructing the low-dimensional space. This is achieved by minimising a KL divergence in the cost function C, as shown in Eq. 2.7.

$$C = \sum_i \text{KL}(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{2.7}$$

In Eq. 2.7, the cost of placing the representations of similar high-dimensional data points far apart is very large. This makes the SNE cost function an attractive optimisation criterion to achieve a larger accuracy in the BDs generated by our proposed architecture.

In t-SNE, Van der Maaten and Hinton [24] introduce two changes to the SNE algorithm. First, they propose to symmetrise the pair-wise similarities by constructing joint probability distributions P and Q, such that $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$ in the high and low-dimensional space respectively. This provides a better robustness to outliers in the data. A given outlier x_i, will be very distant from all other high-dimensional data points and is therefore assigned a low conditional probability $p_{j|i}$ for all j. Consequently, the chosen location of its low-dimensional representation z_i will have little impact on the cost function. In t-SNE, the joint probability p_{ij} is defined as

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}, \tag{2.8}$$

where n is the number of samples in the dataset. This ensures, that every data point's contribution to the cost function has a lower bound, as it follows from Eq. 2.8 that,

$$\sum_j p_{ij} > \frac{1}{2n}.$$

The conditional probability $q_{j|i}$ as defined in the SNE algorithm in Eq. 2.6 is already symmetrised. We define,

$$q_{ij} = \frac{\exp(-||z_i - z_j||^2)}{\sum_{k \neq i} \exp(-||z_i - z_k||^2)}. \tag{2.9}$$

However, the use of a Gaussian distribution in the latent space often leads to a behaviour that the authors refer to as the crowding problem. The area or volume available to place samples around a certain point in the low-dimensional space is more restricted than in the high-dimensional space. Given data points placed approximately uniformly around a point x_i in the high-dimensional space, we encounter a crowding problem in the low-dimensional space when attempting to place their representations. Consequently, latent representations can overfill the center of the space and prevent gaps from forming between natural clusters. To alleviate this problem, the authors propose the use of a probability distribution in the low-dimensional space with heavier tails than a Gaussian distribution. This allows for larger distances between the latent representations of a given pair of data points. In t-SNE, the Student t-distribution with one degree of freedom is chosen to model the low-dimensional similarities. The probability q_{ij} is defined as,

$$q_{ij} = \frac{(1 + ||z_i - z_j||^2)^{-1}}{\sum_{k \neq i} (1 + ||z_i - z_k||^2)^{-1}}. \tag{2.10}$$

As before, we set p_{ii} and q_{ii} equal to 0. The cost function C, now consists of the KL-divergence between the two joint probability distributions P and Q,

$$C = \mathrm{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{2.11}$$

We will implement the SNE and t-SNE cost function as additional components in our VAE loss. The overall loss function is then given by the sum of a construction loss term, a KL divergence term weighted by β and the SNE or t-SNE cost function which we weight with a fixed parameter γ. The form of the overall optimisation objective to be minimised can therefore be written as,

$$L_{\text{VAE}} = C_{\text{construction}} + \beta * KL + \gamma * S, \tag{2.12}$$

where $C_{construction}$ is the negative log-likelihood, KL is the KL divergence criterion and S is the SNE or t-SNE cost function.

Graving and Couzin successfully utilise such a VAE architecture to preserve the local structure of the high-dimensional space in a reconstruction task [25].

A potential alternative to the SNE and t-SNE criterion is presented in the Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) algorithm [26]. Our preference for the SNE-based criterions is driven by the successful validation of their application in a VAE architecture in [25] and the empirical

performance similarity of the UMAP and t-SNE algorithm reported by the UMAP authors [26].

2.4 Dealing with Noise

The implications of the presence of noise in datasets is explored to great extent in the Machine Learning literature. Typically, we distinguish two types of noise: feature noise and class or label noise [27]. Generally, noise can have an adverse impact on a Neural Network's (NN) accuracy, its training time and other performance measures. However, limited feature noise has been shown to have positive effects on a NN's generalisation error [28]. Consequently, the intentional addition of feature noise to a dataset is widely explored in the literature as a method of regularisation [29, 30, 31, 32, 33]. On the other hand, label noise, i.e. the assignment of incorrect labels to data instances, can have a more harmful impact than feature noise [27, 34]. Frenay et al. explain this disproportionality with the fact that for each data instance there are usually many features but only one label [35]. They argue that different features will each have a different importance to the learning process, whereas the label and therefore its correctness always carries a large significance.
Naturally, many works in the literature have explored the task of designing algorithms that equip the learning process with some robustness to label noise. We can broadly divide these into the following categories, with some representative works presented in each category.

Modelling the noise distribution. Bekker and Goldberger propose the definition of a probabilistic model for the transformation from true to noisy labels and a learning scheme based on the Expectation Maximisation (EM) algorithm [36].
The true labels are estimated in the E-step, with these estimates used in the M-step to train the Neural Network. In a follow-up work, Goldberger et al. propose a fully Neural Network based approach that incorporates this EM learning scheme by utilising a softmax layer in the network [37]. While this approach shows good results, its design is restricted to the application to classification tasks and therefore not applicable to our use case.

Selection of clean instances. Han et al. propose the use of two NNs in a paradigm called "Co-teaching" [38]. It is based on the observation made by Arpit et al. [39] that Neural Networks will "memorise" easy instances first, and adapt to more difficult instances only as training epochs become larger. Given sufficient training time and model capacity, instances with noisy labels eventually still will be memorised [39], but, importantly, only later in the training process.
The authors exploit this idea by designating the instances with the smallest loss as the "useful" or clean samples. The instances with the largest loss are removed gradually over time, before the network can start memorising any incorrect labels. While co-teaching utilises two networks, this selection mechanism can be applied in a single NN. We exploit the memorisation effect in the design of our benchmarks in

Chapter 5.

Adjusting the loss incurred. Thulasidasan et al. design a modified cross-entropy loss function that allows the network to abstain from incurring a classification loss on individual samples at the cost of an abstention penalty [40]. Patrini et al. propose a loss correction approach, that utilises an estimated noise label transition matrix to weight the loss for samples likely to have a noisy label [41]. In both cases, we could exploit the memorisation effect again and use the reconstruction loss to construct a likelihood of a given sample containing noisy elements to identify which samples to abstain from, or weight respectively. However, this would likely prove destabilising to the network training, as the reconstruction loss would be used to weight itself.

Choice of loss function. Ghosh et al. derive some conditions on the robustness of loss functions in the presence of label noise [42]. In particular, the authors show that the Mean Absolute Error satisfies these conditions. In general, the L2 or Mean Squared Error criterion most commonly used in regression tasks is heavily affected by outliers due to its squared term and therefore not suited for learning on noisy datasets. We will compare the performance of an L2 criterion against Smooth L1, Huber with $\delta = 0.5$ and L1 loss configurations.

Batch size. Rolnick et al. find that the effective batch size decreases in the proportion of noisy labels [43] and that a larger batch size provides some robustness to label noise. We investigate the impact of different batch sizes in our experiments.

Chapter 3

The Proposed VAE Architecture

The AURORA algorithm performs a reconstruction of the observations to obtain the BDs. The inputs and outputs of the Autoencoder are illustrated in Fig. 3.1. To achieve the desired robustness to noise we instead use the robot's actions as the input data with the goal to construct the associated environment observations. We illustrate the proposed choice of the network's inputs and outputs in Fig. 3.2. By conditioning the latent descriptors on the robot's actions, we eliminate the correlation between the inputs and the stochastic elements in the environment observations. Simultaneously, we provide the network a learnable relationship between these inputs and the elements of the observations that are consequences of the robot's actions. This proposed architecture therefore bears more resemblance to a forward model [44] than a reconstruction model. Consequently, accurate constructions and BDs can be generated despite the presence of observation noise in a stochastic environment.

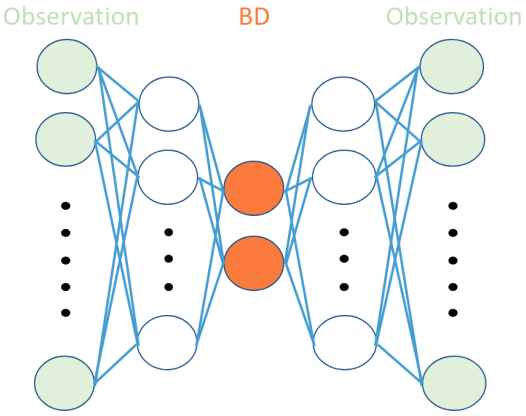


Figure 3.1: Network inputs and outputs in AURORA.

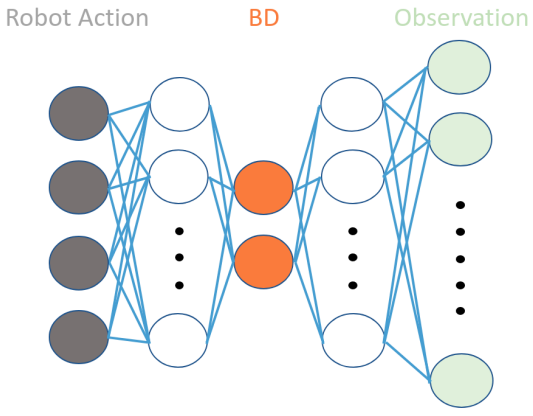


Figure 3.2: Network inputs and outputs in proposed architecture.

We evaluate various configurations of our VAE's architecture. These variants will be formed of different combinations of the features listed in Tab. 3.1. Fig. 3.3 illustrates how these various components are integrated into the VAE.

Feature	Possibilities
β (*Presence of KL Divergence Criterion*)	1 (*Yes*), 0 (*No*)
Sample the Descriptor during Inference?	Yes, No
Sample the Descriptor during Training?	Yes, No
Loss Function (*Presence of Decoder Variance*)	Log-Likelihood (*Yes*), Euclidean (*No*)
Distance Measure in Loss Function	Smooth L1 / Huber, L1, L2
Additional Loss Criterion Component	SNE, t-SNE, None

Table 3.1: Architecture features.

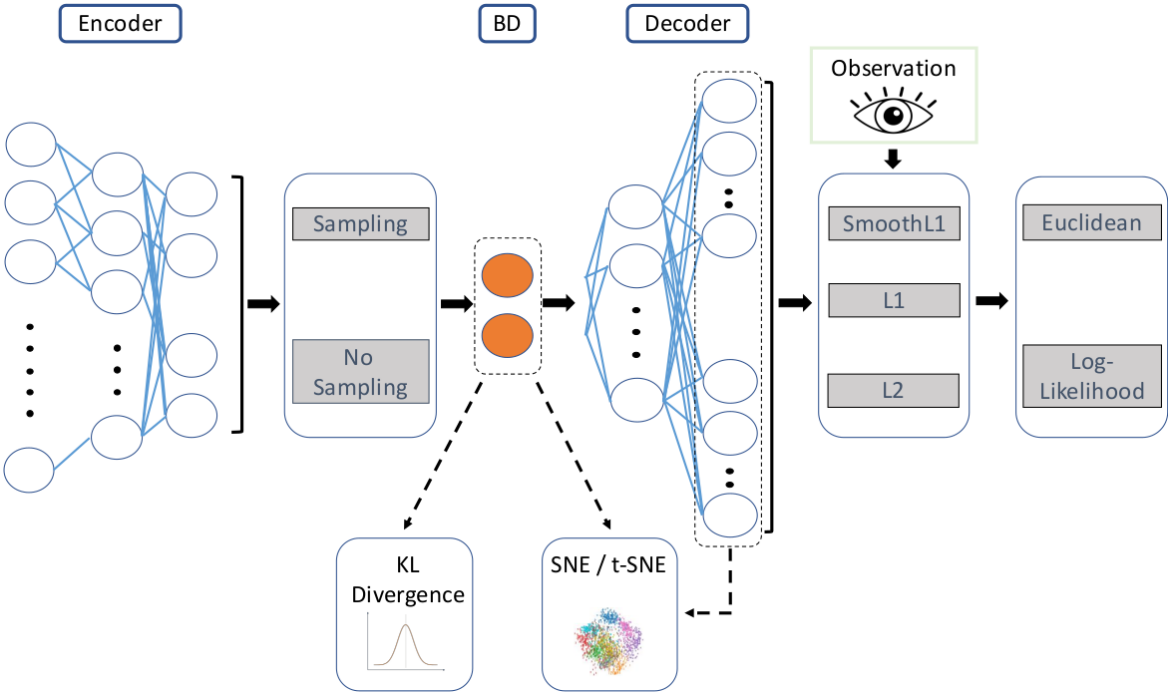


Figure 3.3: Illustration of the location of the considered architecture features in the VAE. [a]

[a] KL, SNE and Observation icons taken from: https://www.cleanpng.com/png-the-bell-curve-normal-distribution-grading-on-a-cu-791248/, https://stats.stackexchange.com/questions/340175/why-is-t-sne-not-used-as-a-dimensionality-reduction-technique-for-clustering-or, https://www.pngfuel.com/free-png/ampwc

Distance Metric. The L2 criterion is frequently the distance metric of choice in many ML applications. Its squared term provides a strong gradient signal but is also heavily affected by outliers and noise. We therefore evaluate the use of an L1 metric and a Smooth L1 or Huber loss criterion.

The Huber loss calculates a squared term for an error a of magnitude δ or less and a linear term elsewhere. We give its piece-wise definition in Eq. 3.1. The value of δ is chosen by the user. The Smooth L1 loss is a Huber loss with a chosen error threshold of 1. Both criterions combine the L1 and L2 metrics in an attempt exploit the strengths while mitigating the weaknesses of both metrics.

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{for } |a| > \delta. \end{cases} \tag{3.1}$$

Beta. The value of β controls the weight on the KL divergence criterion, relative to the other components in the VAE's loss function. A larger β increases the pressure to produce latent descriptors that form an approximately standard Gaussian distribution in the latent space. The potential consequences are:

- A decrease in the construction performance due to the additional constraint on the optimisation problem. This can lead to less accurate descriptors.
- An isometry of the VAE's encoding. This can render the novelty evaluation more effective as it is distance-based in the descriptor space.
- A disentanglement of the latent representation. This can increase the interpretability of the learned Behavioural Descriptors.
- A reduction in the size of the occupied latent space. This can lead to more similar behaviours collapsing towards the same BD and therefore more accurate descriptors.

SNE / t-SNE. The introduction of a SNE or t-SNE criterion can prove beneficial if they are successful in producing an arrangement of descriptors in the latent space that is more reflective of the actual observed behaviours. However, similar to the addition of the KL divergence term, this imposes an additional constraint on the optimisation problem and therefore likely leads to a decrease in the construction accuracy. In order to achieve the desired robustness to noise, we cannot utilise the actual environment observations in the SNE and t-SNE calculations. Instead, we must use the construction output of the VAE. We therefore require a strong construction accuracy for the SNE and t-SNE criterions to function as expected.
We expect to see significant differences in the results obtained from the SNE and t-SNE implementations. The pressure to produce similar representations for similar observations is larger with a SNE criterion than with a t-SNE criterion, due to the crowding effect. All else equal, we therefore expect the SNE variants to generate more accurate BDs.

Sampling the descriptor. We apply the reparameterisation trick [15] to sample a latent representation from the Encoder network. At inference time, sampling to obtain the BD can distort the true similarity between any given solutions. For instance, when the same solution is evaluated repeatedly, the network can produce different BDs in each instance when the Encoder variance is non-zero. This can lead to a re-inclusion of the same or very similar solutions into the archive. While this seems damaging to our goal of achieving a large diversity in our solutions, multiple observations of the repeated execution of the same solution can prove useful to the network's training when the environment is stochastic.
Sampling during training allows the Encoder to utilise its variance term to capture some uncertainty about the produced representations. We will evaluate the impact

on the network's construction accuracy. On the other hand, the presence of a variance term can reduce the pressure to increase the accuracy in the Encoder's predictions of the mean. This can have a negative impact on the accuracy with which the generated BDs capture the observed behaviours.

Loss function. The label noise in our dataset forces the VAE to shift its output closer towards an average of all noisy observations in order to decrease the construction loss incurred on these samples. While we expect this to also occur in the network trained with the full Log-Likelihood criterion, the Decoder variance term in the VAE's construction loss allows the network to respond to the stochasticity by increasing this variance. This can alleviate some of the pressure on the construction output to be shifted to the mean of all noisy observations. The use of a Log-Likelihood loss can therefore lead to an observation construction that is more similar to the actual observation than when the network is trained with the Euclidean distance loss in Eq. 2.3.

3.1 Naming Convention

In all subsequent discussions and illustrations, we refer to the different variants by the composition of the architecture features that define them. The features just introduced each take on a value which we capture as a Component Tag. We list the features, values and their tags in Tab. 3.2.

Feature	Feature Value	Component Tag
Distance Metric	L2	L2
	L1	L1
	Smooth L1 / Huber	L0
KL Divergence	Added ($\beta = 1$)	KL
	Not added ($\beta = 0$)	/
SNE / t-SNE	SNE	SNE
	t-SNE	TSNE
	Neither	/
Sampling in Encoder	No Sampling during Training	NST (**N**o**S**ample**T**rain)
	No Sampling during Inference	NS (**N**o**S**ample)
	Sampling during Training & Inference	/
Construction Loss	Log-Likelihood	LOG
	Euclidean	EUC

Table 3.2: Architecture features, their possible values and tags.

An architecture variant is then denoted by connecting its tags for each feature with a hyphen. Slashes are omitted. For instance, the architecture in which we apply a L1 distance metric, no KL, SNE or t-SNE criterion, sample during training only and use a Log-Likelihood construction loss will be denoted L1-NS-LOG. The AURORA implementation is denoted as L2-AURORA. The L2-NST-EUC variant is a standard Autoencoder and will also be denoted as L2-AE.

Chapter 4

Algorithm Process Flow

In this section we present the two main iterative processes in any AURORA-based algorithm.
We first introduce some terminology. A solution is a data structure containing a single genotype and phenotype and the associated environment observations. A genotype, which we refer to as the solution parameters, consists of a vector of real values between 0 and 1. The set of all possible vectors forms our solution space. The objective in our experiments is to generate controllers for a robot arm. We therefore apply a simple linear transformation to develop the genotype into a vector of the same size with values in the range of $[-\pi, \pi]$. This new vector is the solution's phenotype and its values control the angles at each joint of the arm.

In Fig. 4.1, we illustrate a single iteration of the principal loop of our algorithm. In line with the QD literature, we refer to each iteration as a "generation".
At the start of each generation, we select a predefined number of solutions from the *Archive*. The criteria with which we pick this set of solutions are defined in the *Selector*. In our experiments, each solution has an equal probability of being chosen. This group of solutions forms the set of *Parent solutions* which we split into pairs. Each pair of parents will undergo cross-over and mutation operations to produce two offsprings. Cross-over operations produce new solutions that bear similarities to both parents. For instance, when each genotype consists of a binary string this can be achieved by swapping segments of the parents' genotypes to obtain two new solutions. We utilise an SBX cross-over operator suitable for real-valued genotypes [45]. The mutation operator then introduces some stochastic changes to these new solutions. For instance, a binary string can be mutated by flipping one or several of its bits. We utilise a polynomial mutation operator [46].
If the archive is empty at the start of the generation, we instead generate the same number of solutions by randomly sampling in the solution space.

For each solution in the set of *Offspring solutions*, we generate a BD by feeding parts of the solution into the Autoencoder. In the case of our proposed architecture, these are the solution parameters. In the traditional AURORA algorithm, the inputs to the network are the solution's environment observations.
In the decision to accept a given new solution into the archive, we consider the

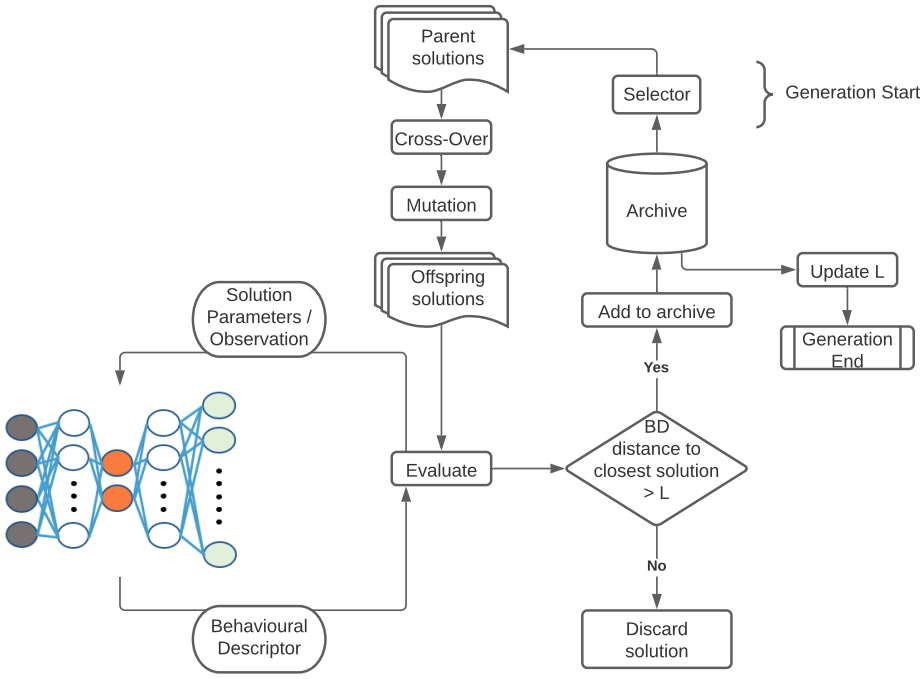


Figure 4.1: Execution steps in one generation of the algorithm.

Euclidean distance of its BD to the descriptor of the nearest solution contained in the archive. If this distance exceeds a minimum threshold, the solution is deemed sufficiently novel to be added to the archive. Otherwise, it is discarded.
We denote this threshold parameter as the distance L [5]. This parameter is initialised at the start of the algorithm, when its value is calculated from the BDs of the initial set of solutions contained in the archive. However, subsequent updates to the value of the parameter are made using only the difference between the size of the archive at the time of the update, and the target archive size which is predefined by the user. In particular, the density in the BD space is not considered in any of the updates. The parameter L can only increase in value if the target population size is exceeded at the end of the generation and vice versa. This mechanism is very important to our subsequent analysis when we evaluate the effects of a more compact Behavioural Descriptor space. Each generation ends in this update procedure.

The principal difference between a traditional QD algorithm and the algorithm shown in Fig. 4.1, is the use of an Autoencoder to generate the BDs in the latter.

Fig. 4.2 illustrates the execution steps in a single training iteration. These training iterations occur at predefined generation intervals. In our experiments, we set this frequency to be every 10 generations. At the start of each iteration, we extract all solutions from the *Archive*. We subsequently clear the archive, as we repopulate it at the end of the iteration. We build a dataset of features and labels from our set of *All solutions* which we then split into a training and validation set. Training stops when a maximum number of epochs has been reached, or when an early stopping criterion on the train and validation errors is satisfied. This early stopping criterion

is detailed in Appendix B.1.
Every solution's BD is then recalculated with the updated network weights. When readding the solutions to the archive, we apply the same novelty acceptance check as described previously. As the size of the archive can decrease substantially, we conclude the iteration with another update of the parameter L.

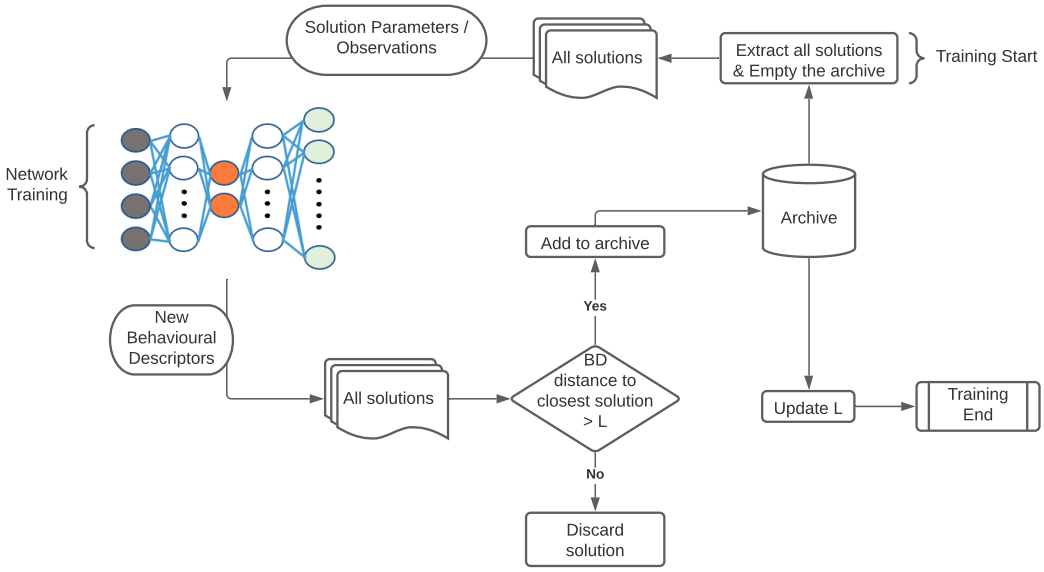


Figure 4.2: Execution steps in one training iteration of the algorithm.

Chapter 5

Benchmarks and Baselines

Given the novelty of the contribution made in this project, we do not have an alternative architecture from related work to compare against. Instead, we use the "memorisation effect" discussed in Chapter 2.4 to devise several benchmarks that adapt the AURORA implementation.

In benchmark EXCLUDE_TRAIN we designate the solutions with the largest reconstruction loss as the noisy samples and exclude them from the training set. In EXCLUDE_ARCHIVE these individuals are instead removed from the set of collected solutions altogether. In both benchmarks, we attempt to eliminate all noise from our data in order to mitigate its impact on the AURORA algorithm.

In benchmarks REGEN and EXTEND, we re-simulate the environment observations for the solutions with the largest reconstruction loss. In REGEN, these new observations replace the solutions' previous observations. In EXTEND, they are simply added as additional samples to the training set. In these benchmarks, we attempt to decrease the proportion of noisy samples in the dataset.

While these approaches might prove effective in environments with low noise levels, we expect their performances to deteriorate as the presence of noise increases its regularity. This is problematic, as noisy elements can be present in every single observation in a real life environment. Additionally, the chosen ratio of samples to exclude or re-simulate is an additional hyperparameter which in itself might not prove robust to different environment settings.

We consider two baselines. First, a set of solutions that is generated by randomly sampling in the solution space. Second, the AURORA implementation from the original paper without any adaptations. We expect the performance of the latter to deteriorate as the environment noise levels increase.

Chapter 6

Experiment 1: A Basic Simulation

This first experiment serves as a proof of concept to validate the core idea of our proposed architecture. We simulate an air hockey puck positioned in a square room. The puck starts in the same location at the beginning of each simulation. There are no obstacles in the room, nor do we model the physical appearance of the robot itself. Each solution has two parameters which dictate the angle and velocity at which the puck leaves its starting location. We will refer to this puck as the "actual" puck.

Environment noise is simulated by adding an additional puck moving along a random, but plausible, trajectory. We call this puck the random puck. We refer to the probability of a random puck appearing in the simulation as the "stochasticity" of the environment. The pucks can bounce off the walls of the room but not off each other.

The pucks' trajectories form our environment observations. A single solution can therefore be associated with multiple observations. A puck's trajectory consists of 50 x-y coordinate positions taken at discrete time-steps, resulting in a 100-dimensional observation space. The size of the Behavioural Descriptor is set to 2. As the solution space is also two-dimensional, we require no dimensionality reduction in this experiment. This leaves the environment stochasticity as the only challenge. A visualisation of an environment observation is presented in Fig. 6.1.

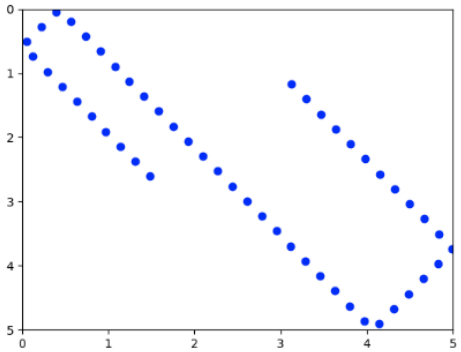


Figure 6.1: Visualisation of a puck trajectory created in our manual simulation.

6.1 Code Implementation

Our algorithm is implemented in C++ with a large focus on its parallelisability across a machine's available CPUs. The project supervisor provided us access to the source code of an application of Quality-Diversity optimisation to a robot arm task. Our application inherits all Quality-Diversity components shown in Fig. 1.1 from this program with the exception of the *Evaluate* module. The components are implemented in the Sferes2 framework [47]. We added additional functionality to record performance metrics and other statistics during the execution of the algoirithm. However, the bulk of our implementation efforts was focused on the creation of a simulation engine and the VAE pipeline to perform the automatic behaviour description.

To perform the simulation, we created a simple program that given an angle and distance produces the position of the puck at the next time-step while taking the room's dimensions into account. The Neural Network is implemented in the C++ Frontend[1] of the PyTorch library [48]. The main challenge in this implementation was posed by different solutions having a different number of associated trajectories. This required some additional attention in the implementation of the forward passes in the network to ensure the correctness and efficiency of the performed calculations. All visualisation tools are implemented using the matplotlib [49] and seaborn[2] libraries in Python. The application is containerised using the Singularity[3] framework. This was done to allow the use of the High Performance Computing services at Imperial College London[4].
The hyperparameters controlling the algorithm and Neural Network architecture are given in Appendix B.1 and B.2.

6.2 Behavioural Descriptor Assignment

AURORA uses the environment observations as inputs to the Autoencoder. In our implementation of the AURORA baseline, we therefore construct a solution's BD as the average of the latent representations of all its associated environment observations. In our proposed architecture, the parameters of any individual solution are encoded into a single BD, regardless of the number of pucks present in the environment.

6.3 Performance Measures

We measure the performance of an architecture by determining the diversity in the generated archive and by the VAE's construction performance. In all subsequent discussions, we present the results obtained in our performance metrics in the form

[1] `https://pytorch.org/cppdocs/frontend.html`
[2] `https://seaborn.pydata.org/`
[3] `https://sylabs.io/docs/`
[4] `http://www.imperial.ac.uk/admin-services/ict/self-service/research-support/rcs/`

of line plots. The lines indicate the median of the scores obtained over 5 repetitions. Shaded areas in the same colour extend from the line to the first and third quartiles of the data distribution. The data used for our diversity metrics is recorded after the last generation of the algorithm. All VAE metrics are calculated using data collected at the end of each generation starting at the 500^{th} generation.

6.3.1 The Diversity

In this set of experiments, the behaviour generated by the robot's execution of a solution is captured by the actual puck's trajectory. We therefore measure the achieved diversity performance by calculating various similarity metrics between the observed puck trajectories of solutions in the archive. In these calculations we will only consider the actual puck.

The Diversity Score

We implement the diversity score from the AURORA paper [1]. The score is calculated by discretising the environment into bins. We use a 20 x 20 discretisation. Each solution in the archive is assigned to the bin in which the puck ends its trajectory. For each bin, we find the number of distinct bins on the grid that any of its assigned trajectories cross. The number is then divided by the number of total bins in the grid. This yields a score in each bin ranging from 0 to 1. Fig 6.2 illustrates this calculation for a single trajectory.

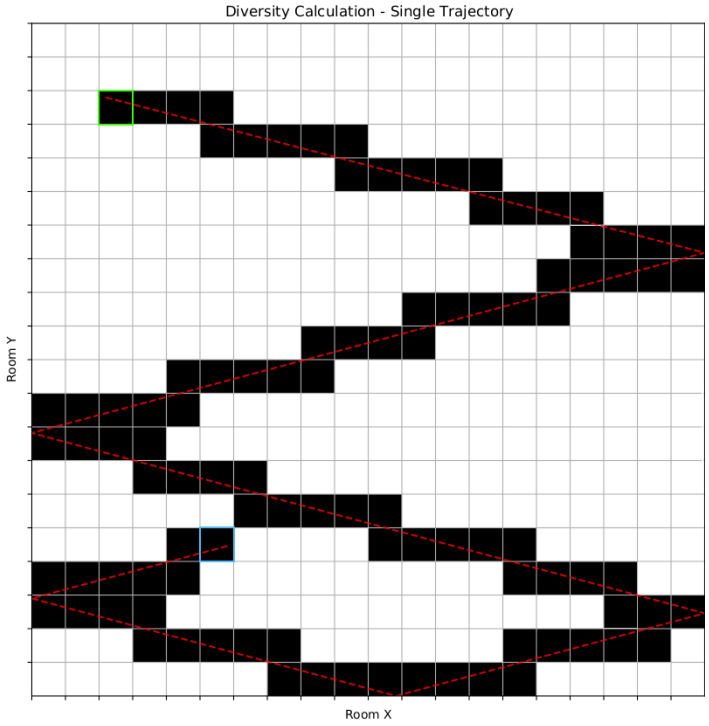
We obtain the total diversity score for the archive by summing the scores of each bin. An architecture demonstrates its robustness to noise by producing a stable diversity score across different levels of stochasticity in the environment. The discretisation required by this metric leads to some additional volatility in the obtained scores which we need to take into account when comparing the scores obtained by different variants. Note, that the diversity score favours longer trajectories.

Visualisation of all solutions

In this experiment, solutions consist of an angle and a velocity. We can translate the latter into the distance travelled by the puck over a fixed time period. This allows us to plot the entire archive in radial coordinates. An example is presented in Fig. 6.3. Each solution is visualised by a single line starting in the origin, which represents the starting location of the puck. The first component of each solution determines the angle of the line relative to the x-axis. The velocity component determines the length and colour of the line.

6.3.2 The VAE Performance

Loss Metrices



$$\text{Diversity Score} = \frac{\text{Number of Boxes Transversed}}{\text{Total Number of Boxes}} = \frac{198}{400} = 0.495$$

Figure 6.2: Diversity Score calculation for trajectory shown in red. The trajectory starts in the box framed in green and ends in the box framed in blue. The blue box is the bin this trajectory is assigned to. Black boxes count towards the diversity score of this bin.

To assess the VAE's performance, we log the Encoder and Decoder outputs and losses. Due to the label noise, the traditional aggregate (re)construction error is not a good indicator of the network's construction accuracy. We calculate an L2 error over the trajectories of the actual puck only. We denote this error as the "Actual L2".

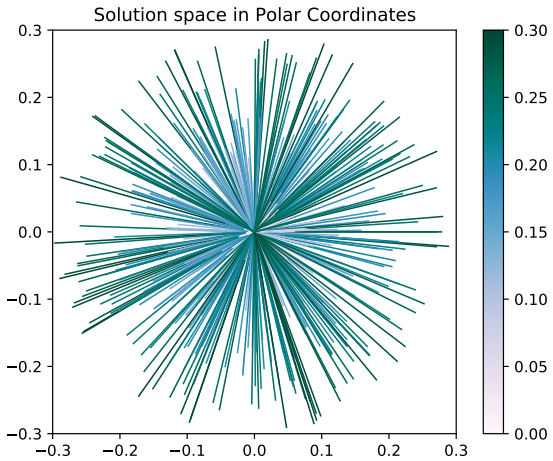


Figure 6.3: Solution archive plotted in Polar Coordinate System. The colour captures the length of each line.

6.4 Results and Discussion

Fig. 6.4 presents the diversity score of the L2-AURORA, L2-AE and L2-KL-EUC implementations. The performance of L2-AURORA deteriorates rapidly as the environment stochasticity increases. Since a solution's BD is calculated as the average of the latent representations of all its trajectory observations, random puck trajectories have a large influence on the perceived behaviour. The distance in the BD space therefore fails to accurately reflect the differences in the actual behaviour.
This is reflected in a drastically altered coverage of the solution space. Figs. 6.5 and 6.6 present the coverage for the AURORA algorithm at 0% and 100% environment stochasticity respectively. The presence of noisy trajectories leads to the assignment of dissimilar BDs to similar solutions. These solutions will therefore be included repeatedly into the archive. This results in overlapping lines and an increased sparsity in Fig. 6.6, despite an approximately equal number of solutions used to produce both graphs.

On the other hand, our proposed variants exhibit the desired robustness to noise by maintaining a stable diversity score in Fig. 6.4. Both variants produce solutions that cover the solution space uniformly even in the presence of stochasticity. We plot the VAE's solution space coverage at 0% and 100% stochasticity in Figs. 6.7 and 6.8.

However, Fig. 6.4 additionally shows large differences in the diversity scores achieved by each of our two variants. The L2-AE variant also produces a consistently lower Actual L2 error in Fig. 6.9. As both implementations use the same Euclidean construction loss criterion we can attribute the performance difference to the presence of the KL divergence loss term and the sampling of the BDs from the Encoder in the L2-KL-EUC variant.

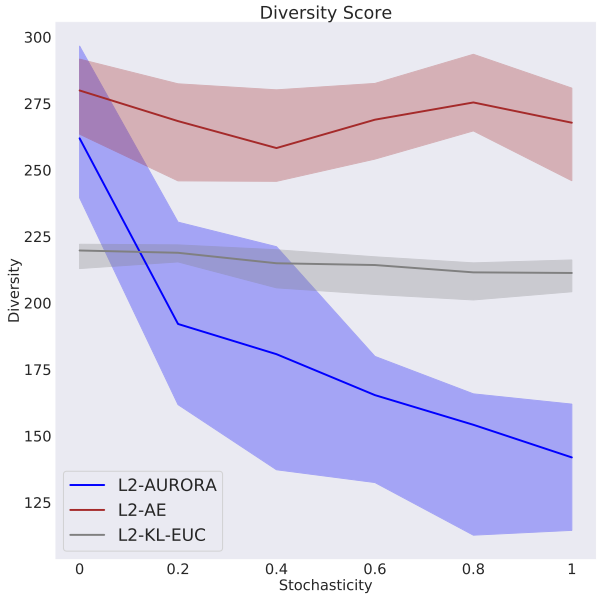


Figure 6.4: Lines indicating the median Diversity Score over 5 repetitions. Shaded areas extend from the median to the first and third quartiles of the data distribution.

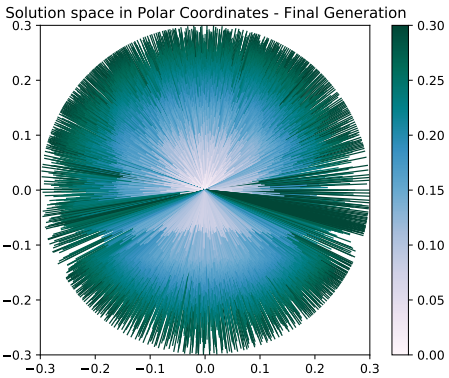


Figure 6.5: L2-AURORA solution archive at 0% stochasticity plotted in Polar Coordinate System. Colour indicates the radial coordinate.

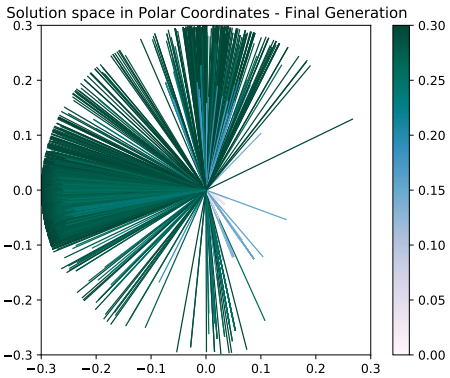


Figure 6.6: L2-AURORA solution archive at 100% stochasticity plotted in Polar Coordinate System. Colour indicates the radial coordinate.

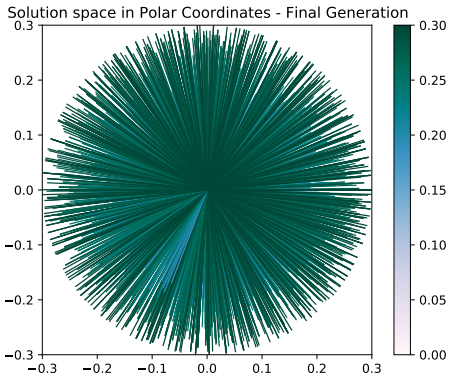


Figure 6.7: L2-KL-EUC solution archive at 0% stochasticity plotted in Polar Coordinate System.

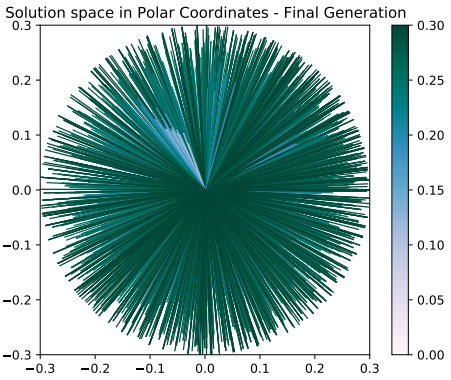


Figure 6.8: L2-KL-EUC solution archive at 100% stochasticity plotted in Polar Coordinate System.

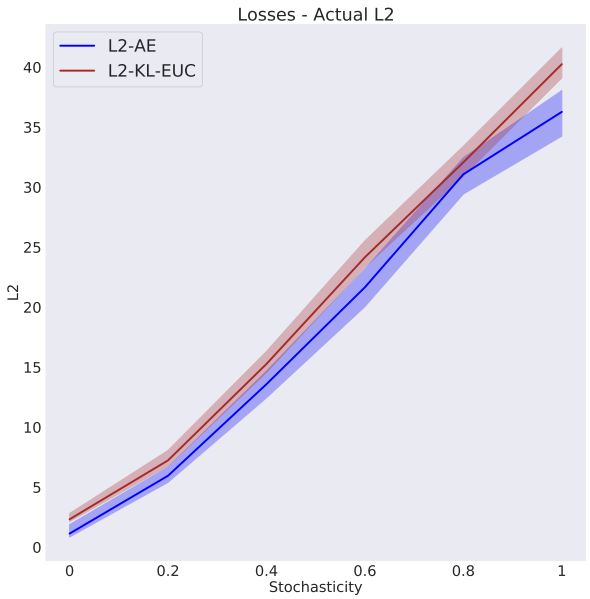


Figure 6.9: The Actual L2 Error.

6.5 Conclusion

In this first set of experiments, we validated the relevance of our proposed work. The AURORA implementation fails to maintain a stable diversity and solution space coverage as the stochasticity in the environment increases. By conditioning the BD on the parameters of the solutions rather than their associated observations, our proposed variants achieve the desired robustness to noise. The AE variant outperforms the VAE variant in our diversity and construction accuracy metrics. This suggests that the presence of the KL divergence criterion and the decision to sample the BD have a significant impact on performance.

Chapter 7

Experiment 2: Simulated Physics

In this experiment, we add the additional challenge of dimensionality reduction. We replace the manual simulation used in the previous experiment with the physics simulation engine Box2D[1]. A snapshot of the Graphical User Interface visualising this experiment's simulation is presented in Fig. 7.1. We use Box2D to model a planar robot arm mounted in the centre of a 5 x 5 square room. The robot arm has 4 degrees of freedom. Each solution has 4 parameters which dictate the angle to be reached in each arm joint by the end of the simulation. This results in a movement of the arm from its fully stretched starting position. Environment noise is created by simulating an additional puck travelling on a random trajectory. By moving the arm, the robot can hit the pucks which can bounce off the walls. In addition to a higher-dimensional solution space, the complexity of the task is increased in two further aspects. First, the arm can hit the puck several times. Second, collisions between the actual and random pucks are possible. Both will lead to more complex and therefore more difficult trajectories to construct.
We do not change the size of the BD or the observations. The latter are again formed of puck trajectories consisting of 50 sequential x-y puck position coordinates.

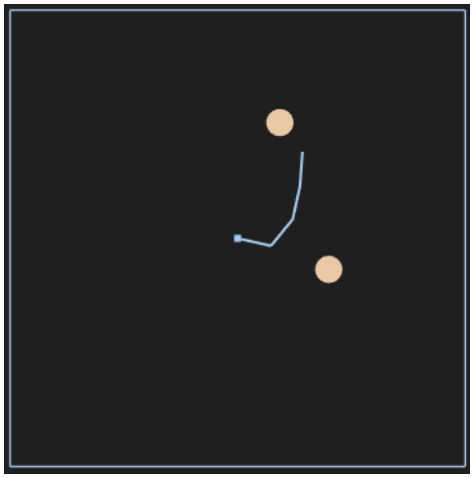


Figure 7.1: Box2D simulation with two pucks present in the environment.

[1] `https://box2d.org/documentation/index.html`

7.1 Code Implementation

In this set of experiments, we replace Experiment 1's manual simulation program with Box2D. We utilise the ROBOX2D library created by the project supervisor. It provides APIs to access the physics engine of Box2D and a Graphical User Interface for the simulated environment. The latter is implemented using the Magnum[2] framework. We extended ROBOX2D to add functionality to extract the underlying ground truth puck position data. The design of the BD generation pipeline from Experiment 1 remains unchanged. The parameters controlling the simulation and network architecture are given in Appendix B.1 and B.3.

7.2 Additional Performance Measures

7.2.1 Diversity Measures

In our metrics, we want to measure the impact of environment noise on the diversity of the produced set of solutions. However, we do not want the noise to influence the actual calculations of these performance scores. Since collisions between the pucks are now possible, we regenerate all observations in a noise-free environment in order to calculate the true diversity in the actual puck's trajectories.

The Entropy in the Puck Trajectories

In this metric, we discretise the environment into a grid and assign a counter to each bin. We extract the set of solutions from the archive that lead to a movement of the actual puck. We iterate over the solutions in this set and increase the counter of the bin in which each solution's actual puck is located at the n-th timestep in its trajectory. This creates a frequency grid over all bins. By dividing each bin by the total number of trajectories, we create a discrete probability distribution over all bins. We calculate the Shannon entropy of this distribution to obtain a score for the n-th timestep. Our diversity measure is then obtained by averaging over the entropy scores of all timesteps.
The diversity score will be the more appropriate metric in gauging the total diversity created when we compare archives containing very different proportions of solutions leading to no puck movement. However, in these situations, the score does not allow us to analyse the characteristics of the solutions that do move the puck. While the entropy score will be highly correlated with the diversity score, by excluding the solutions that do not lead to a puck movement, we create a metric that will provide a different insight into an architecture's performance.

The Variance in the Puck Trajectories

This metric is similar to the entropy score but requires no discretisation. We extract the position of the puck at the n-th timestep in its trajectory and calculate the vari-

[2] https://magnum.graphics/

ance over all extracted positions. The mean variance over the n timesteps forms our diversity metric.
The similarity captured by this metric differs substantially in its calculation from the diversity score. Despite the strong correlation between all diversity metrics, we therefore gain different insights from each one. We refer to this metric as the POSVAR score in our discussions.

7.2.2 Behavioural Descriptor Accuracy

The solution parameters are the inputs to the Encoder network in our proposed architecture. A set of solutions that differ in their parameters can therefore be assigned different BDs even if the Decoder produces the exact same construction. To gauge the accuracy with which the BDs capture the observed behaviour, we can measure the dissimilarity between the descriptors of solutions which produce the same observation.

The Proportion of Solutions Moving the Puck

The subset of solutions which do not produce any movement in the actual puck fit this description perfectly. Despite generating the exact same behaviour, these solutions' parameters differ from each other. Additionally, the associated observations might capture movements of random pucks appearing in a stochastic environment. Consequently, an appropriate measure of the BD accuracy is the proportion of solutions in the archive that lead to no actual movement of the puck. When performing optimally, the algorithm produces at most one such solution as we do not consider the arm movement itself in the BD. To align this metric with our diversity measures, we instead record the percentage of solutions that do move the puck, such that a larger score indicates a better performance. In our discussions, we refer to this metric as the PCTMOVE score.

The Variance in the No-Move Solutions' BDs

We measure the variance in the no-move solutions' latent descriptors to quantify an error on the achieved BD accuracy. This variance should equal 0 in the optimum.

7.2.3 VAE Performance Measures

The Variance in the No-Move Solutions' Constructed Trajectories

The network should produce the same construction for all no-move solutions. The variance in these constructions therefore quantifies the network's ability to filter out any environment noise in its produced outputs. With perfect noise discrimination, this variance score is equal to 0.

7.3 Results and Discussion

Figs. 7.2 and 7.3 present the diversity and entropy scores of our L2 variants and AURORA. L2-AURORA produces a rapidly deteriorating performance in both metrics as the stochasticity increases.

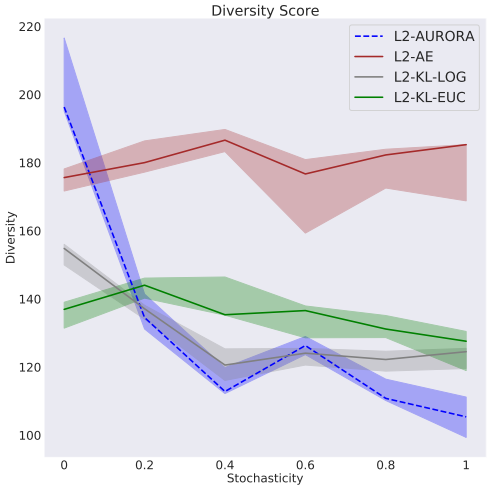


Figure 7.2: The Diversity Score.

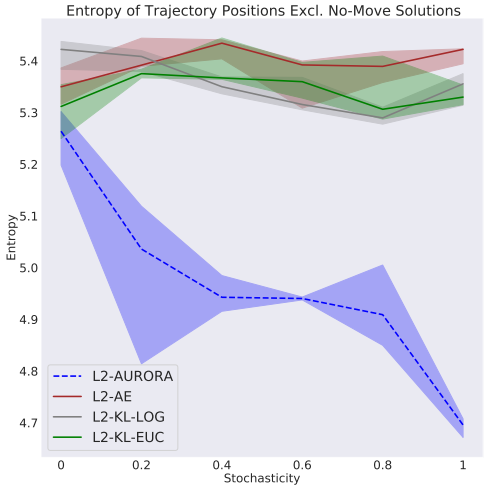


Figure 7.3: The Entropy Score.

However, when the environment is noise-free, the AURORA implementation's direct conditioning of the BD on the environment observations leads to a larger BD accuracy. In Fig. 7.4, we show the BD space populated by the AURORA baseline at 0% stochasticity in the last generation of the algorithm.
The colour-coding indicates whether a descriptor is associated to a solution that causes the actual puck to move. The inclusion of only one no-move solution indicates a maximised BD accuracy. As a result, the AURORA algorithm achieves a larger diversity score at 0% stochasticity in Fig. 7.2 than our proposed variants which produce a lower BD accuracy.

Despite the additional challenges of reducing the dimensionality and constructing more complex trajectories, our proposed architectures are again successful in producing the desired robustness to noise. Figs. 7.2 and 7.3 demonstrate a stable diversity and entropy score as the stochasticity increases.
In the diversity score, we can observe a performance difference between the VAE variants and the AE implementation. This is due to two additional architecture features in the VAE: the KL divergence criterion, and the sampling of the BD from the Encoder network.

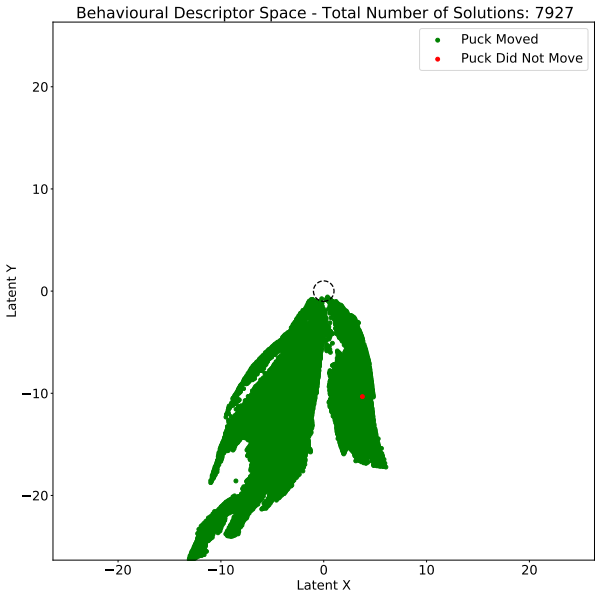


Figure 7.4: BD Space constructed by L2-AURORA at 100% stochasticity in the last generation of the algorithm. The green dots mark descriptors of solutions that move the puck. The red dots mark descriptors of no-move solutions. A unit circle is additionally plotted for an indication of the scale of the occupied space.

7.3.1 Effects of the KL Divergence Term

In our discussion of Tab. 3.1, we suggested several impacts of the imposition of a KL divergence loss term on the overall performance of our algorithm.

The first is a curtailment of the construction performance. In Fig. 7.5, the KL variants incur a larger Actual L2 error than their no-KL counterparts and the AE implementation. Unlike all other variants the Log-Likelihood implementations produce an error that is larger when the environment is noise-free than when it is stochastic. This is due to an interesting dynamic between the environment noise and the Decoder variance term in the loss function. We discuss this in detail in Chapter 7.3.5.

The next two points in our discussion on the KL criterion are centered around the potential positive impact of an isometry of the BD encoding and a disentanglement in the learned BDs.
In Figs. 7.6 and 7.7, we show a BD space constructed with and without the KL criterion respectively. In both occupied spaces we see a roughly straight line which marks the border between the set of solutions that move the puck and the solutions that do not. In Fig. 7.6 this line is vertical when the KL criterion is added. By influencing the latent space to take the shape of a standard Gaussian distribution, we promote a diagonal covariance matrix in the latent features. This leads to the disentanglement effect discussed in Chapter 2.3. In Fig. 7.6, the latent X feature

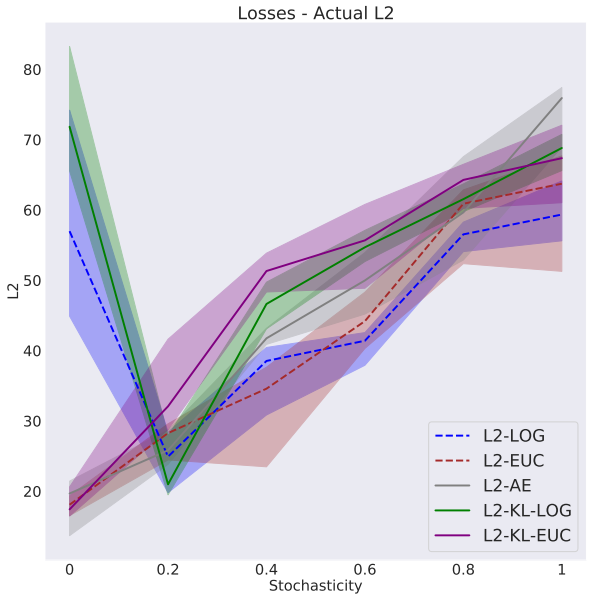


Figure 7.5: The Actual L2 Error.

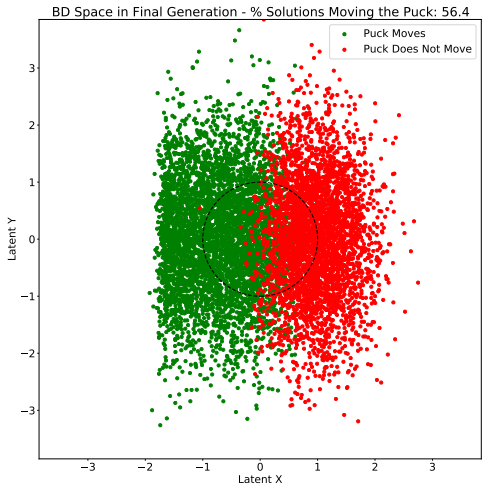


Figure 7.6: BD Space constructed by L2-KL-LOG at 100% stochasticity.

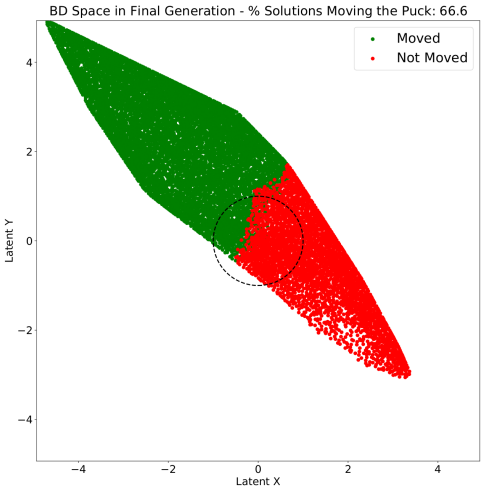


Figure 7.7: BD Space constructed by L2-LOG at 100% stochasticity.

encodes the length of the trajectory, while the Y dimension encodes its shape. This is illustrated in Fig. 7.8, in which we show two trajectory constructions associated to solutions that share the same latent Y coordinate in their descriptors. A lower value in the X feature produces a shorter trajectory construction in the bottom half of the illustration, while still following approximately the same path as the trajectory construction shown in the top half.

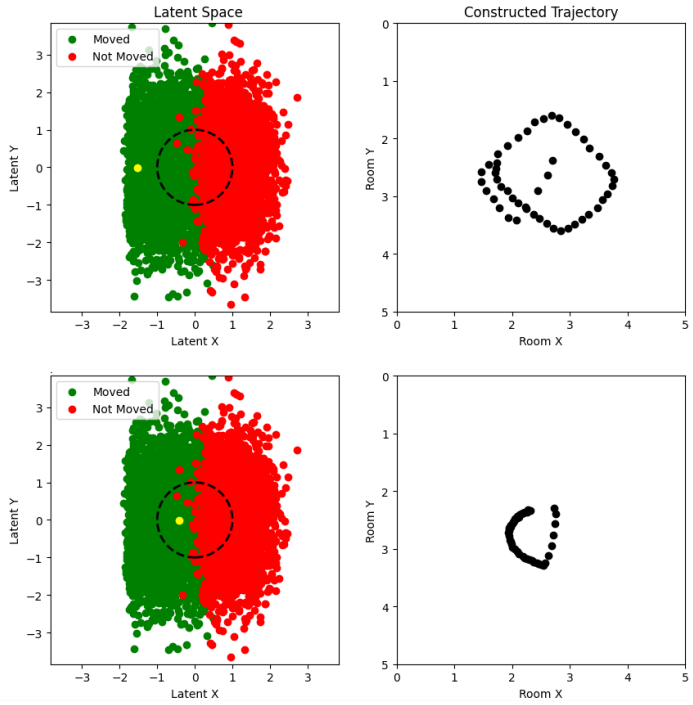


Figure 7.8: Trajectory constructions (right) associated to yellow highlighted latent codes (left).

In Fig. 7.7 this border is neither vertical nor horizontal as there is no KL criterion to enforce the disentanglement effect. However, the mere existence of the border suggests that the occupied latent space still encodes the length and shape of the trajectories in two orthogonal directions.

We confirm this in Fig. 7.9, where we present a visualisation of the trajectory constructions associated to two BDs located along the diagonal axis of the BD space.

This suggests that while the encoding does exhibit an isometry when a KL loss term is applied, we obtain representations of the observations that capture some of these desired features even without the KL criterion. With respect to this discussion point, we therefore conclude that the imposition of the KL divergence only impacts our algorithm marginally if at all.

Finally, we discussed the impact on the Encoder mean and variance. The Encoder is incentivised to produce variance values of 1 and mean values of 0 in order to achieve a small KL divergence between the distribution of the latent representations and the standard Gaussian distribution. This can result in all BDs moving closer to the latent space origin. Such a contraction of the occupied space can increase the accuracy of the created BDs by moving the descriptors of two similar solutions closer to each other.

However, to evaluate the overall impact on the diversity in the archive, we also need to consider the consequences of an increased BD space compactness on our novelty evaluation mechanism. At this point, it is important to note that we do not start training the VAE until we reach the 10^{th} generation of the algorithm. The set of

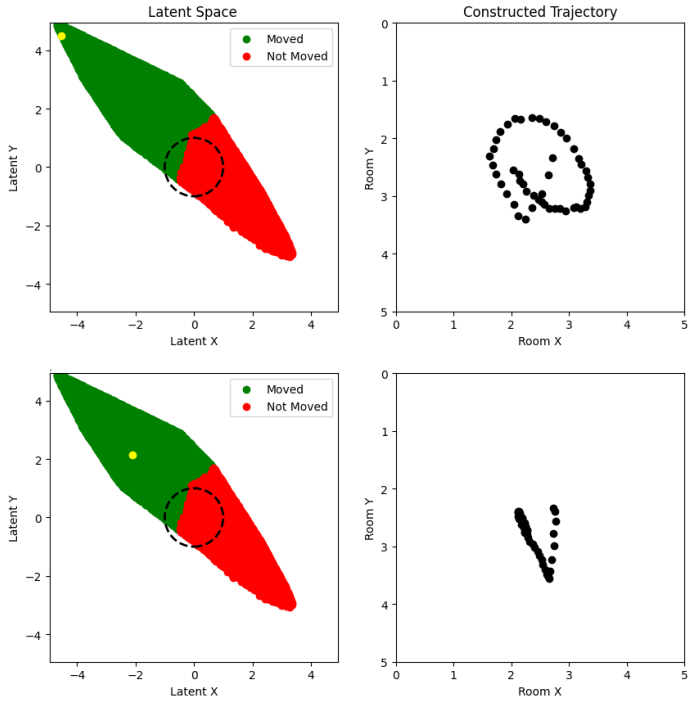


Figure 7.9: Trajectory constructions (right) associated to yellow highlighted latent codes (left).

BDs we use to initialise the novelty threshold parameter L in the first generation is therefore largely random. This implies that the composition of the loss function and the presence of the KL criterion in particular, do not have an impact on this initial value. Moreover, as discussed in Chapter 4, all subsequent updates to L consider only the difference in the archive's size from its target size. They do not consider the density in the BD space. Consequently, the progression in the value of L is very similar for all our architectures in the initial generations of the algorithm.
On the other hand, when descriptors are positioned closer to each other, more solutions are accepted into the archive at any given value of L. In the KL variants, an increased compactness in the BD space would therefore also result in a faster growing archive size. This in turn translates into a slower increase in the value of L as the number of generations increases.
Fig. 7.10 shows the value of L at the end of each generation in our NST-EUC variants with and without the KL divergence criterion. We use the data from these variants as they provide a less distorted picture of the influence of a BD space contraction on the value of L for reasons that will become apparent in the subsequent discussion. We can observe that the values only start diverging after approximately 300 generations after which they consistently remain lower for the KL implementation.

Increases to the compactness of the BD space therefore produce a trade-off. On one hand, the BD accuracy increases as descriptors move closer together. At a given threshold value this results in a larger number of solutions with similar behaviours to be rejected than before. On the other hand, at a given large generation number, this threshold is lower. This leads to the acceptance of a greater number of

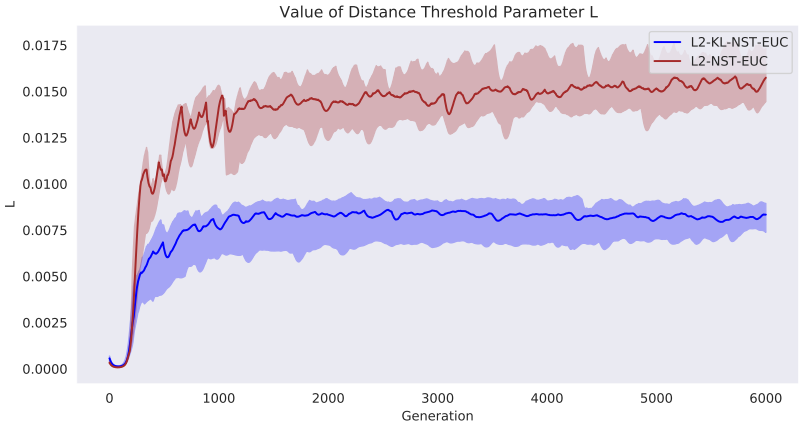


Figure 7.10: Progression of the value of the threshold parameter L at 0% stochasticity in the NST-EUC variants.

similar solutions than before. An appropriate measure to evaluate the outcome of this trade-off is the number of no-move solutions in the archive as they produce an identical behaviour. If the reduction in the distance between descriptors outweighs the relative decrease in the value of L, an increased compactness will result in the acceptance of fewer no-move solutions.

Fig. 7.11 shows that the KL variants produce a lower score in this metric. While

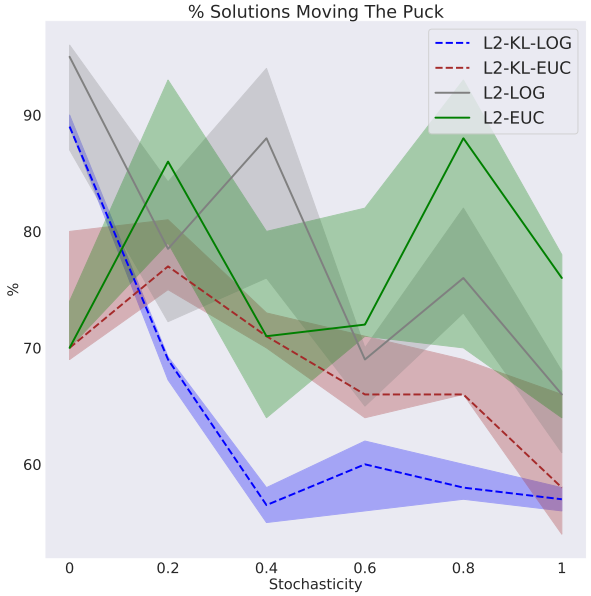


Figure 7.11: The proportion of solutions moving the puck.

this seems to all but confirm an overall negative impact of the KL criterion, the En-

coder variances shown in Fig. 7.12 prevent us from drawing that conclusion just yet. Fig. 7.12 illustrates that the Encoder learns to produce a variance of 0 when we do not include the KL criterion. The addition of the KL loss term therefore leads to an increased Encoder variance. Note, that the variance increases further as the noise level grows. As the environment stochasticity increases, the gradient of the construction loss decreases in magnitude. This is due to the inclusion of a larger number of noisy trajectories into a mini-batch which therefore increases in size. As some of the noisy trajectories will produce contradicting gradients, this results in a smaller mean gradient. This reduction in its magnitude shifts some of the construction criterion's weight in the network's gradient update to the gradient of the KL loss whose magnitude remains unaffected by the noise. In Fig. 7.13, we therefore observe a KL loss that decreases in the environment stochasticity. These lower losses in turn explain the larger Encoder variances.

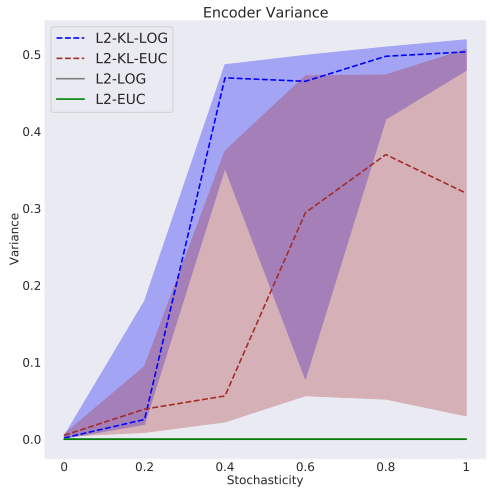


Figure 7.12: The Encoder variance.

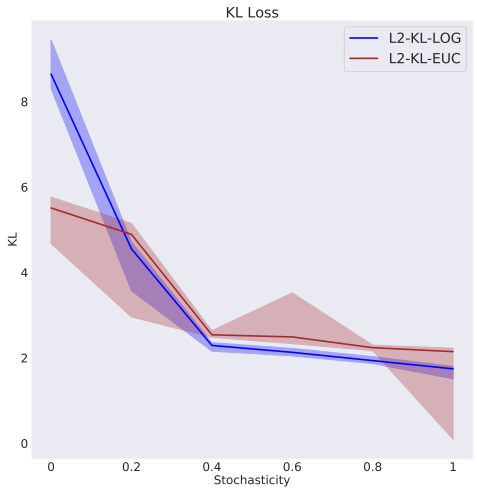


Figure 7.13: The KL Loss.

Given the KL variants' larger Encoder variance values, we cannot say for certain whether the addition of the KL criterion results in the BD space compression that we want to analyse or rather an expansion of the occupied space. Their lower proportions of solutions moving the puck in Fig. 7.11 translate into significantly lower diversity scores in Fig. 7.14. Both L2 variants outperform their L2-KL counterparts and produce a similar diversity score as the AE implementation. While we cannot use these results to draw a clear conclusion on the impact of a BD space contraction, they do indicate that we can likely improve the KL variants' performance significantly by eliminating the Encoder variance term.

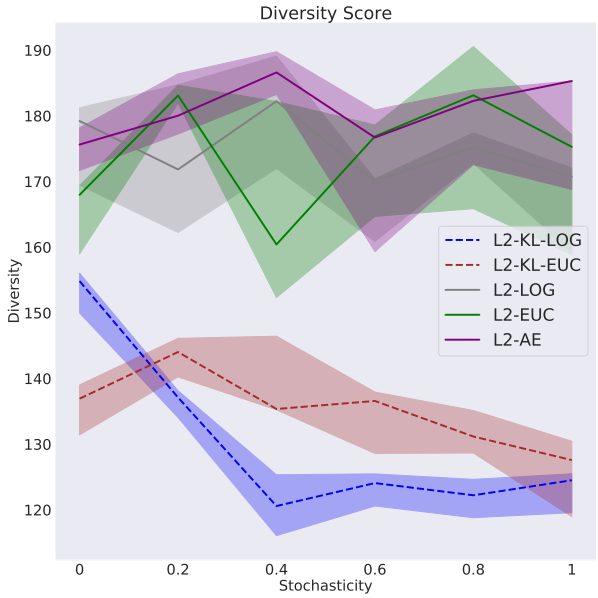


Figure 7.14: The Diversity Score.

7.3.2 Sampling from the Encoder during Inference

The apparent optimality of our L2 variants' low Encoder variances coincides with the second factor explaining the performance difference between the L2-AE and L2 implementations: the decision to sample the BD from the Encoder.

We now modify our variants to assign the produced Encoder mean as the BD of a given solution. Given their minimal learned Encoder variances, the L2 variants effectively already mimick the L2-NS configurations. This similarity unsurprisingly leads to the same performance across all our metrics. We demonstrate this using the PCTMOVE and POSVAR scores in Figs. 7.15 and 7.16.

For the KL variants on the other hand, the scores achieved in our metrics change significantly when we stop sampling during inference. Fig. 7.17 shows a reduction in the variance in the no-move solutions' latent descriptors. This indicates smaller distances between the BDs and therefore a contraction in the occupied BD space. Figs. 7.18 and 7.19 show an improved PCTMOVE metric and diversity score respectively. These results suggest that an increased compression of the BD space does indeed have a positive impact on our algorithm.

Given the KL variants' substantial performance gains, the size of the relative performance differences between the KL and no-KL variants now changes substantially from the results obtained in the previous section. The contraction of the occupied space that follows the imposition of the KL criterion is very visible. The L2-KL-NS-LOG implementation constructs a much smaller sized BD space in Fig. 7.20, than the

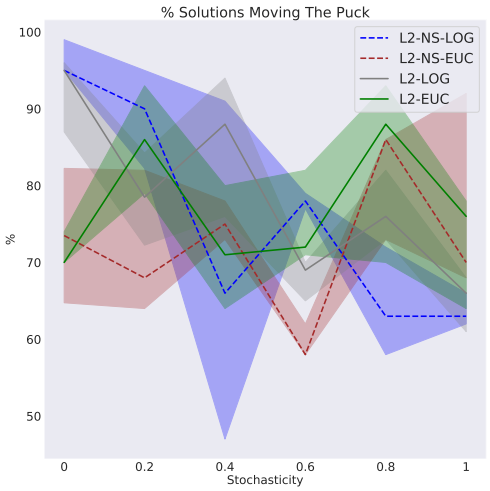


Figure 7.15: The proportion of solutions moving the puck.

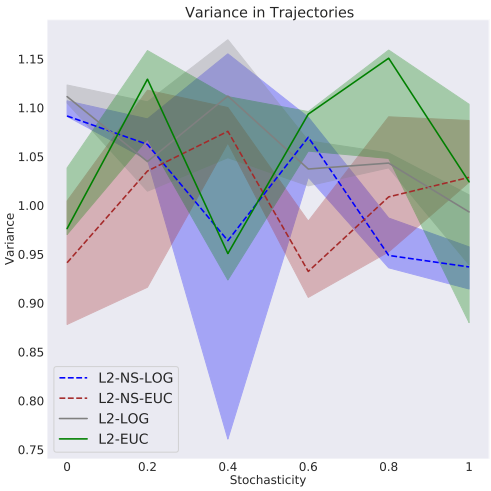


Figure 7.16: The variance in the puck trajectories.

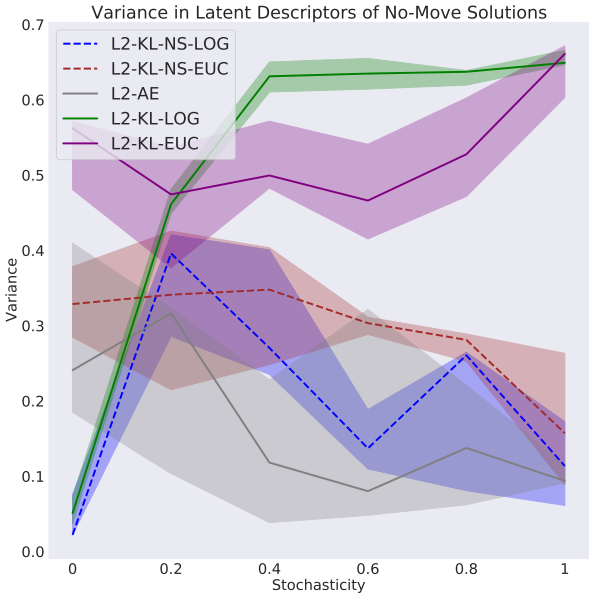


Figure 7.17: The variance in the no-move solutions' latent descriptors.

BD space constructed in Fig. 7.21 by the L2-NS-LOG variant.

We present the PCTMOVE and POSVAR scores achieved by these variants in Figs. 7.22 and 7.23. Both scores seem to indicate that the addition of the KL criterion has a positive impact overall. However, given the small magnitude of these performance differences and the large interquartile ranges captured by the shaded areas in the

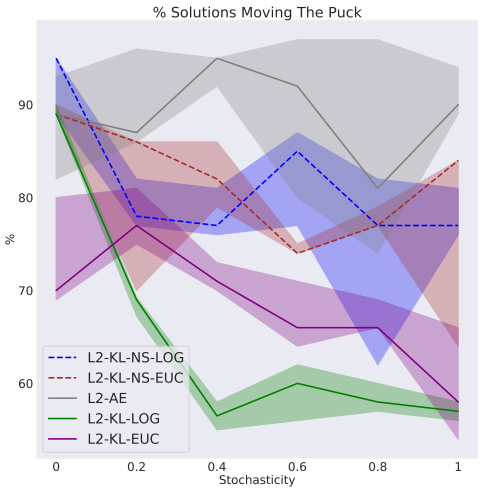


Figure 7.18: The proportion of solutions moving the puck.

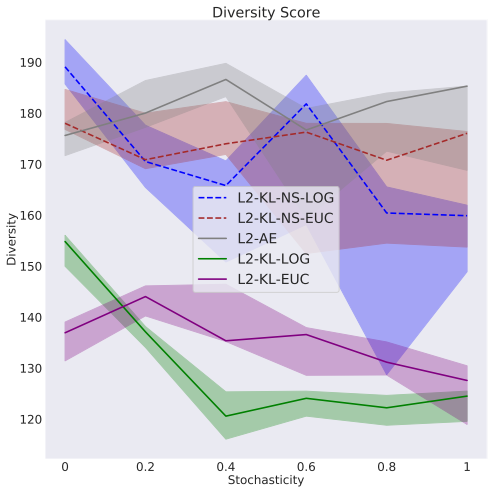


Figure 7.19: The Diversity Score.

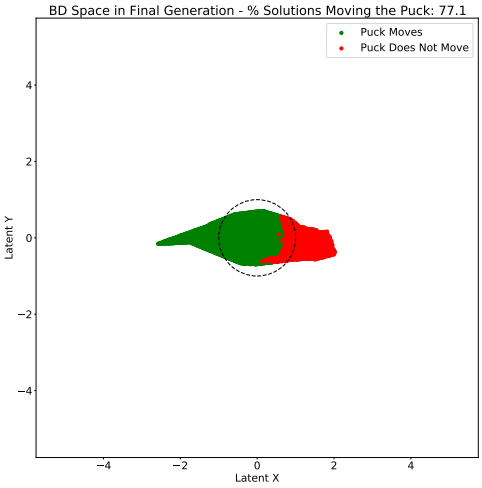


Figure 7.20: BD Space constructed by L2-KL-NS-LOG at 100% stochasticity.

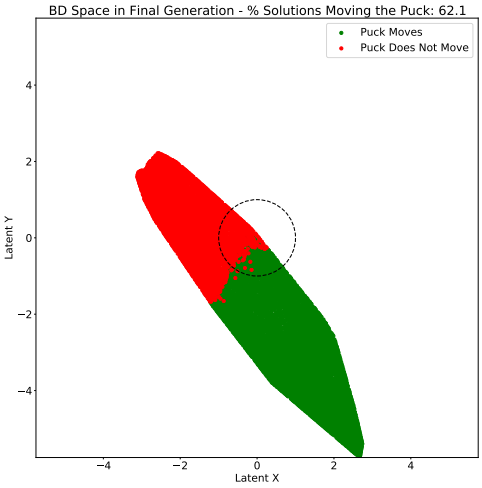


Figure 7.21: BD Space constructed by L2-NS-LOG at 100% stochasticity.

graphs, we cannot make this conclusion unequivocally. However, note, that the influence of the KL variants' larger Encoder variance values is not fully eliminated yet, as they remain in use during the training of the VAE.

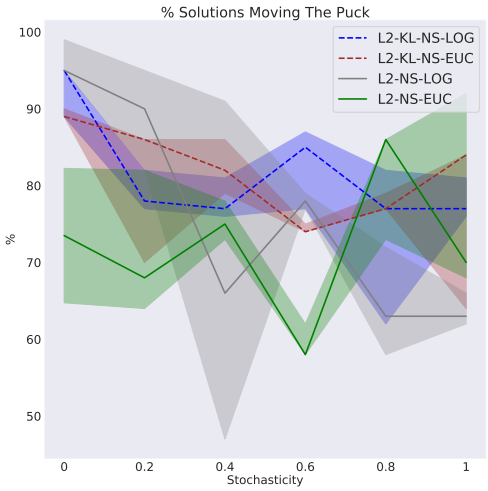


Figure 7.22: The proportion of solutions moving the puck.

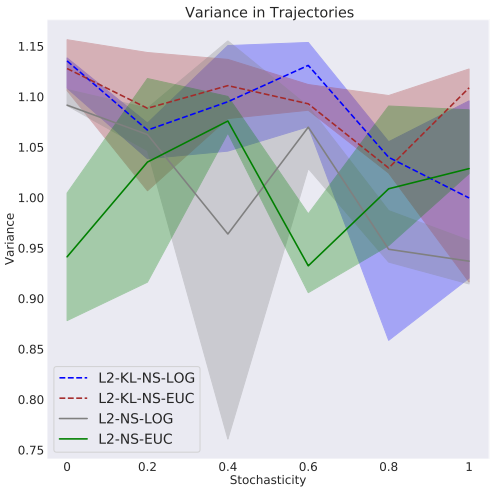


Figure 7.23: The variance in the puck trajectories.

7.3.3 Sampling from the Encoder during Training

When we sample during training, the VAE repeatedly experiences that the relationship between the Encoder outputs and the generated latent codes is nondeterministic. The network therefore learns to produce similar constructions for all latent descriptors situated in a neighbourhood around a given Encoder mean. Consequently, Encoder means for similar observations are placed further apart such that the network can separate their subsequently sampled latent codes. As the Encoder variances increase these neighbourhoods grow larger. When we assign the Encoder means as the BDs, this leads to an expansion of the occupied space.

Figs. 7.24 and 7.25 present the BD spaces constructed by the L2-KL-NST-LOG and L2-KL-NS-LOG variant respectively. The visualisations are scaled to the same size. All Encoder means produced by the L2-KL-NST-LOG variant fall well within the unit circle in Fig. 7.24. On the other hand, the space constructed by the L2-KL-NS-LOG variant in Fig. 7.25 is much larger in size and gaps between the means are still visible.
The NST variants' increased compactness in the BD space yields a stronger performance. Figs. 7.26 and 7.27 show larger PCTMOVE and POSVAR scores when the KL variants do not sample during training.

The L2-NS variants' learned Encoder variances are much smaller than the KL variants' but importantly not equal to zero. Moreover, we previously observed a rise in their magnitudes as the stochasticity increased. Fig. 7.28 shows this more clearly. The optimality of larger Encoder variances for greater levels of stochasticity suggests that the creation of larger latent representation neighbourhoods is beneficial to the VAE's construction accuracy in the presence of noise. The presence of the Encoder

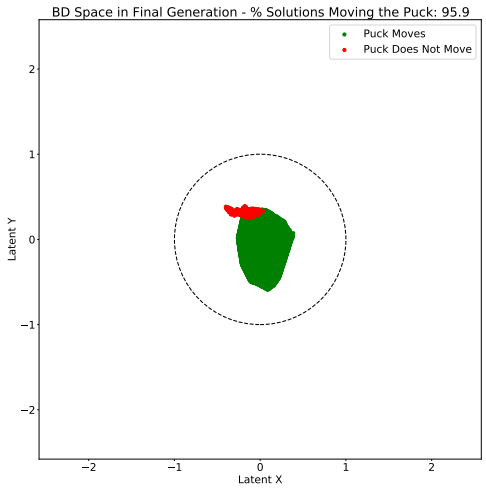


Figure 7.24: BD Space constructed by L2-KL-NST-LOG at 100% stochasticity.

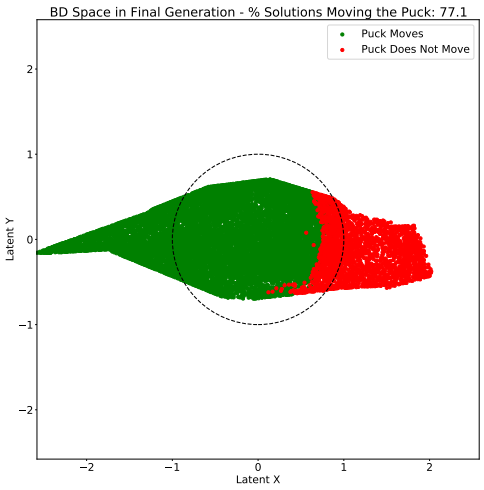


Figure 7.25: BD Space constructed by L2-KL-NS-LOG at 100% stochasticity.

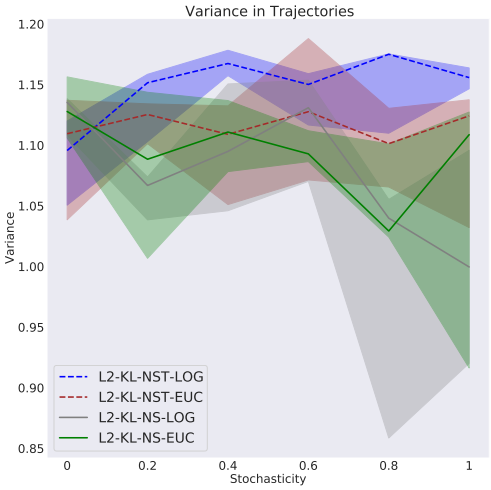


Figure 7.26: The variance in the puck trajectories.

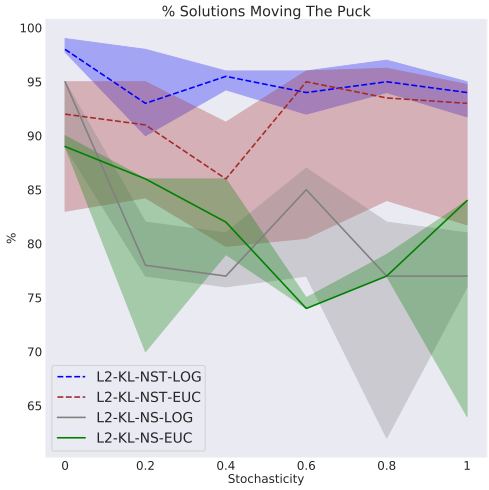


Figure 7.27: The proportion of solutions moving the puck.

variance term allows the network to incorporate the information it has gained from the solutions' observations to vary its certainty with which it creates their latent representations. This improves the overall construction ability. In Fig. 7.29 we illustrate the L2-NST and L2-NS implementations' variances in the constructions made for the no-move solutions. At large stochasticity levels, the L2-NST implementations produce larger variances which indicate a decreased construction accuracy.

Despite their small magnitudes, the L2-NS variants' Encoder variances still lead to a weaker compression of the BD space when we sample during training. In Fig. 7.31,

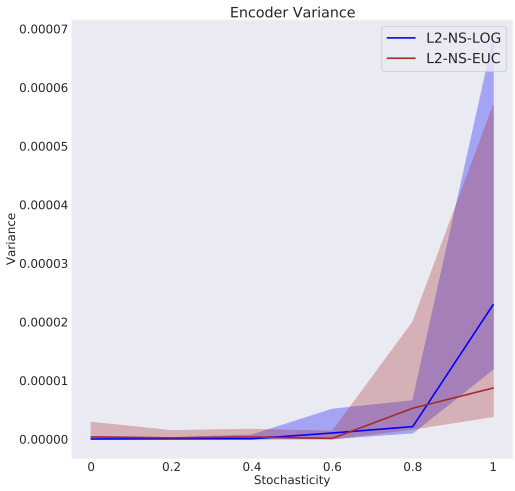


Figure 7.28: The Encoder variance.

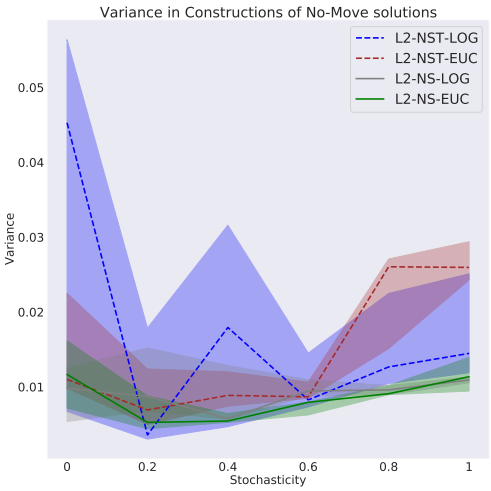


Figure 7.29: The variance in the no-move solutions' constructions.

the BD space constructed by the L2-NS-LOG variant is significantly larger in size than the space created without any sampling during training in Fig. 7.30.

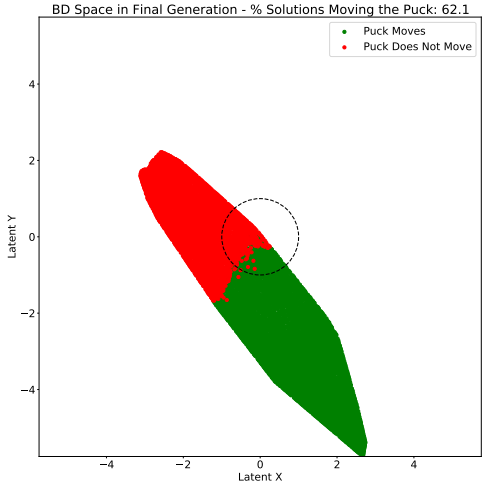


Figure 7.30: BD Space constructed by L2-NS-LOG at 100% stochasticity.

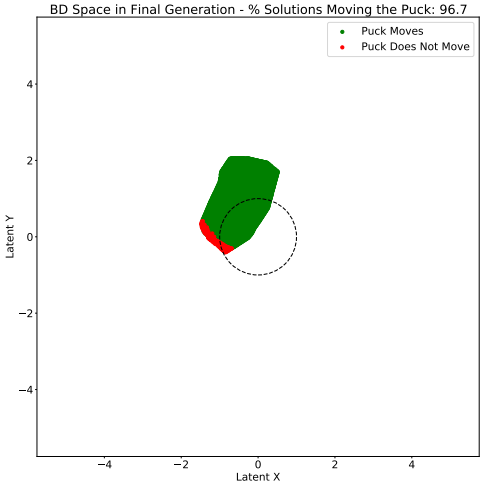


Figure 7.31: BD Space constructed by L2-NST-LOG at 100% stochasticity.

The diversity metrics indicate that these large increases in the compactness outweigh any marginal impairments to the construction accuracy. Figs. 7.32 and 7.33 show improvements in the variants' PCTMOVE and POSVAR scores when we remove the Encoder variance term.

With the influence of the Encoder variance now fully eliminated from all our vari-

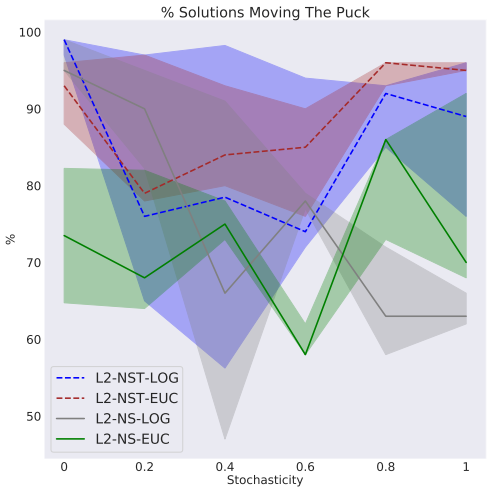


Figure 7.32: The proportion of solutions moving the puck.

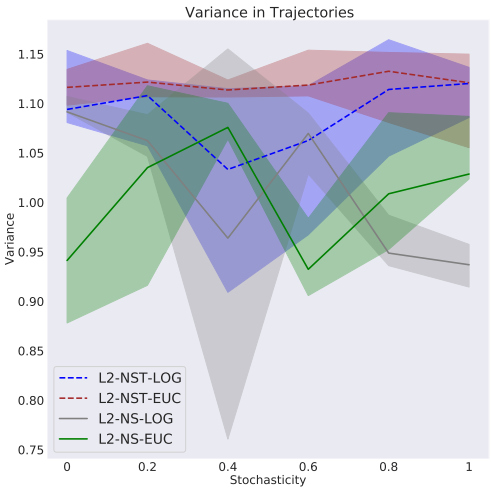


Figure 7.33: The variance in the puck trajectories.

ants, we can make a final assessment of the impact that the size of the occupied BD space has on the algorithm's performance.
The application of the KL divergence criterion in our NST variants significantly increases the BD space compactness. This is demonstrated by a substantially lower variance in the no-move solutions' latent descriptors in Fig. 7.34.

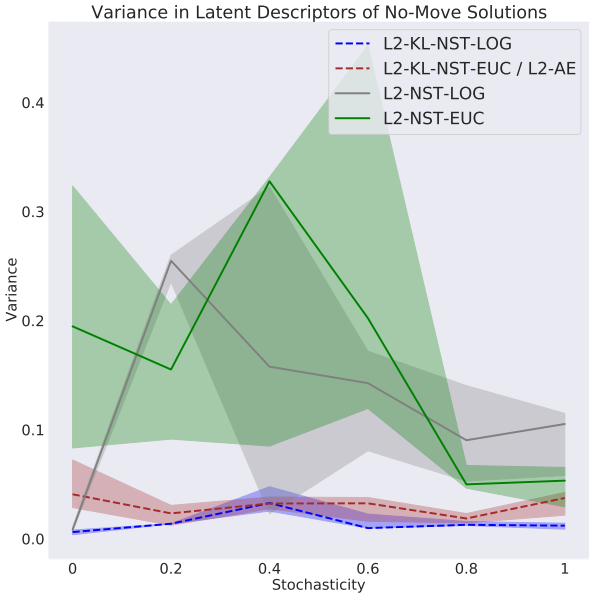


Figure 7.34: The variance in the no-move solutions' latent descriptors.

The result is a larger PCTMOVE and POSVAR score in Figs. 7.35 and 7.36. The in-

crease is especially significant for the Log-Likelihood variant which produces the best results of all analysed architectures so far. We discuss the relative performance differences between the Log-Likelihood and Euclidean implementations in Chapter 7.3.5.

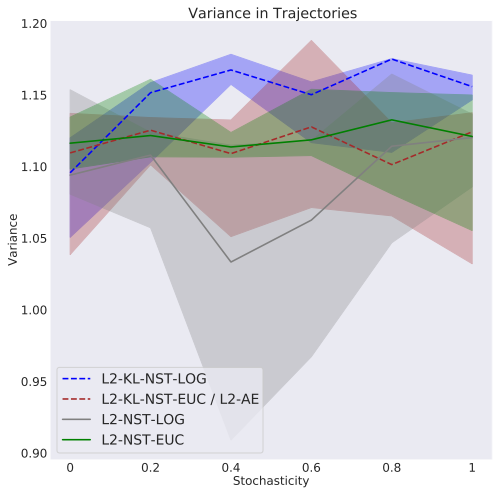


Figure 7.35: The variance in the puck trajectories.

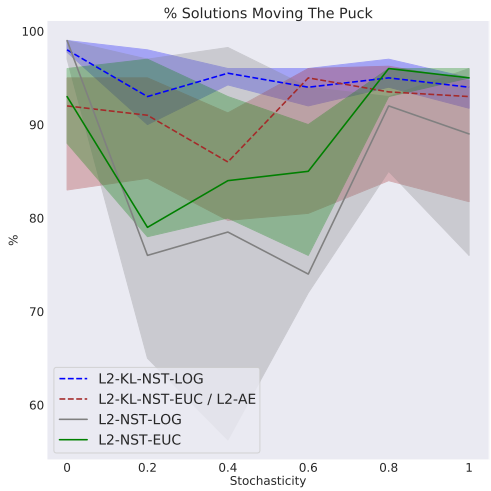


Figure 7.36: The proportion of solutions moving the puck.

7.3.4 SNE and t-SNE

In the SNE and t-SNE variants, we also attempt to produce a compact placement of similar solutions' latent descriptors. However, rather than pulling all representations closer towards the origin, these methods focus on preserving the local neighbourhood structure of the high-dimensional observation space when creating the latent space. This provides an explicit objective to map solutions with the same associated observations to the same BD.

Replacing the KL criterion with a SNE criterion results in a significant increase in the PCTMOVE scores in Fig. 7.37. The scores are similar between the L2-SNE-NST-LOG and the L2-SNE-NST-EUC variants when the environment is stochastic. However, the former produces a significantly larger proportion of no-move solutions when the environment is noise-free. This is due to an instability in the LOG variants' optimisation process that is particularly strong when the environment is noise-free. We analyse this irregularity in detail in Chapter 7.3.5.

Interestingly, however, despite producing a much larger PCTMOVE score than the KL variants, the metrics derived from the puck trajectories' characteristics indicate that the diversity is significantly lower when we apply the SNE criterion. This is shown in the POSVAR and diversity scores in Figs. 7.38 and 7.39. Furthermore, we see in both metrics that the SNE variants are very susceptible to noise. This follows from our use

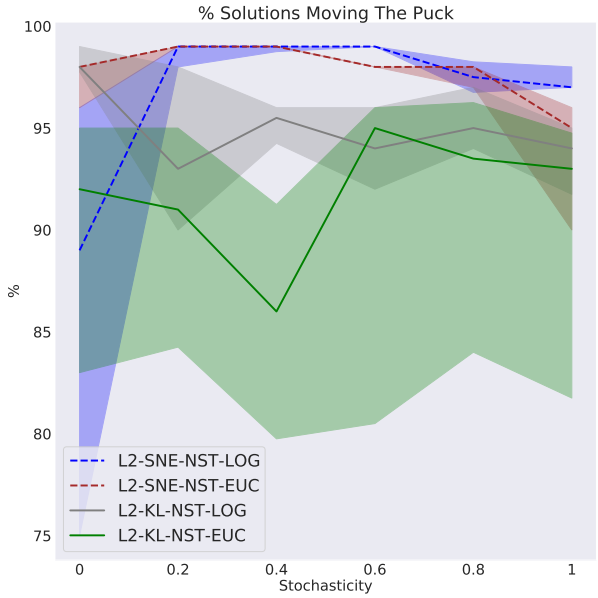


Figure 7.37: The proportion of solutions moving the puck.

of the network's trajectory constructions in the SNE calculations. The deterioration in the network's construction accuracy caused by rising noise levels has a significant and much more direct impact on the BDs than in our KL variants. This also leads to the acceptance of a larger proportion of no-move solutions into the archive at the largest levels of stochasticity in Fig. 7.37. We analyse this in detail further along in this section.

Nonetheless, even at moderate levels of stochasticity are the SNE variants unable to convert a significantly larger PCTMOVE score into an improved diversity performance. This result is surprising. From our findings so far we have learned to expect a larger proportion of solutions moving the puck to translate into performance gains in our diversity metrics. This irregularity is a consequence of the SNE criterion's impact on the archive's composition.

Fig. 7.40 shows that the solutions in the archives generated by the SNE variants send the puck on longer trajectories compared to our KL variants. Note, that the distance values shown in the graph are not the straight line distances between the pucks' start and end positions but the distances that the puck travels along its path.

In itself, this tendency to produce solutions with longer trajectories does not explain a lower diversity. Indeed, we would expect larger distances to result in larger scores across all our diversity metrics but especially in the diversity score which measures the number of bins transversed by the puck. However, only in a noise-free environment do the SNE variants produce a diversity score that is similar to the KL variants' in Fig. 7.40.

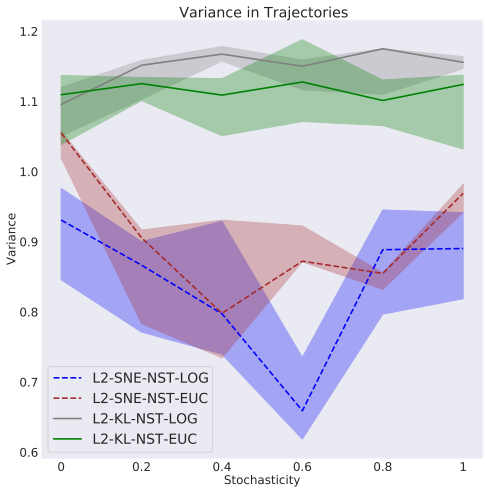


Figure 7.38: The variance in the puck trajectories.

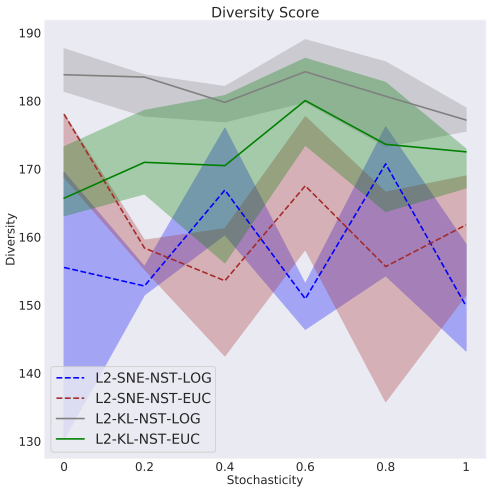


Figure 7.39: The Diversity Score.

While the trajectory distances increase, the variance of these measured distances decreases significantly. This is illustrated in Fig. 7.41. Overall, this implies that the generated puck trajectories are not only generally longer but also more exclusively so. The rise in the variance at higher levels of stochasticity can again be attributed to the deterioration of the network constructions performance. We discuss this further along in this section.

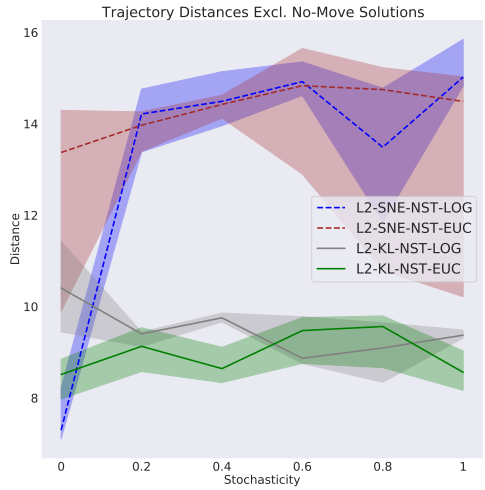


Figure 7.40: The trajectory distances of solutions that move the puck.

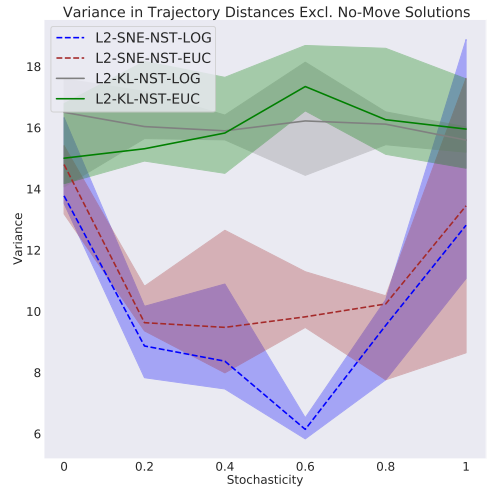


Figure 7.41: The variance in the trajectory distances of solutions that move the puck.

The results obtained so far suggest that these longer trajectories are dissimilar enough to be added to the archive, yet too similar to increase the diversity measures significantly. As a consequence of this clustering of trajectories, large spaces of the room are not frequented as often and as diversely in the SNE variants, as they are in our KL implementations. This leads to an overall poorer diversity.
In Fig. 7.42, we illustrate for a discretised grid of the room the average number of solutions whose trajectories end in a given bin. The high frequency values in a select few bins indicate large clusters of similar trajectories. Conversely, the KL variants produce a set of solutions that produce a much more uniform frequency distribution over all bins in Fig. 7.43.

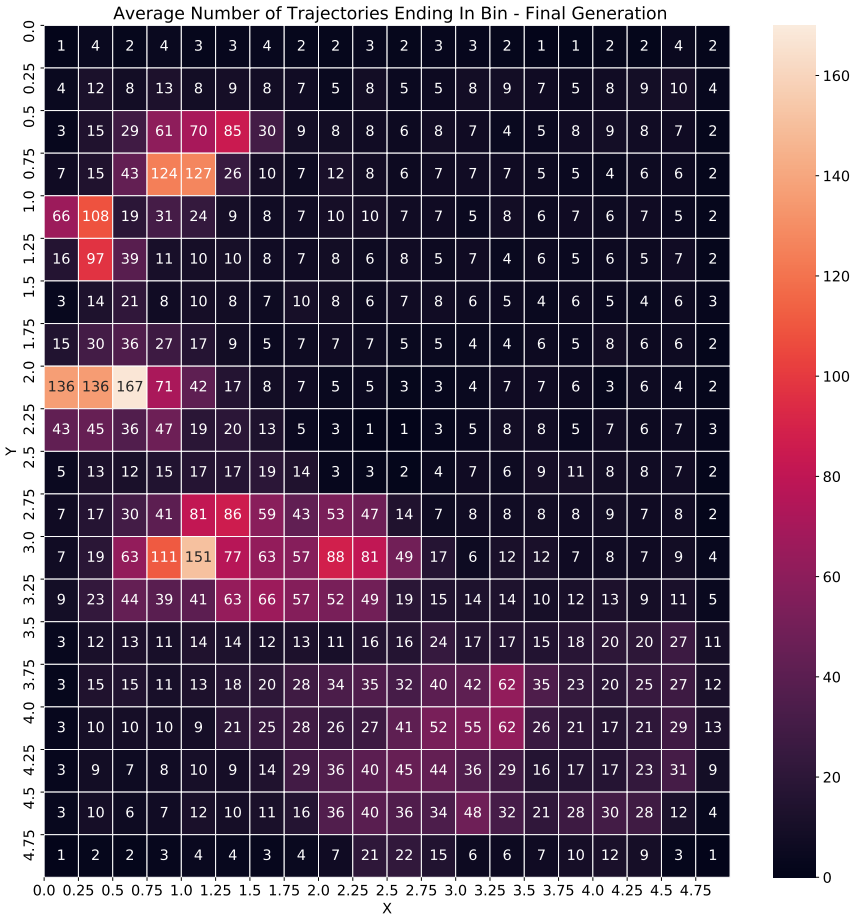


Figure 7.42: Average number of trajectories ending in each bin in L2-SNE-NST-LOG over 5 repetitions at 100% stochasticity. Annotations in each bin display the frequency, rounded to the nearest integer.

To confirm a large similarity of these long trajectories, we present in Fig. 7.44 a vi-

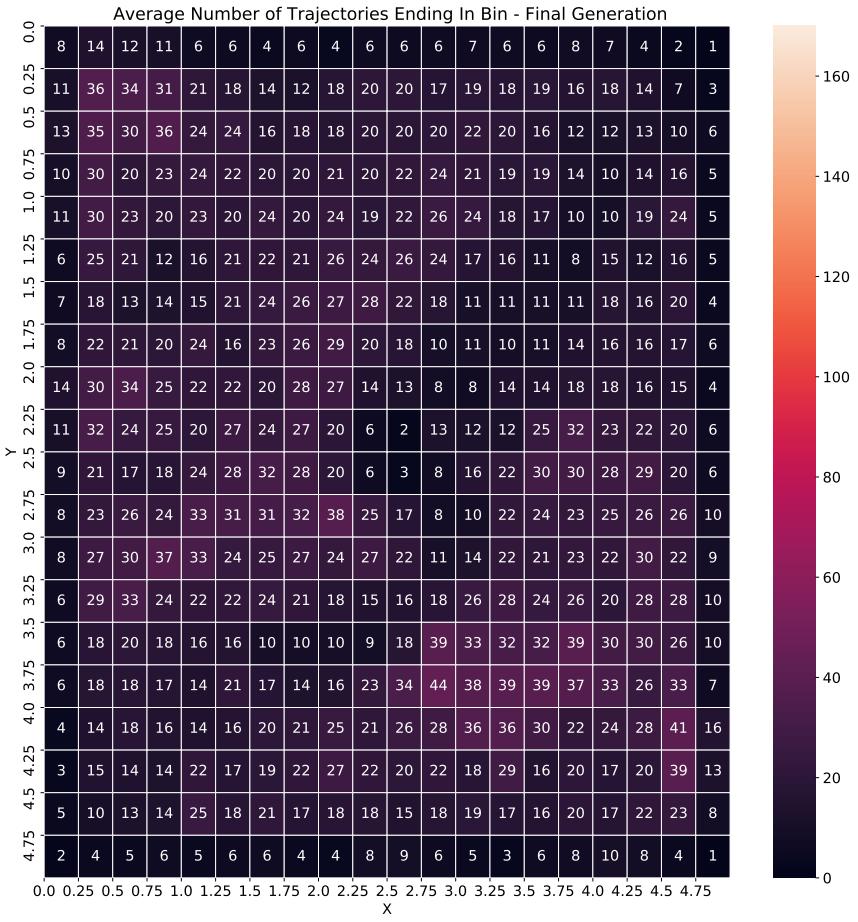


Figure 7.43: Average number of trajectories ending in each bin in L2-KL-NST-LOG over 5 repetitions at 100% stochasticity. Annotations in each bin display the frequency, rounded to the nearest integer.

sualisation of all trajectories in 5 archives generated by L2-SNE-NST-LOG that end in the visualised red box. This box corresponds to the bin with the largest frequency value of 167 in the SNE variant's frequency grid in Fig. 7.42. The bulk of these trajectories follow the same two paths thus generating a low diversity score despite the large number of trajectories. A large similarity between long trajectories also explains why the high frequency bins tend to occur in patches in Fig. 7.42.

In the diversity score calculations, we apply the same discretisation as shown in the frequency grids. Figs. 7.45 and 7.46 present the average diversity score for each bin. Note the relatively low diversity scores in the SNE implementation in the lower triangular part of the grid despite the large number of trajectories in some of the bins.

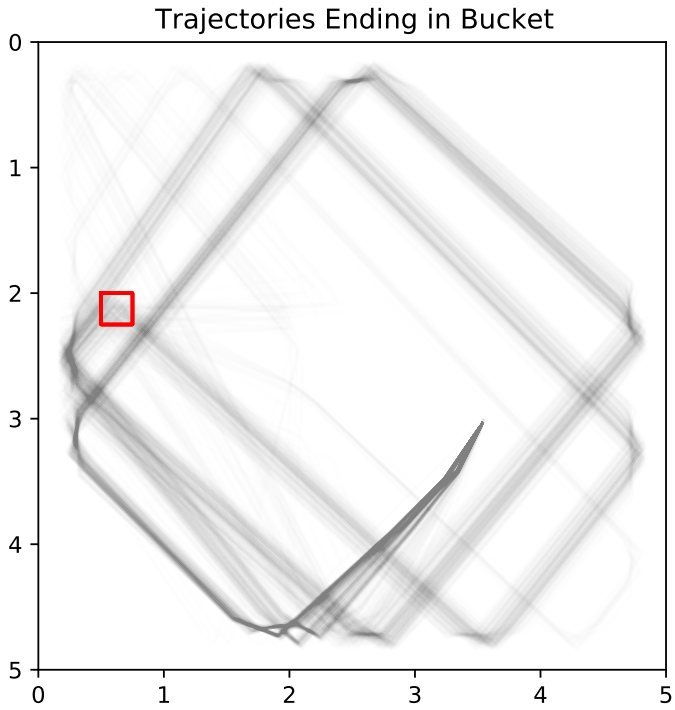


Figure 7.44: Actual puck trajectories ending in red box in 5 archives generated by L2-SNE-NST-LOG at 100% stochasticity.

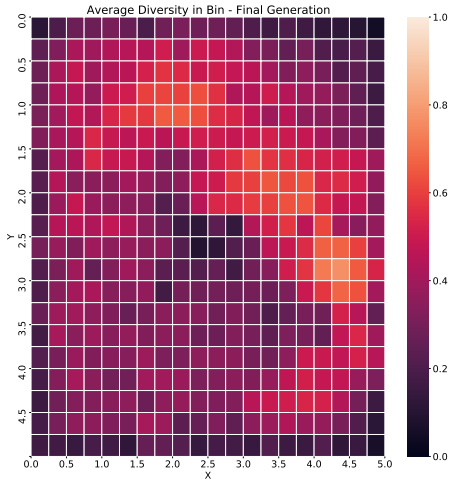


Figure 7.45: Average diversity score of each bin in L2-SNE-NST-LOG over 5 repetitions at 100% stochasticity.

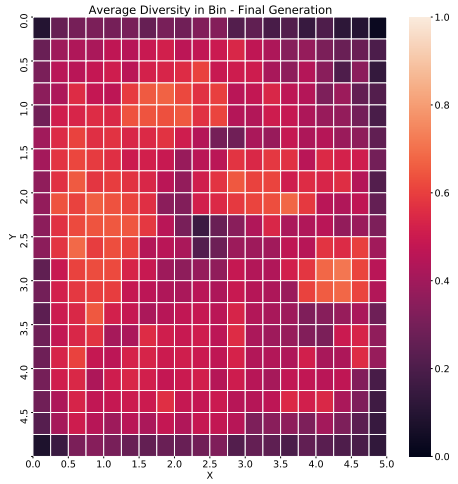


Figure 7.46: Average diversity score of each bin in L2-KL-NST-LOG over 5 repetitions at 100% stochasticity.

To understand why the SNE variants tend to generate solutions with significantly

longer trajectories, we need to consider the calculations made to determine the similarity between observations. As discussed in Chapter 2.3, the application of the SNE algorithm can often lead to a crowding of low-dimensional representations in the latent space. In our application, the PCTMOVE metric benefited heavily from this effect. For all no-move solutions that receive a correct observation construction from the VAE, the SNE criterion exercises a large pressure on their representations to collapse towards the same location in the BD space.
However, when our observations are made up of the puck trajectories' coordinates, this also exerts a disproportionally larger pressure on short trajectories than long trajectories. The SNE component introduces a strong relationship between the distance travelled by solutions' trajectories and the distance between their representations. Consider the two pairs of trajectories shown in Fig. 7.47.

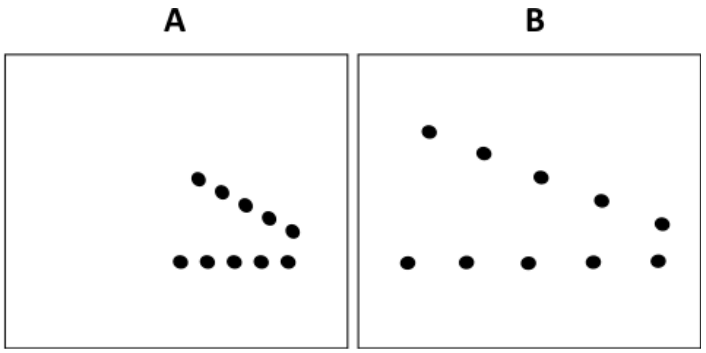


Figure 7.47: Two pairs of trajectories.

Pair A consists of two short trajectories that move in a straight line. If we "stretch" these trajectories to increase their lengths while preserving their directions, we obtain Pair B in the right box. Each pair of trajectories is likely to be considered equally similar by many. However, the sum of pair-wise distances between the elements of the trajectories is larger for Pair B. In the SNE algorithm, this decreases the similarity of the trajectories in Pair B. This in turn leads to a reduced target similarity when placing their descriptors into the latent space. All else equal, the two long trajectories' representations will therefore be further apart than the short trajectories' descriptors. The SNE criterion introduces an incentive to use a smaller BD subspace to store the representations of the shortest 5% of trajectories, and a relatively larger subspace for the longest 5% of trajectories.

The increased compactness among the short solutions translates into a larger perceived similarity in our algorithm. Consequently, a larger proportion of a given set of short solutions will be rejected from the archive when we add the SNE criterion. Conversely, a larger proportion of a given set of long solutions will be accepted into the archive. Fig. 7.48 provides a mock illustration of this mechanism. In this illustration, shorter solutions are placed closer together, with the solutions shaded in light blue failing to be accepted into the archive at the given novelty distance threshold as they are too close to existing solutions.

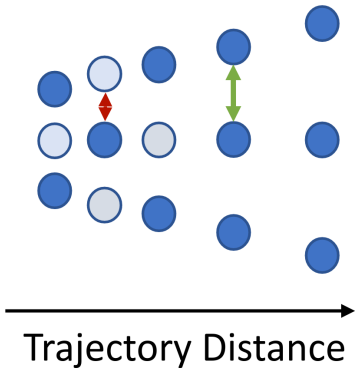


Figure 7.48: Solutions arranged in a fictive latent space. Solutions with longer trajectories are placed further to the right of the illustration and are more distant from each other at a given trajectory distance. The red arrow indicates that the given distance is below the minimum novelty threshold. The green arrow indicates an acceptable distance.

This disproportionality is further exacerbated by our use of the VAE's constructions as the high-dimensional data points. While the VAE constructions typically match the shape, they are often much shorter than the actual trajectory when the environment is stochastic. This is the result of the network's attempt to minimise the loss on the random trajectories by moving all elements of the construction closer towards the center of the room. We illustrate this observation in Figs. 7.49 and 7.50 in which we show two trajectories and their constructions at 0% and 100% stochasticity respectively. Shorter trajectories will receive even shorter constructions which increases their perceived similarity yet again.

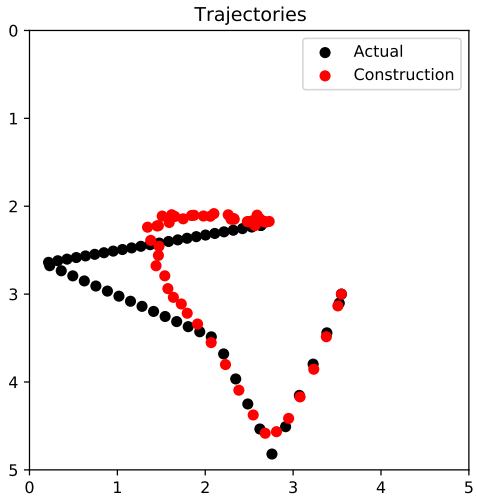


Figure 7.49: Actual trajectory and construction made by L2-SNE-NST-EUC at 0% stochasticity.

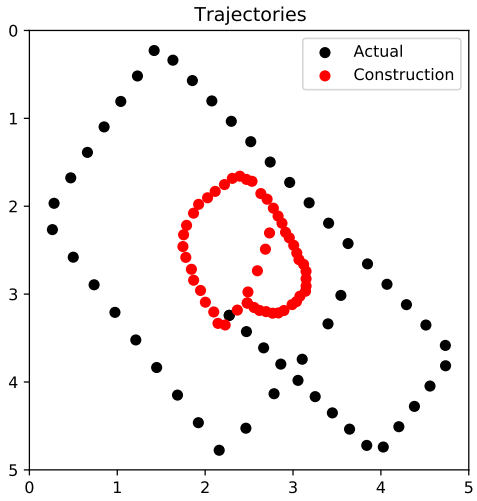


Figure 7.50: Actual trajectory and construction made by L2-SNE-NST-EUC at 100% stochasticity.

We can visualise the large changes to the distribution of trajectory distances by colour-coding the solutions' BDs with the distances travelled by their associated trajectories. In Fig. 7.51 we present a BD space constructed by the SNE variant under 100% stochasticity. When we compare this to the BD space constructed by the KL variant in Fig. 7.52, we can see that the proportion of long distance solutions is substantially larger for the SNE variant.

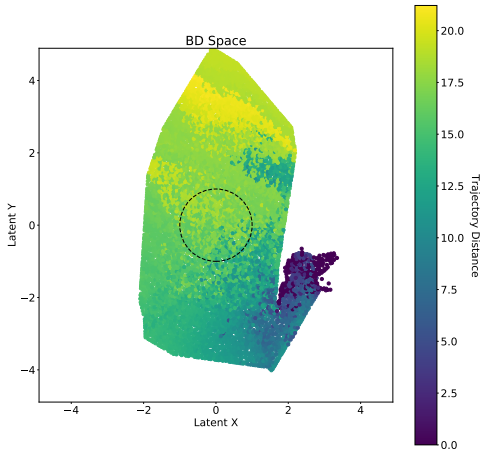


Figure 7.51: BD Space constructed by L2-SNE-NST-LOG at 100% stochasticity. Colour bar indicates the distance travelled by the trajectory.

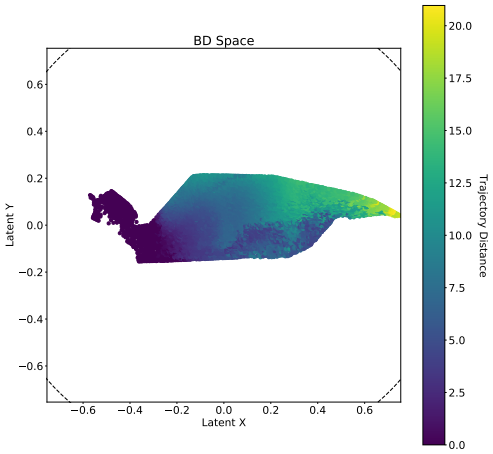


Figure 7.52: BD Space constructed by L2-KL-NST-LOG at 100% stochasticity. Colour bar indicates the distance travelled by the trajectory.

Equivalently, in the KL variant we expect a smooth and linear change in the trajectories' distances as we move from one end of the occupied BD space to the other. Conversely for the SNE variant, we expect the crowding of short solutions to yield a rate of change that is larger in the subspace containing the shortest solutions than at the other end of the space. To illustrate this, we plot the average absolute difference in the distance travelled by a given solution's puck trajectory compared to the trajectories of its 30 nearest neighbours in the BD space. Fig. 7.53 shows that for the SNE variant, this rate of change is indeed higher in the low-distance subspace than in the high-distance subspace. The KL implementation shows a much smoother progression in Fig. 7.54.

Finally, note that this disproportional rejection of short trajectory solutions only occurs when the distance between their representations is shorter than the novelty distance threshold L. If this was not the case, we would expect to see larger gaps between the descriptors of long-distance solutions than between the descriptors of short solutions. However, Figs. 7.55 and 7.56 demonstrate a uniform BD space in both the KL and SNE variant.

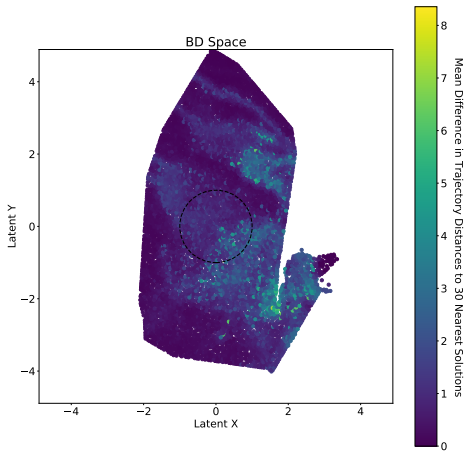


Figure 7.53: BD Space constructed by L2-SNE-NST-LOG at 100% stochasticity. Colour bar indicates the mean difference in trajectory distances of each solution compared to the trajectory distances of the 30 most similar solutions.

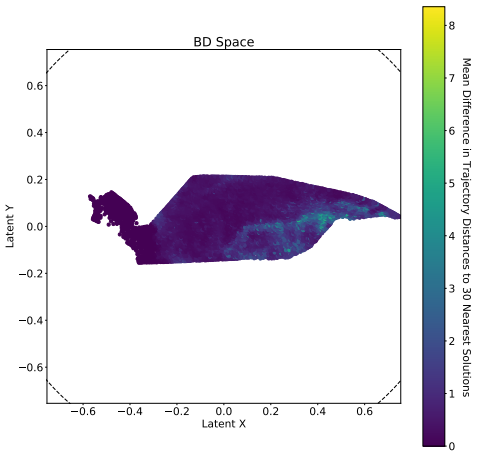


Figure 7.54: BD Space constructed by L2-KL-NST-LOG at 100% stochasticity. Colour bar indicates the mean difference in trajectory distances of each solution compared to the trajectory distances of the 30 most similar solutions.

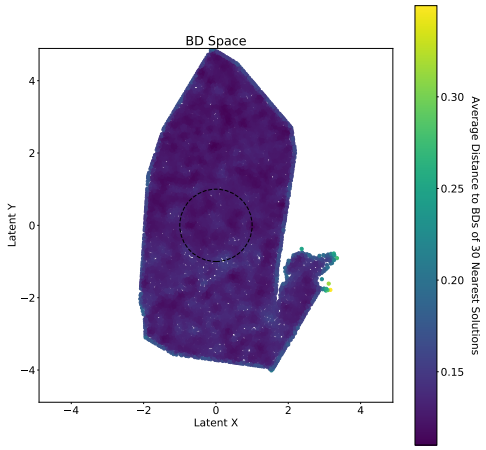


Figure 7.55: BD Space constructed by L2-SNE-NST-LOG at 100% stochasticity. Colour bar indicates a solution's average distance to the 30 most similar solutions in the BD space.

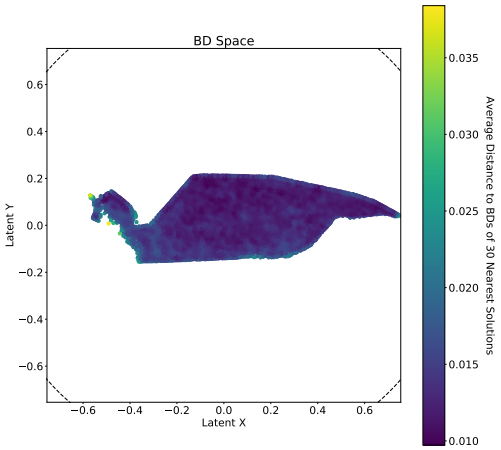


Figure 7.56: BD Space constructed by L2-KL-NST-LOG at 100% stochasticity. Colour bar indicates a solution's average distance to the 30 most similar solutions in the BD space.

Earlier in this section, we partially deferred discussions on the observations of a lower PCTMOVE score and a larger variance in the trajectory distances for the largest

stochasticity levels in Figs. 7.37 and 7.41, which we will now address.

Fig. 7.57 shows that the variance in the descriptors of no-move solutions is larger for the SNE variants than for the KL variants. In our analysis so far, large values in this metric indicated the failure to assign the same BD to all solutions generating the same single behaviour of not moving the puck. Consequently, large values in this variance metric are typically reflected in a lower PCTMOVE score. However, the SNE criterion increases the compactness of the BD space substantially which produces the large PCTMOVE scores we observed in Fig. 8.43. In cases like these, a large variance in the latent descriptors is instead an indication of a construction performance that is impacted by the noise in the environment. This results in actual no-move solutions not being recognised and constructed as such when the stochasticity is large. This in turn leads to their BDs being placed among the set of solutions that do move the puck and therefore a large variance in their descriptor positions. Fig. 7.58 visualises these scattered no-move solutions in a BD space constructed by the SNE Euclidean variant at 100% stochasticity.

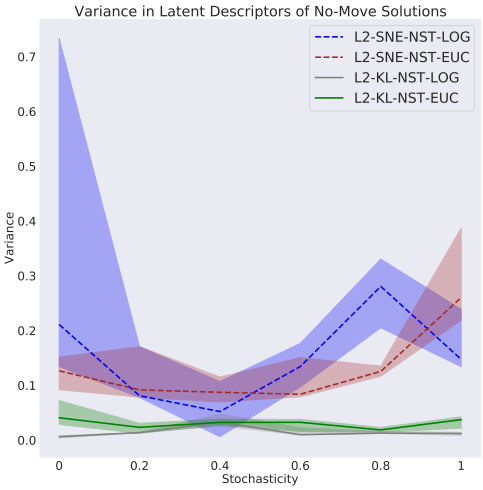


Figure 7.57: The variance in the no-move solutions' latent descriptors.

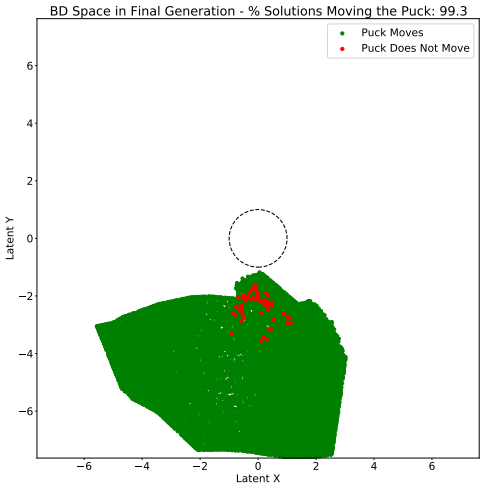


Figure 7.58: BD Space constructed by L2-SNE-NST-EUC at 100% stochasticity.

This deterioration in the construction accuracy causes the large increase in the variance of latent descriptors, the large decrease in the PCTMOVE score and the increase in the variance in the trajectory distances at 80% and 100% stochasticity observed in Figs. 7.57, 7.37 and 7.41 respectively. In the case of the latter metric, the shortest of trajectories are misconstructed and misdescribed as longer trajectories, allowing them to enter the archive. This increases the variance.

By design, the t-SNE algorithm alleviates the crowding problem by employing a Student t-distribution rather than a Gaussian distribution to calculate the similarities in the latent space. The heavier tails of this distribution allow for the representations of a given pair of similar high-dimensional observations to be placed further apart

than under a SNE criterion.

This reduced pressure leads to a nullification of the effects observed in the SNE implementation. Fig. 7.59 demonstrates that the t-SNE variants produce solutions with significantly shorter trajectory distances. At the same time, the variance in these distances in Fig. 7.60 increases relative to the SNE implementations. In both metrics, the scores produced by the t-SNE variants are similar to those of the KL variants.

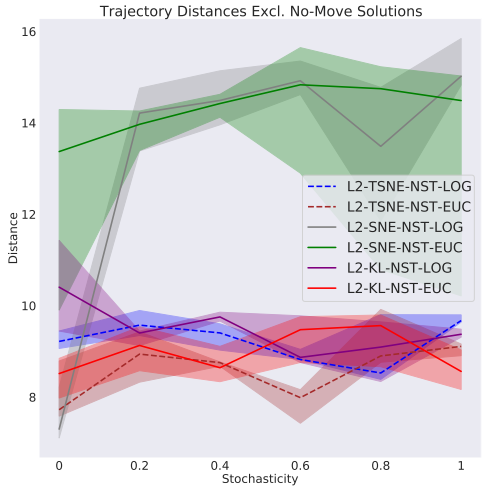


Figure 7.59: The trajectory distances of solutions that move the puck.

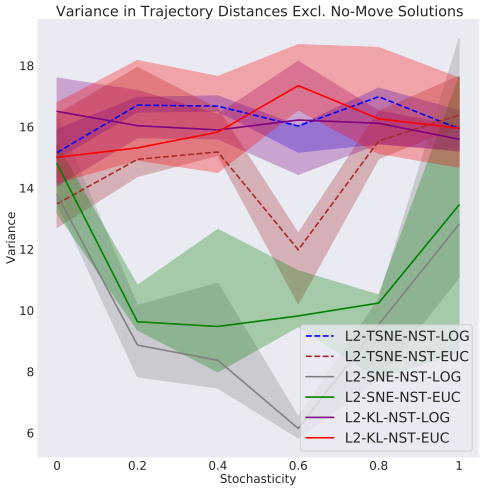


Figure 7.60: The variance in the trajectory distances of solutions that move the puck.

The large differences between the SNE and t-SNE criterions' impact on our algorithm can also be seen in the constructed BD spaces. Fig. 7.61 presents a BD space created by the L2-TSNE-NST-EUC variant. The colour-encoding captures the distance travelled by each trajectory. The proportions of short and medium-distance solutions are larger while the proportion of long solutions is smaller than in the BD space constructed by the SNE variant in Fig. 7.51. The colour ranges in both illustrations are scaled to the same size.
In Fig. 7.62, the colour-coding shows the average absolute difference in a solution's trajectory distance compared to the trajectories' of its 30 nearest neighbours in the BD space. We can see that the rate of change in the low-distance areas of the occupied space is much more gradual than in the SNE variant in Fig. 7.53.

The elimination of the crowding effect also results in a lower PCTMOVE score for the TSNE variants in Fig. 7.63. While the decrease relative to the SNE implementations is expected, the TSNE variants also produce a lower score than the KL variants. Fig. 7.64 shows that the t-SNE variants produce a significantly larger variance in the latent descriptors of no-move solutions than the KL implementations. Given the low PCTMOVE scores achieved, these large variance values are now again more likely

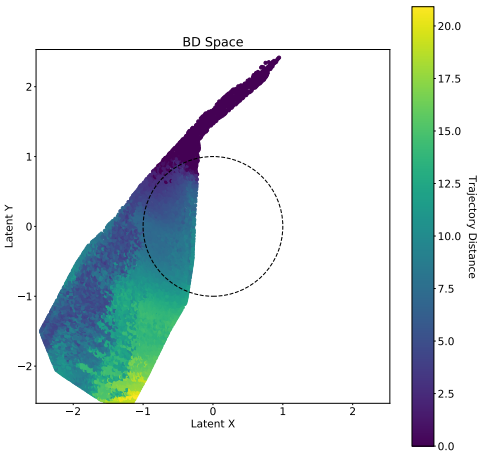


Figure 7.61: BD Space constructed by L2-TSNE-NST-EUC at 100% stochasticity. Colour bar indicates the distance travelled by the trajectory.

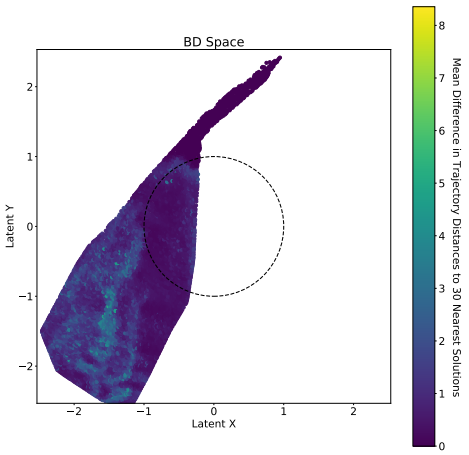


Figure 7.62: BD Space constructed by L2-TSNE-NST-EUC at 100% stochasticity. Colour bar indicates the mean difference in trajectory distances of each solution compared to the trajectory distances of the 30 most similar solutions.

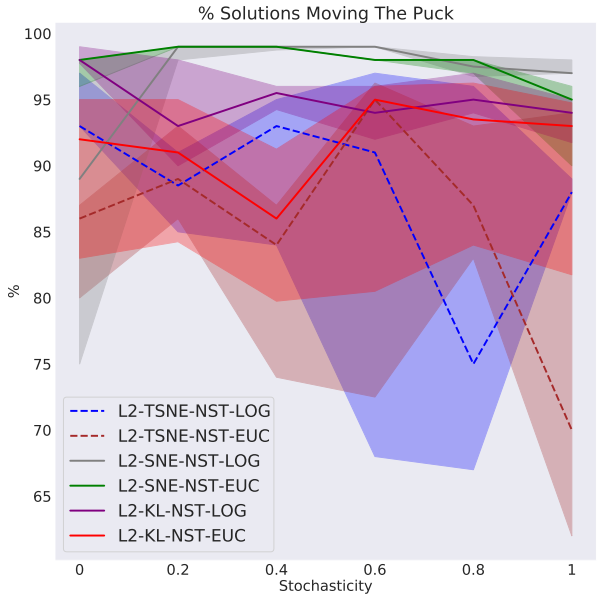


Figure 7.63: The proportion of solutions moving the puck.

to serve as a confirmation of a lack of compression of the BD space rather than a poor construction accuracy. The visualisations of the constructed BD spaces seem to

confirm this. The no-move solutions are not scattered across the BD space as we saw in the SNE implementations in Fig. 7.58, but clustered into the same area. We show a BD space constructed by the L2-TSNE-NST-LOG variant at 100% stochasticity in Fig. 7.65.

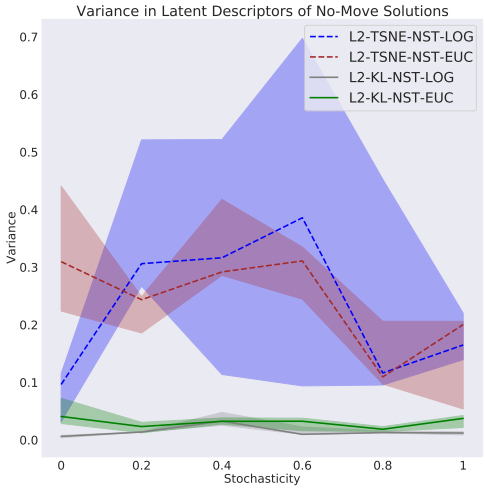


Figure 7.64: The variance in the no-move solutions' latent descriptors.

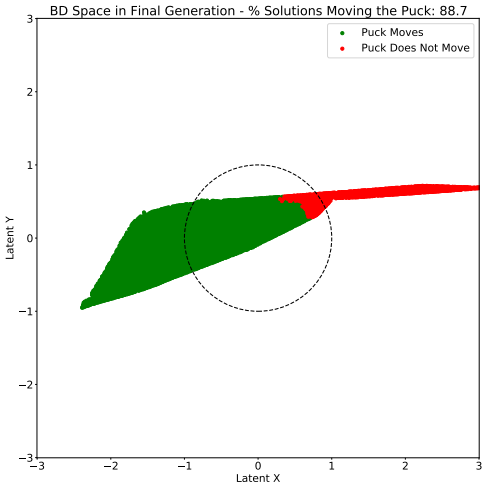


Figure 7.65: BD Space constructed by L2-TSNE-NST-LOG at 100% stochasticity.

This suggests that the driver of the TSNE variants' poor PCTMOVE scores is indeed a weak compression of similar behaviours. However, in inspecting the constructed BD spaces we also find that many produce an interesting arrangement of the no-move solutions' representations. In our KL variants, the descriptors of all solutions typically form one single large cluster. The t-SNE variants however, often arrange the no-move solutions in a separate shape protruding out of the set of solutions that do move the puck. This can be seen in the BD space shown in Fig. 7.65.
When we visualised the trajectory constructions for this set of solutions, we found that they were indeed quite similar but more importantly they were very different from a still-standing puck trajectory. We provide an example in Fig. 7.66. A poor construction performance therefore proves to be a contributing factor to the t-SNE variants' poor PCTMOVE scores after all.

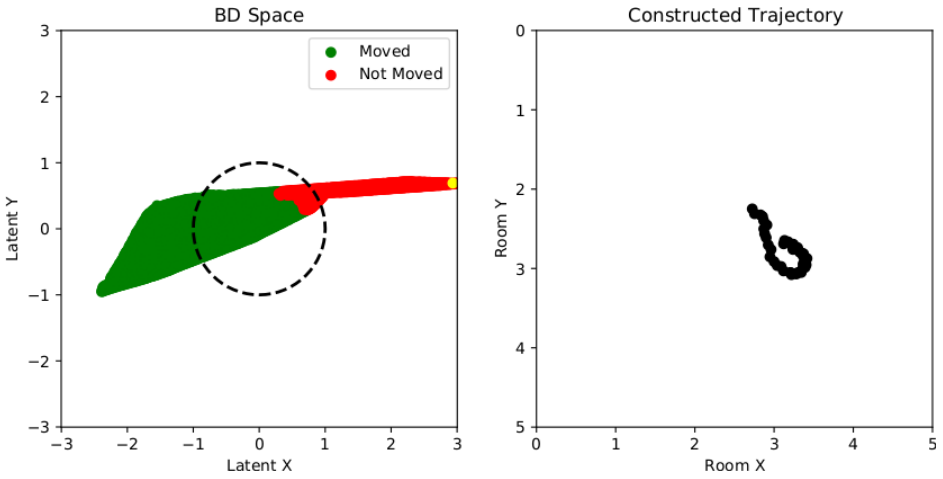


Figure 7.66: Trajectory constructions (right) associated to yellow highlighted latent codes (left). L2-TSNE-NST-LOG at 100% stochasticity.

When we add the KL divergence criterion to the t-SNE implementation, the variance in the no-move solutions' latent descriptors shrinks significantly in Fig. 7.67. This indicates an increased compression which inspections of the constructed BD spaces confirm. The L2-KL-TSNE-NST-LOG variant produces a much more compact space in Fig. 7.68 than the no-KL variant in Fig. 7.69.

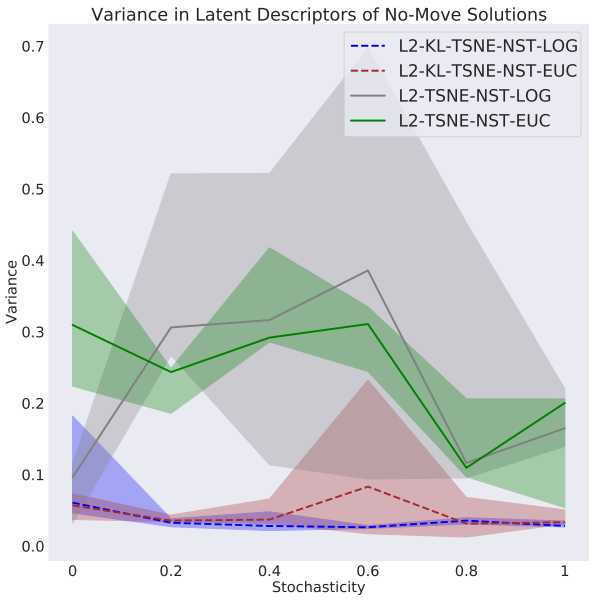


Figure 7.67: The variance in the no-move solutions' latent descriptors.

However, we also observe the same type of arrangements of the no-move solutions as shown previously in Fig. 7.65. This indicates a poor construction performance. Moreover, the addition of the KL criterion to the network's optimisation process

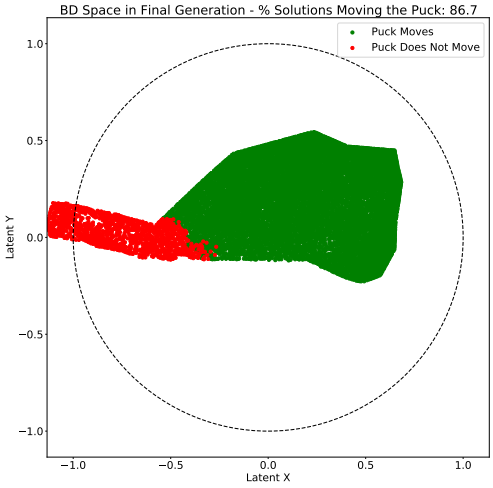


Figure 7.68: BD Space constructed by L2-KL-TSNE-NST-LOG at 100% stochasticity.

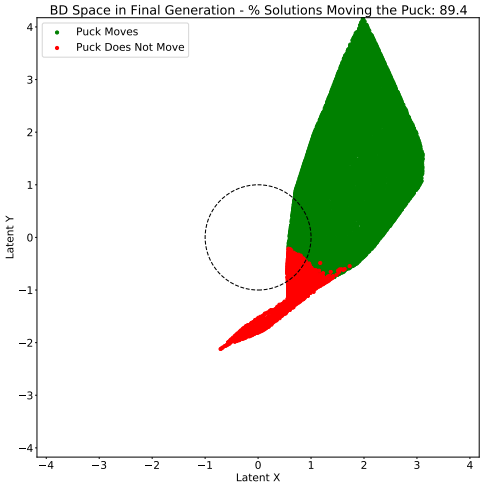


Figure 7.69: BD Space constructed by L2-KL-NST-LOG at 100% stochasticity.

results in a further impairment of the construction accuracy. These inaccuracies leave the KL-TSNE variants unable to translate an increased compression of the BD space into any performance gains. The PCTMOVE score remains largely unchanged in Fig. 7.70 and the diversity metrics are also unaffected by the addition of the KL criterion. This is shown in the diversity and POSVAR scores in Figs. 7.71 and 7.72.

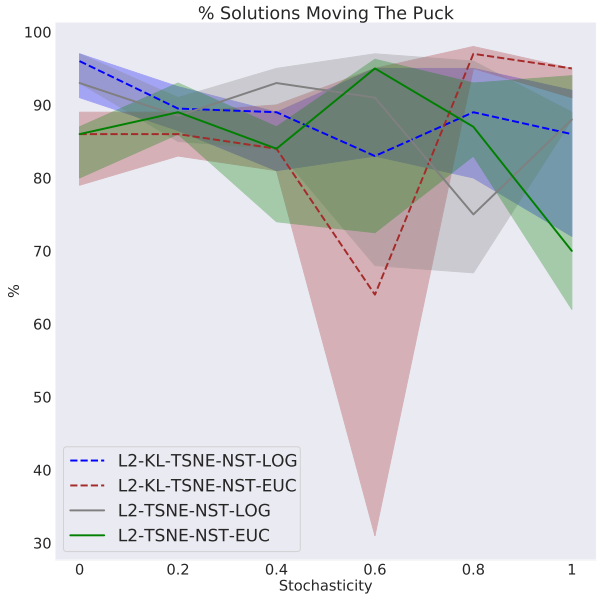


Figure 7.70: The proportion of solutions moving the puck.

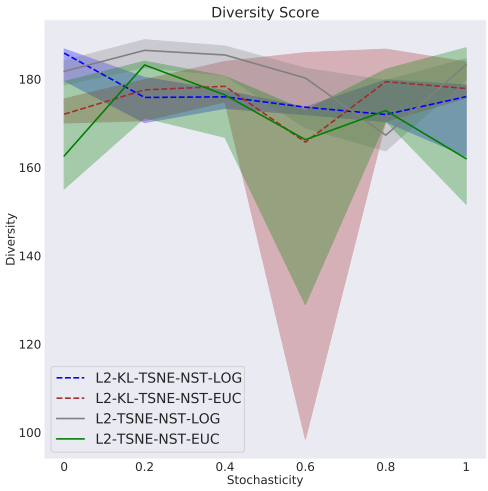


Figure 7.71: The Diversity Score.

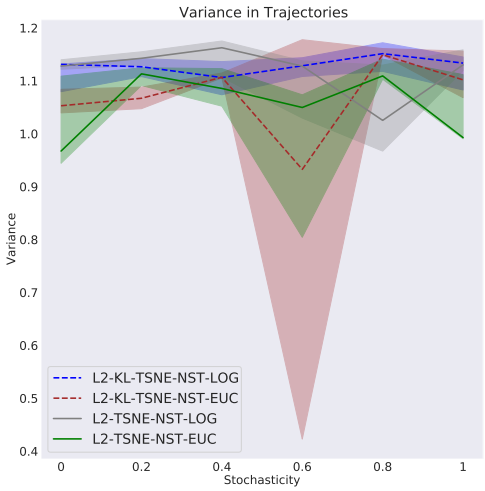


Figure 7.72: The variance in the puck trajectories.

The addition of the KL criterion to the SNE variants leaves the variances of latent descriptors in Fig. 7.73 at their previous values. The SNE variants' use of a Gaussian likelihood in the latent space produces a much stronger influence on the positions of the BDs than the Student t-distribution with its thicker tails. The SNE variants therefore do not afford the same attention to the optimisation of the KL criterion as the TSNE variants. This can be seen in a significantly larger KL loss in Fig. 7.74.

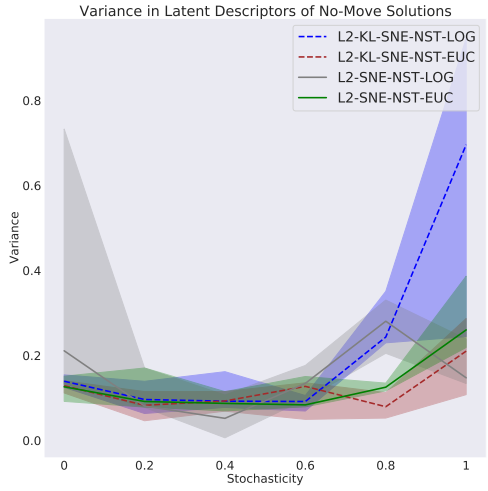


Figure 7.73: The variance in the no-move solutions' latent descriptors.

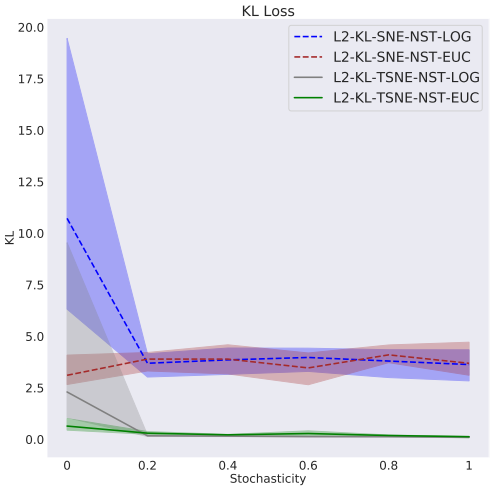


Figure 7.74: The KL Loss.

Indeed, in our BD space visualisations, we observe that the occupied spaces shift

closer to the origin as a whole but retain their sizes. We show this in two BD spaces constructed with and without the KL criterion in Figs. 7.75 and 7.76.

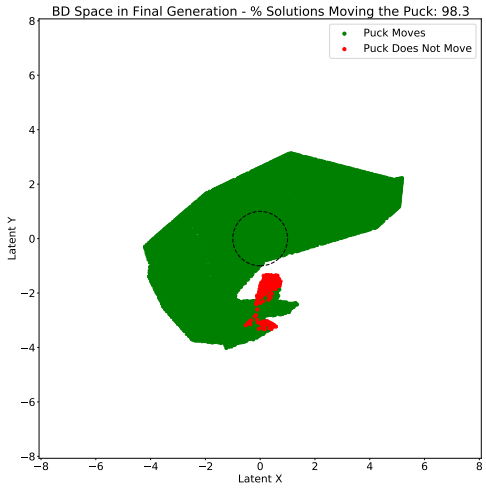


Figure 7.75: BD Space constructed by L2-KL-SNE-NST-LOG at 100% stochasticity.

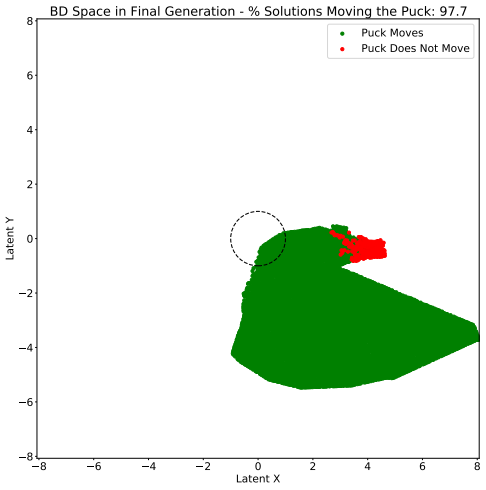


Figure 7.76: BD Space constructed by L2-SNE-NST-LOG at 100% stochasticity.

With the criterion's influence restricted to a re-positioning of the occupied BD space and no additional compression, the PCTMOVE score in Fig. 7.77 stays largely unaffected by the addition of the KL term. As a result, the POSVAR score in Fig. 7.78 is left unchanged.

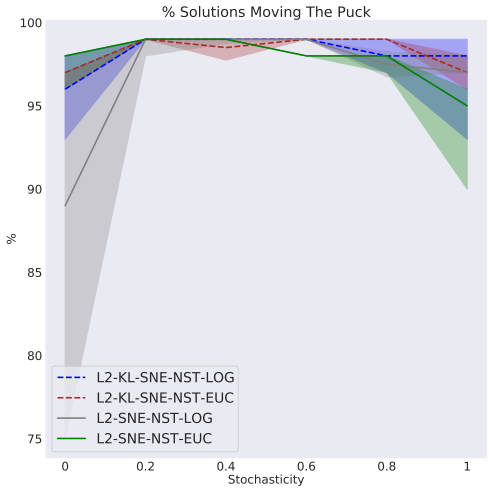


Figure 7.77: The proportion of solutions moving the puck.

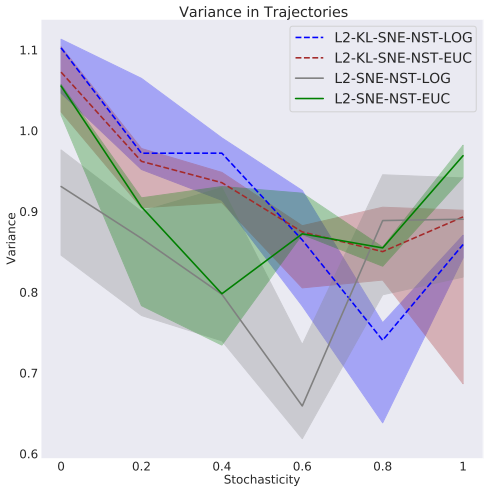


Figure 7.78: The variance in the puck trajectories.

Both the SNE and the t-SNE criterion therefore fail to improve on the diversity per-

formances of our KL variants. However, the SNE criterion produces some interesting changes to the characteristics of the produced set of solutions. In Chapter 7.2.2, we proposed the PCTMOVE score as an appropriate measure of the overall architecture's performance in discriminating the noise in observations and generating accurate BDs. By crowding the no-move solutions closer together in the latent space, the SNE variants achieve almost 100% in this metric. However, this comes at the expense of rejecting a larger proportion of solutions generating short puck trajectories for being too similar. The resulting effect on the overall diversity of the produced repertoires was quite clearly negative.

The generation of specific characteristics in the archive of solutions can be a desirable property in certain use cases. Indeed, a case could be made for longer trajectories to be considered more performing. However, we are interested in creating the most diverse archive of solutions rather than the most performing. Moreover, the SNE criterion is strongly influenced the environment noise and is likely to have vastly different effects for different environments, tasks and observation types. We will see evidence for the latter in Chapter 8.3.5.
The KL divergence implementations are therefore the more appropriate choice for creating a diverse set of solutions in the presence of stochasticity. Furthermore, they are also more likely to deliver consistent results when we change environments or observation types. This robustness is one of the defining features of the AURORA algorithm and its preservation is an important result.

We conclude our discussion on the SNE variants, with two final observations. First, we note that the SNE variants produce a BD space that frequently changes in shape over the generations. We present an example in Fig. 7.79. Conversely, all our other variants retain the shape of the space created after 500 generations for the remainder of the algorithm. We show some examples of this in Chapter 7.3.7. A frequently changing shape in the SNE variants' constructed BD spaces is likely caused by the combination of two factors. First, the use of the VAE's constructions as the high-dimensional observations. Changes in the network's construction accuracy directly affect the SNE similarity calculations and lead to a reshuffling of the descriptors. While we also use the network's constructions in the t-SNE calculations, we do not observe such a regularly changing latent space in any of the t-SNE BD spaces we inspected. This leads us to the second factor, which is the SNE criterion's dominance in the network's optimisation. We have seen from the comparison of the SNE and t-SNE variants, that the use of a Gaussian distribution in the low-dimensional latent space results in the SNE criterion exerting a much larger influence on the network's optimisation decisions than the t-SNE criterion.

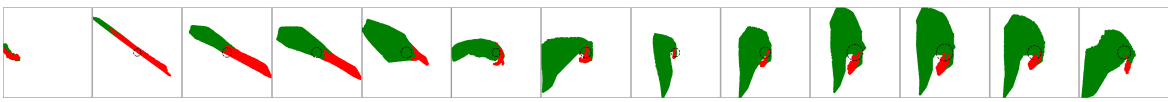


Figure 7.79: Progression of BD space constructed by L2-SNE-NST-LOG over 6000 generations. Snapshots taken in steps of 500 generations.

The second note we make is the observation of the SNE variants' tendency to construct a curved BD space. The solutions for which the network produces short trajectory constructions often form a small cluster that is shaped as if it was bent towards the main cluster of solutions. This is clearly visible in Figs. 7.51, 7.75, 7.76 and in the progression of the BD space shown in this section in Fig. 7.79. On the contrary, the KL and t-SNE variants produce BD spaces that are aligned in a straight line without any indication of the shorter solutions being curved inwards towards the core of the occupied space. This is well visible in all examples shown previously. We highlight the latent space constructed by the t-SNE variant in Fig. 7.61 due to the colour-coding of the distances.
In Chapter 7.3.1 we found that even without a KL criterion the VAE chooses the length and the shape of the trajectory as the latent generative factors according to which it arranges the latent space. The SNE variant also performs its arrangement according to these two attributes. The solutions with the shortest trajectories are placed at one end of the occupied latent space and the longest trajectories at the other. We visualised this in Fig. 7.51 with a colour-encoding of the distances. Using trajectory visualisations, we confirmed that the other dimensionality indeed captures the shape of the trajectory.
However, this arrangement according to the trajectories' shapes leaves the SNE variants unable to form the BD space in a straight line, as a subset of the shorter trajectories is always very similar in shape to some of the longer trajectories. The large similarities between these trajectories pull their latent descriptors closer, leading to a curved shape overall and a sharp bend at the end of the occupied space that stores the shortest solutions. In Fig. 7.80, we provide a visualisation of two constructions and their latent descriptors. The first trajectory has its BD placed in the core of the occupied space due to its moderate distance. The other trajectory is significantly shorter but of similar shape and therefore placed in the small cluster of solutions bending sharply towards the main block.

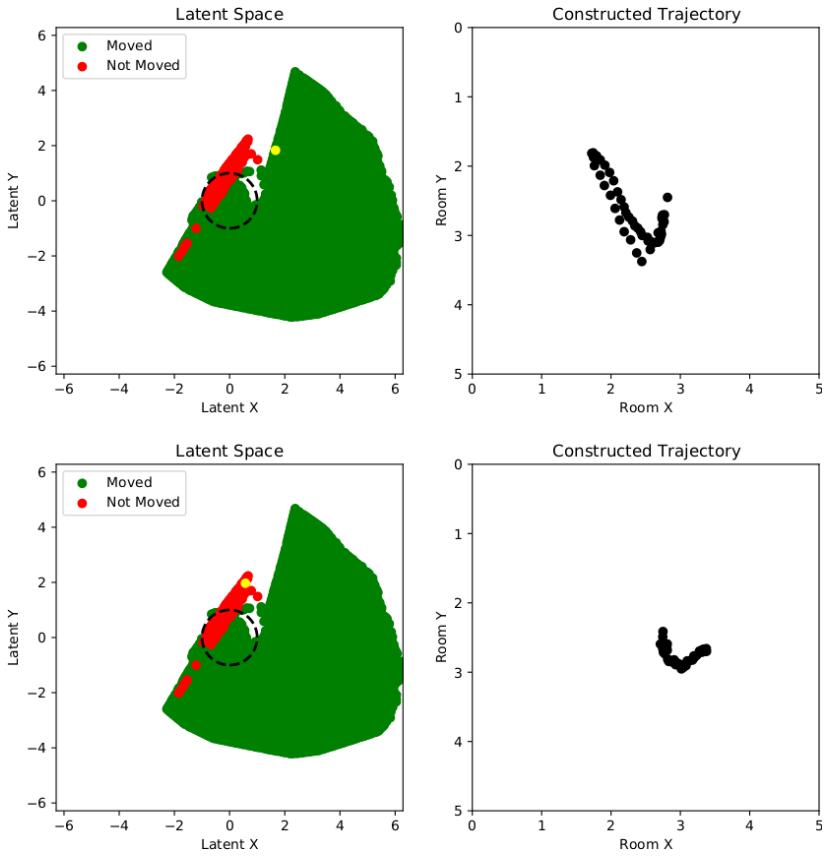


Figure 7.80: Trajectory constructions (right) associated to yellow highlighted latent codes (left). L2-SNE-NST-LOG at 80% stochasticity.

7.3.5 Log-Likelihood Loss

When noisy samples enter the dataset, the network is forced to shift its constructions away from the actual trajectories and closer towards a mean of all random trajectories to decrease the overall losses incurred. In Chapter 3, we proposed an improvement to the VAE's noise discrimination ability by using a Log-Likelihood loss. We hypothesised, that the presence of the Decoder variance term provides an additional channel for the network to compensate for the noisy trajectories. By increasing the variance, it can reduce the construction loss artificially when faced with noisy labels and therefore decrease the pressure to shift its construction outputs closer to the mean random trajectory.

To isolate the impact of the Decoder variance term, we first consider the L2-NST variants in which we do not employ a KL criterion. We plot the Actual L2 Errors in Fig. 7.81. The Log-Likelihood variant suffers a large spike in the error when the environment is noise-free. This has been a common feature of all the loss plots we have seen in our analysis so far.

In Eq. 7.1 we rewrite the construction log-likelihood we derived in Eq. 2.2. c_i denotes the Decoder output of the mean, o_i the corresponding observation and l_i the Decoder

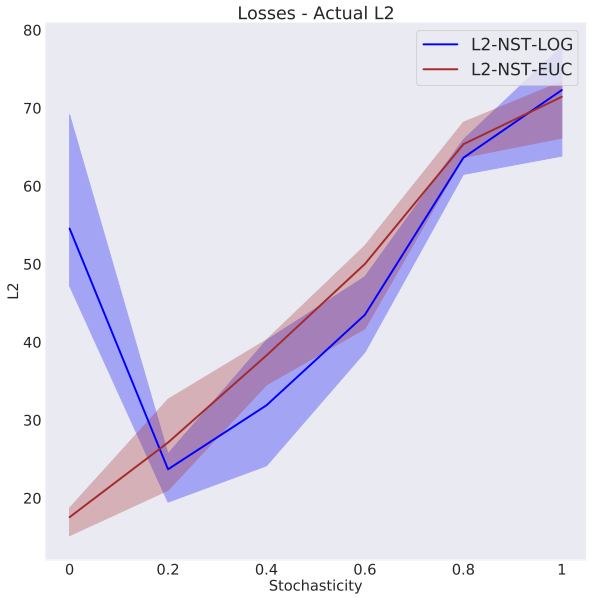


Figure 7.81: The Actual L2 Error.

log-variance (logvar) output.

$$L_i = \frac{||c_i - o_i||^2}{2 * \exp(l_i)} + \frac{1}{2}(l_i + \log(2\pi)) = \frac{L2}{2 * \exp(l_i)} + \frac{1}{2}(l_i + \log(2\pi)) \tag{7.1}$$

The gradients with respect to the Decoder log-variance and mean are given in Eq. 7.2 and 7.3 respectively.

$$\frac{\partial L_i}{\partial l_i} = -\frac{L2}{2 * \exp(l_i)} + \frac{1}{2} \tag{7.2}$$

$$\frac{\partial L_i}{\partial c_i} = \frac{\nabla_{c_i} L2}{2 * \exp(l_i)} \tag{7.3}$$

In Eq. 7.2, the first term is negative and therefore pulls the log-variance output towards larger positive values. The magnitude of this term increases as the logvar decreases. This term is balanced by the second constant additive term, which pulls the gradient in the opposite direction.

Next, we need to consider the characteristics of the dataset in the initial training iterations. When we reach the first training iteration after 10 generations of the algorithm, the archive contains a maximum of 2560 solutions since at most 256 solutions can be added to the archive in each generation. The size of the training set is smaller than the archive as we reserve 20% of the solutions for a validation set. Furthermore, a large proportion in the training set is likely made up of no-move solutions given that we initialise our first set of solutions by randomly sampling in the

solution space. These solutions are all associated with the exact same observation of repeating values of the coordinates of the resting puck.
The combination of these factors produces a small and homogeneous training dataset when we commence the first training iteration. This leads to a fast reduction in the L2 error. As the error decreases in magnitude, the logvar gradient in Eq. 7.2 increases. This in turn produces a rapidly decreasing logvar value. Fig. 7.82 shows the mean L2 error and logvar outputs for each epoch in the first training iteration. At the end of this iteration, the L2 error is minimal and the Decoder logvar is negative. The latter continues to decrease over the subsequent training iterations.

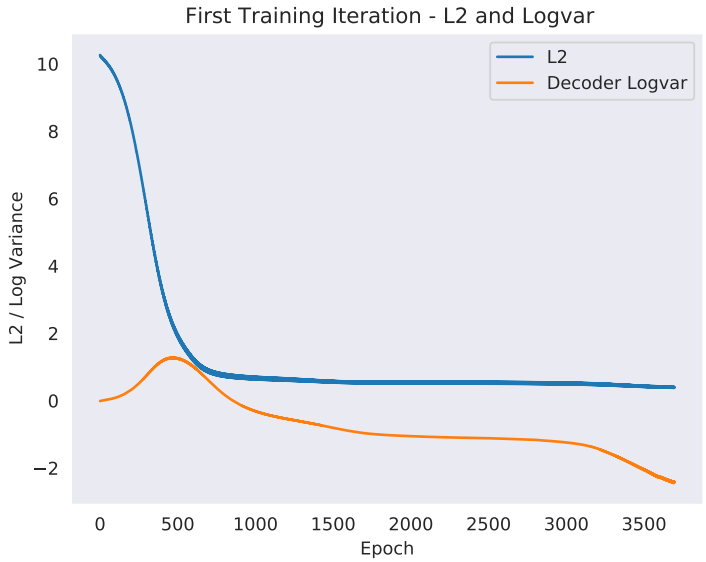


Figure 7.82: Mean L2 Loss and Decoder logvar outputs for each epoch in the first training iteration of the algorithm. L2-KL-NST-LOG at 0% stochasticity.

By design we add only novel solutions to the archive. However, their novelty implies that the network has likely not seen any similar trajectories before. An increasing number of samples will start incurring large L2 losses. With the Decoder logvar outputs at large negative values, these large losses lead to an abrupt explosion in the magnitude of the first term in the logvar gradient in Eq. 7.2. The result is a very rapid increase in the Decoder logvar outputs when these novel individuals are added to the archive in increasingly larger numbers.

We illustrate this in Fig. 7.83, in which we plot the median L2 error and logvar outputs over all samples in the dataset at the end of each generation. We use the median rather than the mean because it more adequately captures the data points of the novel samples that we are interested in since the distribution in both metrics is highly positively skewed. However, the use of the median still does not capture the most problematic samples in the tail of the distributions which incur the largest L2 errors while having the smallest logvar values.
The abrupt increases and decreases in Fig. 7.83 mark the training iterations occurring every 10 generations. The logvar values continuously decrease in the initial 100

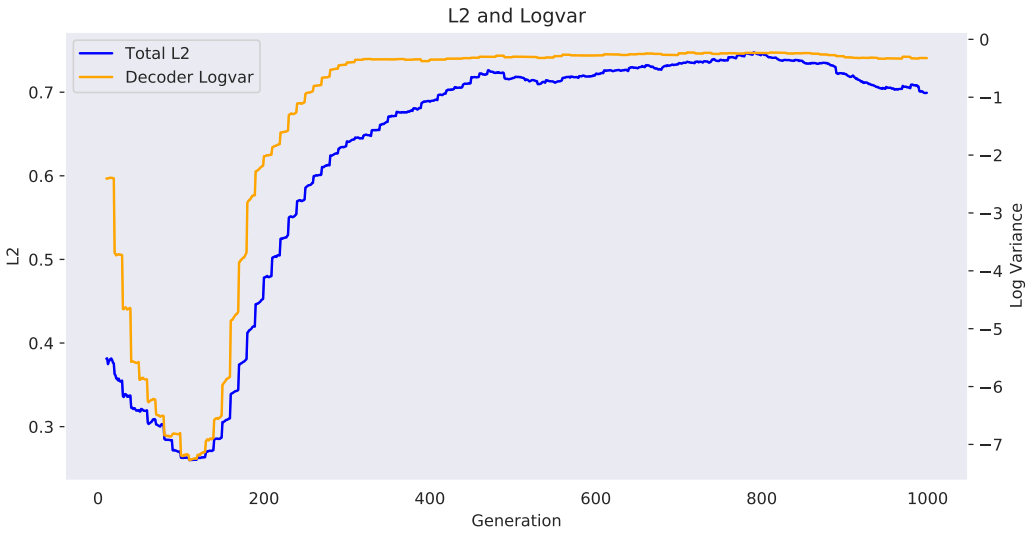


Figure 7.83: Median L2 error and Decoder logvar values recorded at the end of each generation. L2-KL-NST-LOG at 0% stochasticity.

generations as the L2 error is shrinking to lower values. This is followed by abrupt increases as the L2 error starts rising in the subsequent 200 generations. As the logvar values grow, the magnitude of the first term in the logvar gradient in Eq. 7.2 shrinks. This leads to a deceleration in the increase.

For the novel samples with the largest L2 errors, these large logvar values then result in a significantly smaller construction gradient magnitude in Eq. 7.3. A poor construction accuracy for these samples can therefore be improved only very slowly and their L2 errors remain large throughout the remaining generations.

This destabilising sequence of variance updates occurs only within the first 300 generations in Fig. 7.83. This is because the majority of all solutions we retain in the archive are generated in these initial generations. The diversity score in Fig. 7.84 shows that after 500 generations only very few novel solutions are added to the archive. With fewer changes in the training dataset, new sudden increases in the L2 errors occur less frequently. The Decoder variance can stabilise and the network can start its slow reduction of the L2 error.

When we use an L1 loss, the large losses incurred initially on novel samples are no longer squared. This leads to a smaller explosion in the first term of the logvar gradient in Eq. 7.2. We therefore observe the L1 variants incurring a relatively smaller loss at 0% stochasticity in Fig. 7.90 in Chapter 7.3.6.

Finally, this effect also loses its intensity when the environment is stochastic. In the very first training iteration, noisy labels are already present in the dataset. The L2 loss can therefore not be reduced to the same low levels as in a noise-free environ-

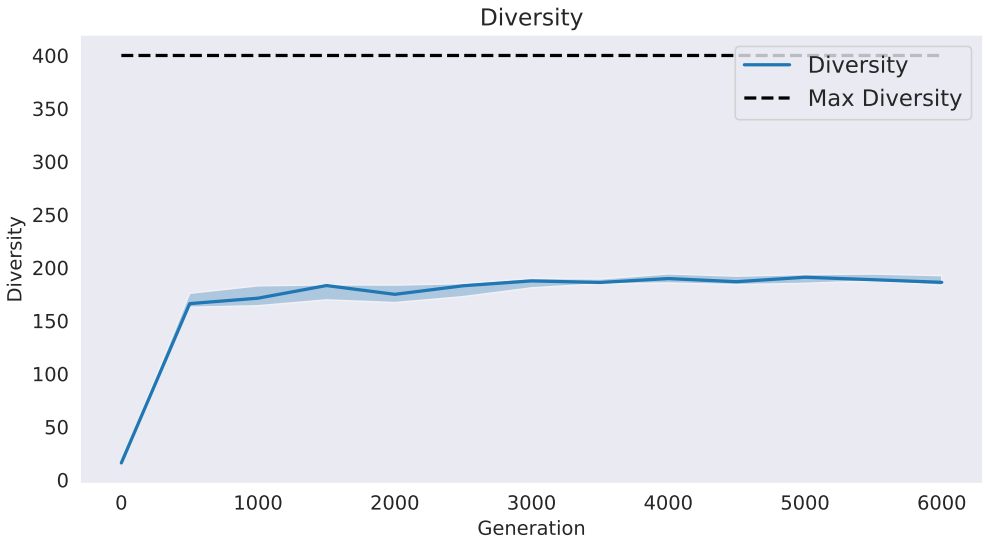


Figure 7.84: The Diversity Score over 6000 generations. L2-KL-NST-LOG at 0% stochasticity.

ment. Consequently, the Decoder logvar outputs remain at a larger magnitude and novel samples lead to a smaller positive shock in the logvar gradient.

We have not found this issue documented in any resources we searched. This is likely due to the high specificity of circumstances required which happen to be created in our algorithm. In most applications of a VAE, the dataset does not change during training. This makes it rather unlikely for construction losses to take very small values initially and explode subsequently.

Returning to our comparison of the Euclidean and Log-Likelihood variant, we observe that the latter achieves a marginally lower Actual L2 error in Fig. 7.81 than the Euclidean variant when the environment is noisy. However, this difference does not appear statistically significant when we consider the large interquartile ranges. The POSVAR score in Fig. 7.85 even indicates a marginally reduced diversity performance in the Log-Likelihood variants.

When we add the KL divergence criterion, we obtain more interesting performance differences. While the L2 construction performance in Fig. 7.86 remains largely unchanged, the Log-Likelihood variant achieves a lower KL divergence loss in Fig. 7.87. When the L2 errors are similar, the addition of a Decoder variance term can reduce the magnitude of the construction gradient. This shifts some of the network's attention in its optimisation from the construction criterion to the KL criterion. By reducing the KL loss further, the Log-Likelihood variant benefits from an increased compactness in the BD spaces. This results in larger PCTMOVE and diversity scores in Figs. 7.88 and 7.89. We therefore conclude that, contrary to our hypothesis, the use of a Log-Likelihood loss does not produce a significantly improved construction

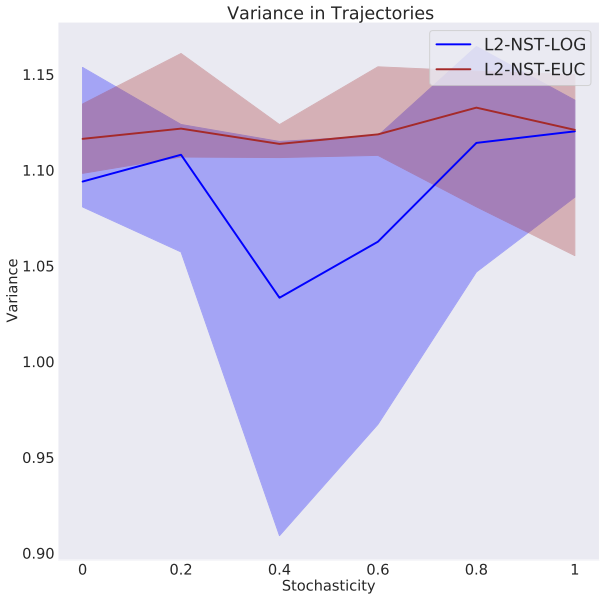


Figure 7.85: The variance in the puck trajectories.

performance in a stochastic environment. However, it does allow the VAE to reduce its KL divergence loss further. The increased BD space compactness that follows results in an improved diversity performance.

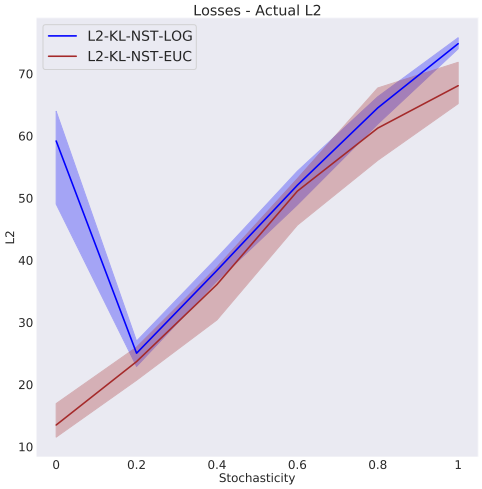


Figure 7.86: The Actual L2 Error.

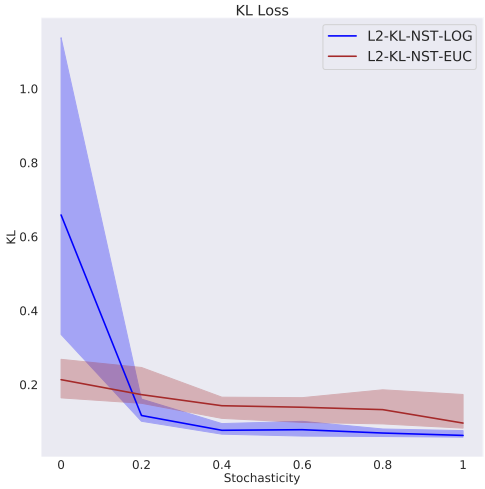


Figure 7.87: The KL Loss.

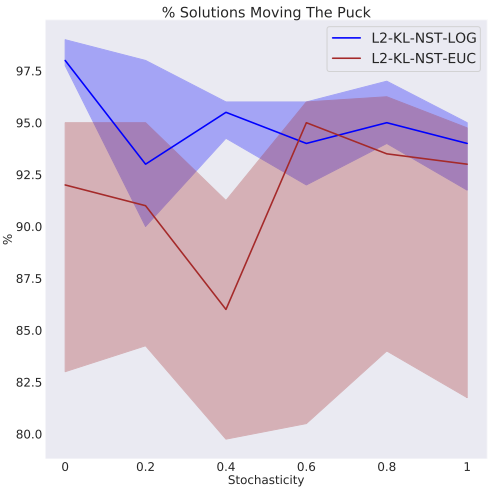


Figure 7.88: The proportion of solutions moving the puck.

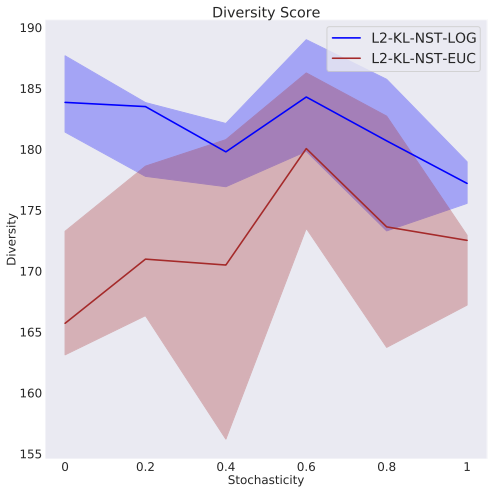


Figure 7.89: The Diversity Score.

7.3.6 Distance Measure in Loss

In Chapter 3, we discussed the disadvantages of the squared term in the L2 criterion when outliers pollute the dataset. We suggested a potential improvement to the construction performance could be achieved by replacing the L2 with an L1 criterion.

In Figs. 7.90 and 7.91, we present the Actual L2 and Total L2 errors incurred by the L1-NST and L2-NST variants. The L1 variants produce significantly lower Actual L2

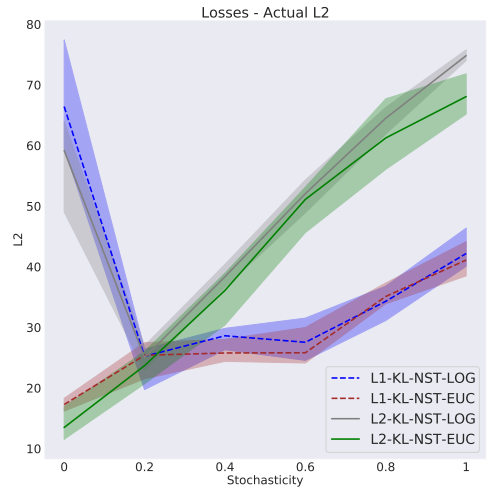


Figure 7.90: The Actual L2 Error.

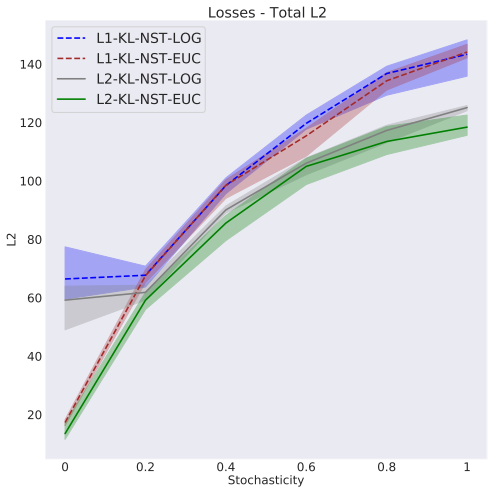


Figure 7.91: The Total L2 Error.

errors when the environment is stochastic. This performance gap diverges rapidly

as the noise levels increase. Simultaneously, they produce a larger Total L2 error. This metric includes the losses incurred on all noisy trajectories. This confirms that the construction outputs have shifted away from the mean of random trajectories towards the actual trajectories.

At 0% stochasticity, the L2 criterion provides a stronger learning signal than the L1 criterion due to a larger gradient magnitude. This leads to a faster convergence, and, in the absence of misleading labels, a superior construction accuracy. Conversely, the L1 metric's smaller gradient magnitude should allow the network to shift its optimisation focus from the construction criterion to the KL criterion. However, we observe similar KL losses for all our variants in Fig. 7.92. This suggests, that the L1 metric's smaller gradient is balanced by the larger total loss in Fig. 7.93.

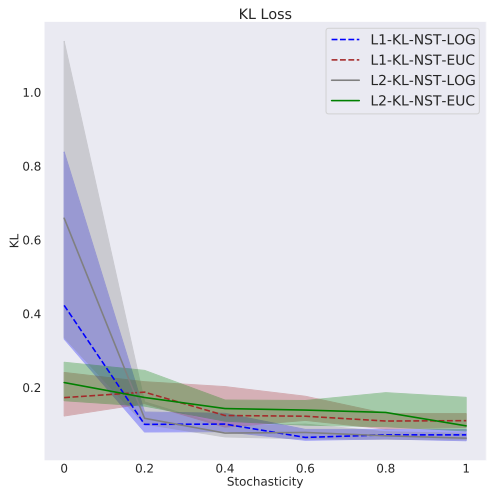


Figure 7.92: The KL Loss.

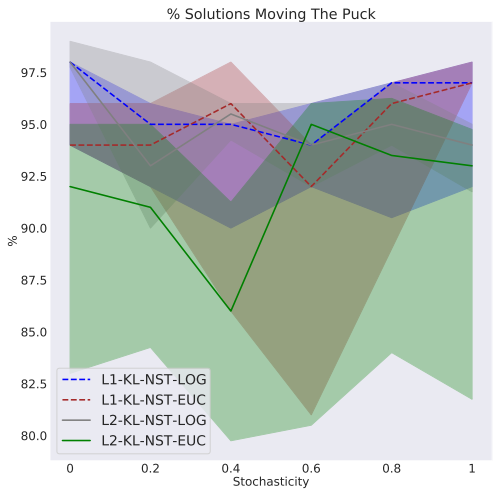


Figure 7.93: The proportion of solutions moving the puck.

The L1 variants' large gains in construction accuracy also fail to materialise in any significant improvements to our performance metrics. With no significant changes in the KL loss, the constructed BD spaces remain unaffected. The PCTMOVE scores in Fig. 7.93 are largely unchanged, as are the diversity and POSVAR scores in Figs. 7.94 and 7.95.

These results suggest that the L2 variants already produce a construction accuracy and noise discrimination that is sufficiently strong for the algorithm to deliver a robust diversity performance. A compression of the BD space is therefore the main driver in improving the diversity further. Nonetheless, the improvements to the construction accuracy are still of significant magnitudes and they make the L1-KL-NST-LOG variant the best performing architecture in this set of experiments.

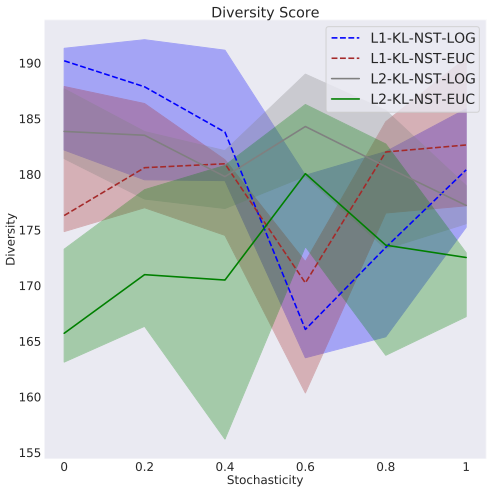


Figure 7.94: The Diversity Score.

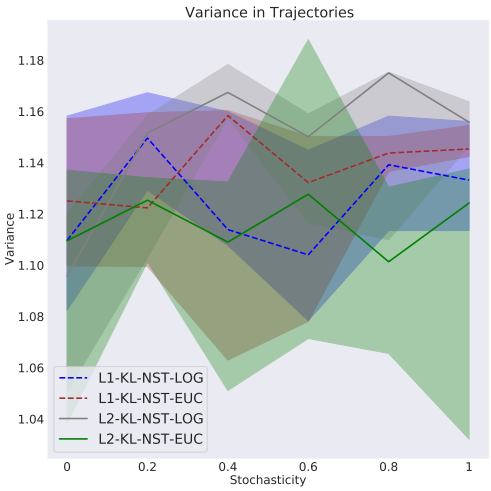


Figure 7.95: The variance in the puck trajectories.

We also investigate the use of a Smooth L1 criterion in an attempt to exploit the strengths of the L1 and L2 criterions while mitigating their weaknesses. The Smooth L1 loss calculates a squared term if the absolute element-wise error falls below 1 and an L1 term otherwise.

In the following figures, we present only the better performing Log-Likelihood variants to keep the plots legible. The conclusions we make also apply to the Euclidean variants.

Fig. 7.96 presents the Actual L2 errors. In the absence of any noise, the Smooth L1 criterion behaves similar to the L2 metric as construction errors can be reduced to fall within the unit error band. Their performances start diverging as the noise levels increase. The construction errors increase at a faster rate for the Smooth L1 variant than for the L1 implementation. A larger number of random trajectories falls within the unit error band as the stochasticity increases. This magnifies the L2 criterion's influence which yields a worse construction performance compared to the L1 variants.

The Smooth L1 variants produce very volatile median scores and large interquartile ranges in all other performance metrics. Figs. 7.97 and 7.98 demonstrate this in the variants' PCTMOVE and diversity scores that fail to match our previously obtained results.

The volatility indicates that the use of the unit error threshold value introduces a large sensitivity to the specific characteristics of the dataset. The volatility could likely be reduced by tweaking this value. However, this adds another hyperparameter which in itself would prove very sensitive to the specifics of the environment and

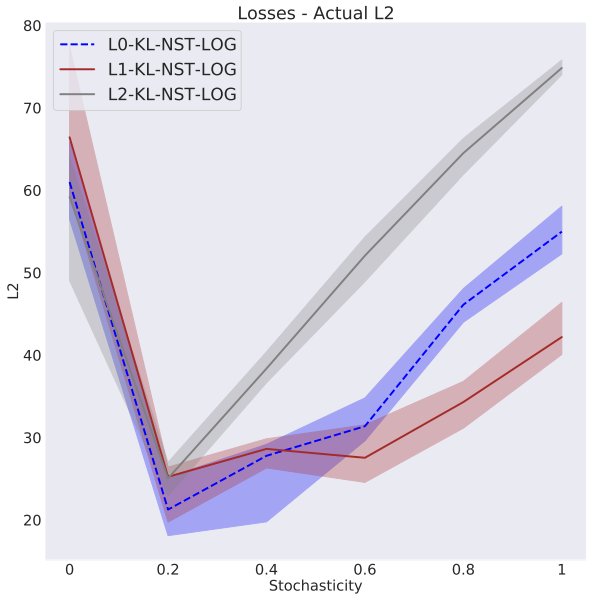


Figure 7.96: The Actual L2 Error.

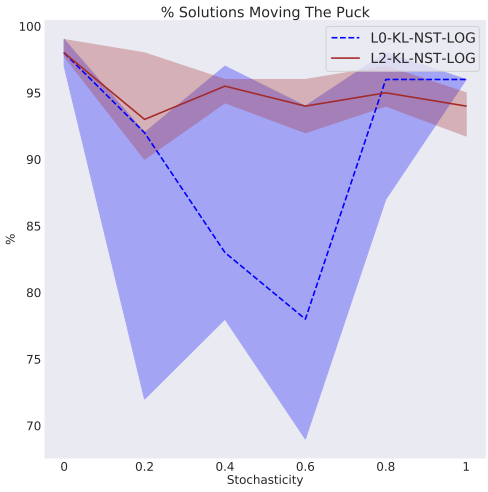


Figure 7.97: The proportion of solutions moving the puck.

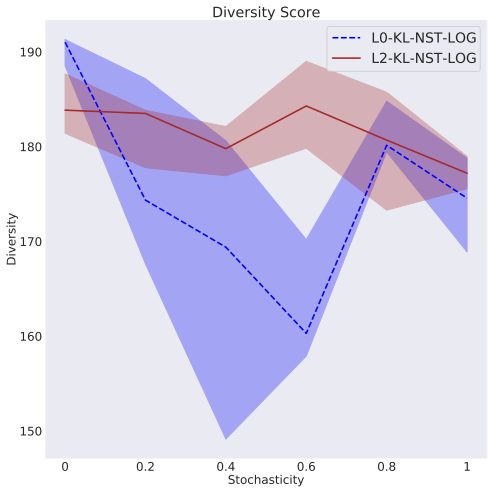


Figure 7.98: The Diversity Score.

the observations. This inconsistency in the performance makes the use of a Smooth L1 criterion rather unappealing in our search for robustness.

7.3.7 Analysing the Best Architecture

The L1-KL-NST-LOG variant produces strong scores in all our diversity metrics already at 500 generations. The performance improves further until it stabilises at 1000 generations. This overall progression can be observed at all stochasticity levels. We present the development of the POSVAR and PCTMOVE metrics in Figs. 7.99 and 7.100 at 0% and 100% stochasticity respectively. Fig. 7.99 shows an increased volatility in the performance due the presence of the Decoder log-variance.

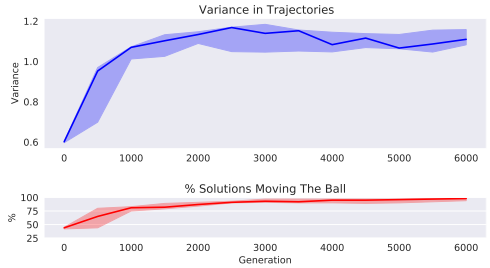


Figure 7.99: Top: POSVAR score. Bottom: PCTMOVE score. L1-KL-NST-LOG at 0% stochasticity.

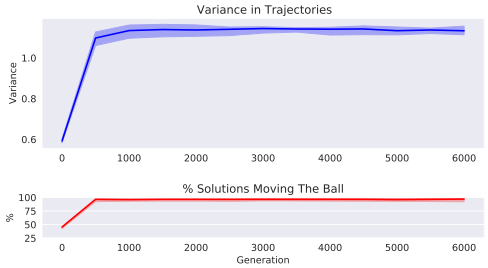


Figure 7.100: Top: POSVAR score. Bottom: PCTMOVE score. L1-KL-NST-LOG at 100% stochasticity.

We show the losses achieved over the algorithm generations at 100% stochasticity in Fig. 7.101.

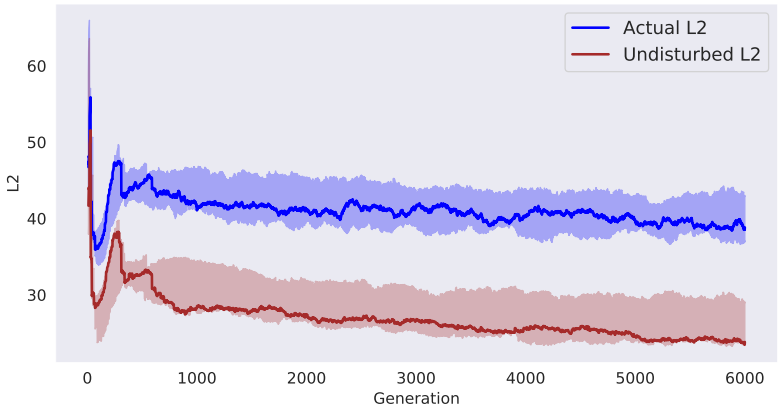


Figure 7.101: Actual and Undisturbed L2 Error. L1-KL-NST-LOG at 100% stochasticity.

To calculate the Undisturbed L2, we regenerate all trajectory observations in a noise-free environment. The difference in the Actual and Undisturbed L2 construction losses arises from collisions between the two pucks. These collisions lead to changes in the trajectory of the actual puck that the network largely ignores in the constructions due to their irregularity. The Undisturbed L2 error is therefore substantially lower. This is an additional advantage of creating a construction of the observation rather than a reconstruction. Even for relatively infrequent occurring collisions, its

direct conditioning on the observation makes the AURORA algorithm more susceptible to capturing collisions with the random puck in the BD.

The stability observed in the performance metrics is also visible in the constructed BD spaces which assume their final shape after 500 generations. We show an example of the progression observed at 100% stochasticity in Fig. 7.102.

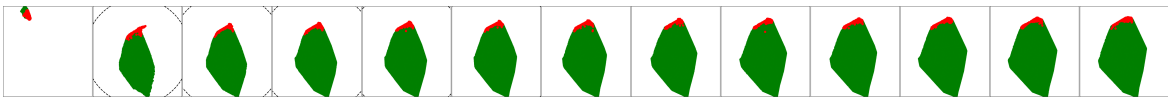


Figure 7.102: Progression of BD space constructed by L1-KL-NST-LOG over 6000 generations. Snapshots taken in steps of 500 generations at 100% stochasticity.

Fig. 7.103 presents a visualisation of all trajectories in the archives generated at 100% stochasticity. The L1-KL-NST-LOG variant produces a similar archives across 4 repetitions, demonstrating the repeatability of its achieved performance.

In our final piece of analysis in this section, we investigate whether our proposed architecture produces an archive of solutions that is suitable for the application of the Quality-Diversity selection, cross-over and mutation operators we introduced in Chapter 4. A successful integration of these operators and the automatic BD generation mechanism is vitally important for any AURORA-based architecture to be effective.

We use the L1-KL-NST-LOG variant to create our RANDOMGENERATION architecture. In this implementation, we ignore the solutions stored in the archive and instead generate each generation's set of offspring solutions by randomly sampling the solution parameters.

The L1-KL-NST-LOG variant produces a lower number of no-move solutions and a larger diversity score when we do utilise the QD operators. We show these metrics in Figs. 7.104 and 7.105. This confirms the successful integration of our proposed architecture into the AURORA framework.

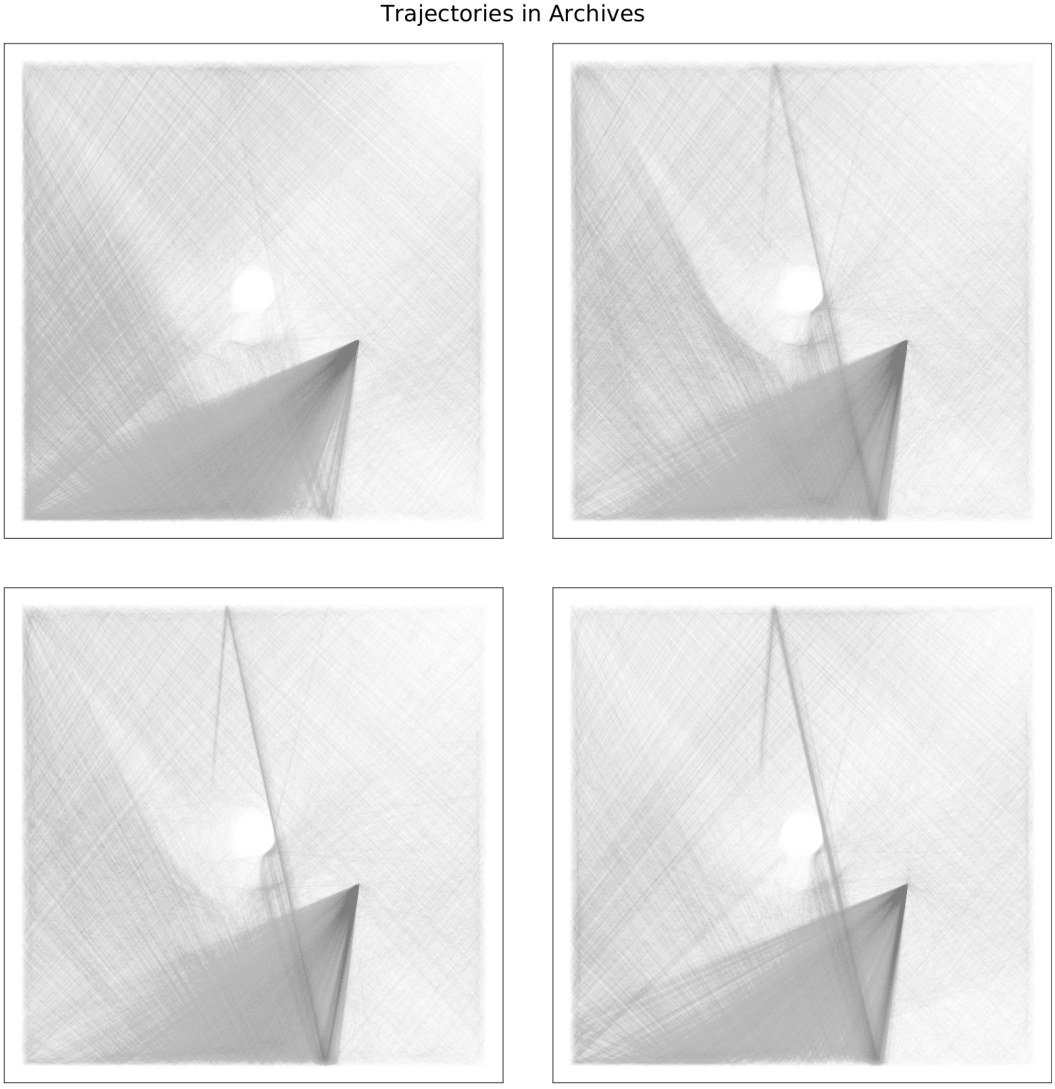


Figure 7.103: Visualisation of the actual trajectories of all solutions in 4 archives generated by L1-KL-NST-EUC at 100% stochasticity.

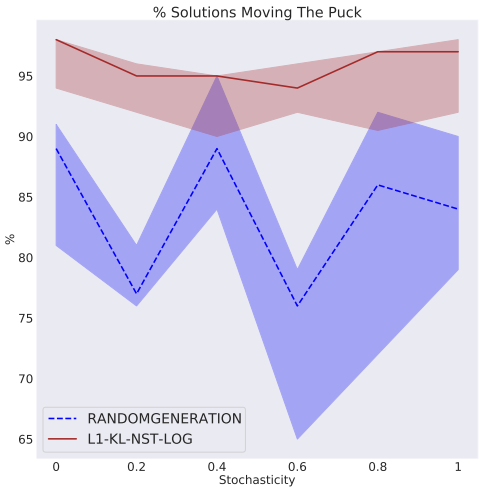


Figure 7.104: The proportion of solutions moving the puck.

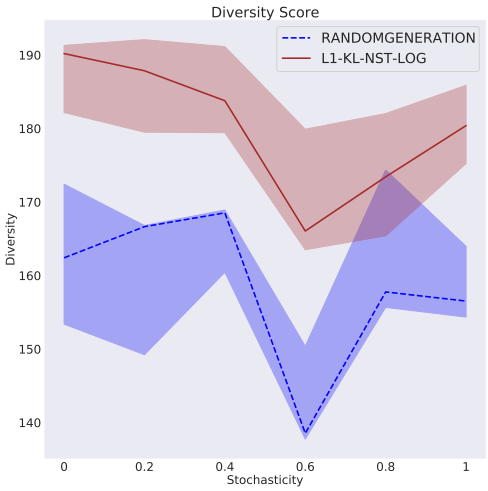


Figure 7.105: The Diversity Score.

7.3.8 Evaluation against Benchmarks and Baselines

In benchmarks EXCLUDE_TRAIN and EXCLUDE_ARCHIVE, we attempt to eliminate the impact of environment noise on the AURORA algorithm by removing any individuals for which we suspect the observations to contain noisy puck trajectories. We exploit the "memorisation" effect discussed in Chapter 2.4, by designating the solutions with the 30% largest L2 reconstruction errors as having been influenced by noise.

In EXCLUDE_TRAIN these individuals are removed from the training set. The random trajectories are likely to differ substantially in appearance from the actual puck trajectories. By removing a large proportion of the random observations, the network will therefore lack the required number of samples to learn to reconstruct these noisy trajectories accurately. This could force the network to produce an average reconstruction for all noisy trajectories to cut its losses. The produced BD would therefore be similar for all random trajectories. With the noise affecting all solutions in the same way, its impact would be reduced.
Quite clearly, this effect is based on many assumptions which are likely to hold only in a very limited number of cases. In our EXCLUDE_ARCHIVE benchmark we propose the alternative approach to remove these noisy solutions from our archive altogether.

In Fig. 7.106 we present the PCTMOVE score for these two adaptations of the AURORA implementation and L2-AURORA.

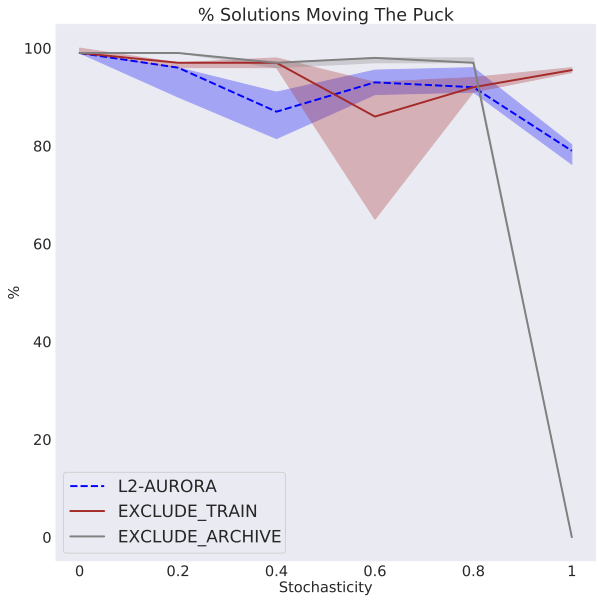


Figure 7.106: The proportion of solutions moving the puck.

The EXCLUDE_TRAIN benchmark performs quite similarly to the original implementation. More interestingly, the EXCLUDE_ARCHIVE benchmark delivers a strong and

consistent score until its performance collapses. Indeed, Fig. 7.106 indicates that at 100% stochasticity the archive is filled exclusively with solutions that do not move the puck. The AURORA algorithm usually does not allow the repeated acceptance of no-move solutions into the archive. However, at 100% stochasticity the random puck is present at all times. Each solution's BD is formed of the average latent representation of both pucks' trajectories. This allows no-move solutions to be accepted repeatedly into the archive, so long as their associated random trajectories differ from those of the existing no-move solutions. However, this also holds true for the L2-AURORA implementation for which we observe a gradual decrease in the PCTMOVE score rather than a sharp collapse at 100% stochasticity.

The EXCLUDE_ARCHIVE benchmark's observed behaviour is driven by the exclusion mechanism's bias to preserve solutions with shorter trajectories. No-move solutions generally tend to incur the lowest errors due to the ease of reconstructing the trajectory of a still-standing puck. Equivalently, the longest solutions in the archive are most likely to incur the largest losses and therefore most likely to be excluded in each iteration. Over time, this leads to an elimination of all the solutions that move the puck. Furthermore, by predominantly retaining no-move solutions in our archive, we increase the likelihood with which a no-move solution is selected by our uniform selection operator. This increases the likelihood further that the cross-over and mutation operators produce yet another no-move solution.

Conversely, consider the case when there is no stochasticity in the environment. Solutions with the largest loss are not the individuals with associated noisy observations since the environment is noise-free. Instead, they are the solutions that have observations that are most different from existing solutions' observations present in the dataset. By definition, this makes them diverse but this diversity leads to a large reconstruction error. The exclusion mechanism encourages the accumulation of solutions that are very similar to those already found. The EXCLUDE_ARCHIVE benchmark therefore produces archives in which all solutions have a very similar actual puck trajectory observation.

This also results in a much smaller archive size when the environment stochasticity is low and there are no noisy trajectories to differentiate the BDs of these solutions. Fig. 7.107 presents the median size of the archives in the last generation of the algorithm. At 0% stochasticity, the archive size does not even reach half of its targeted population size. As the noise levels increase, similar solutions are more likely to be assigned dissimilar BDs and therefore more likely to be accepted repeatedly into the archive. This leads to a growing archive size as the stochasticity increases. Indeed, only at 100% stochasticity do the archives successfully reach the target size of 8000 solutions despite containing almost exclusively no-move solutions.

To understand why the PCTMOVE score does not gradually decrease across noise levels but instead experiences a sharp drop when moving from 80% to 100% stochasticity, we visualised the observations and reconstructions made by the Autoencoder

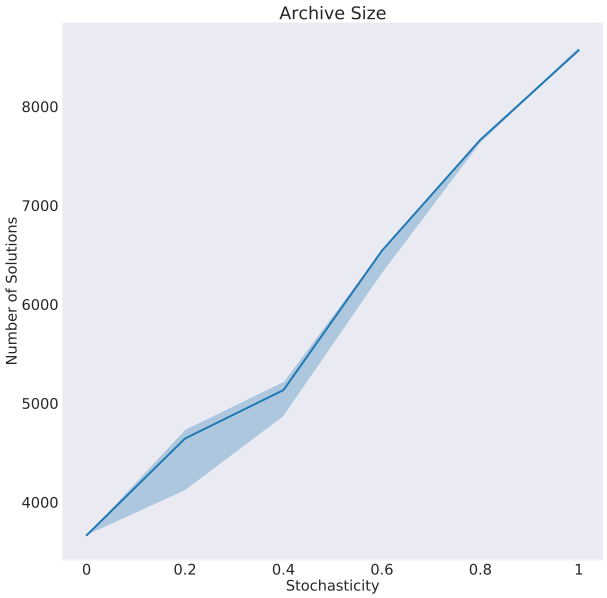


Figure 7.107: Number of solutions in the archives generated by the EXCLUDE_ARCHIVE benchmark.

in EXCLUDE_ARCHIVE.

At 80% stochasticity, the majority of solutions in the archive share the same observation or construction, or indeed in many cases both. This suggests, that in the initial generations of the algorithm a large set of solutions sharing a similar trajectory emerges among the 20% of solutions that are not influenced by noise. Their large number and similarity allows the network to learn a good reconstruction for these solutions early in the training. The solutions in this set are therefore unlikely to be selected by the exclusion mechanism as they will persistently have some of the lowest errors of all samples. We refer to this set as the INIT_SET. Very importantly, the no-move solutions cannot emerge as this INIT_SET, as they share the same BD. They can therefore not be found in the archive in large numbers without having an additional random trajectory.

Over time, two additional types of solutions are added and, more importantly, retained in the archive. Solutions of the first type have an actual puck trajectory that is very similar to the trajectories in the INIT_SET and an additional random trajectory such that their overall BDs are sufficiently novel, yet their loss is still low enough to avoid exclusion. The second type are solutions that have a random trajectory that is very similar to the trajectories in the INIT_SET but a sufficiently dissimilar actual trajectory to, again, be accepted, yet not excluded from the archive subsequently.

Solutions of the first type will be generated more frequently as the large number of solutions in the INIT_SET increases the likelihood of members of this set to be selected for cross-over and mutation operations. Furthermore, there is a limited

variability in the achievable actual trajectories due to the fixed starting location of the puck and the robot arm. Conversely, for the random puck any physically plausible trajectory is possible. This again makes the occurrence of the first type of solution more likely, and the generation of the second type less likely.

Fig. 7.108 presents visualisation of all actual puck trajectories in an archive generated by the EXCLUDE_ARCHIVE benchmark at 80% stochasticity. The large frequency of the first type of solution and the shape of the trajectories in the INIT_SET is very visible. In Fig. 7.109, we present the observation and reconstruction for one of these solutions. We can confirm a strong reconstruction accuracy for the actual puck trajectory of solutions in the INIT_SET.

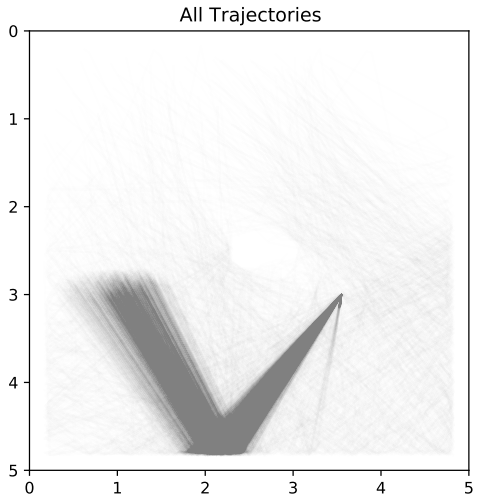


Figure 7.108: Visualisation of the actual trajectories of all solutions in an archive generated by EXCLUDE_ARCHIVE at 80% stochasticity.

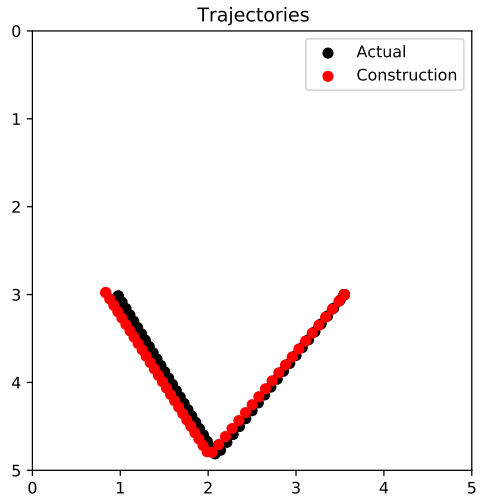


Figure 7.109: Observation and reconstruction of the trajectory associated to a solution.

Since the INIT_SET cannot be a set of no-move solutions, the EXCLUDE_ARCHIVE benchmark generates archives that predominantly contain solutions that move the puck. This explains the large PCTMOVE scores despite substantial amounts of environment noise. At 100% stochasticity all solutions have an additional random trajectory. A set of similar solutions without any random trajectories can therefore no longer emerge as the INIT_SET. Instead, the no-move solutions form the set of individuals that incur the lowest losses, leading to the accumulation of more such solutions over the generations as discussed previously.

Given its tendency to produce only very similar solutions, the EXCLUDE_ARCHIVE benchmark unsurprisingly delivers a very poor diversity performance. We show a minimal diversity score in Fig. 7.110. The EXCLUDE_TRAIN benchmark delivers a competitive performance but fails to improve on the L2-AURORA implementation.

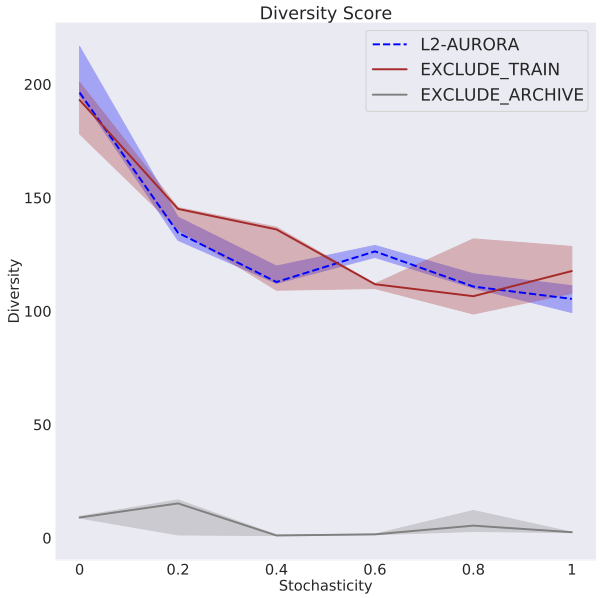


Figure 7.110: The Diversity Score.

In benchmarks EXTEND and REGEN we again select the solutions with the 30% largest reconstruction errors. However, instead of excluding these individuals, we regenerate their associated observations.

In EXTEND, the regenerated solutions are added as additional samples to the training dataset and discarded when the training iteration ends. This benchmark pursues a similar idea to the EXCLUDE_TRAIN benchmark in its attempt to lower the proportion of noisy samples in the dataset. However, it differs in its execution by producing an even larger number of training samples. Despite reducing their relative frequency, this implies that the EXTEND benchmark's dataset contains a larger number of noisy trajectories. This is the opposite of what we try to achieve in the EXCLUDE_TRAIN benchmark. The diversity score in Fig. 7.111 confirms that the approach taken in EXTEND is indeed inferior to the mechanism used in EXCLUDE_TRAIN.

In REGEN, the regenerated observations permanently replace the selected solutions' current observations. In this benchmark, we attempt to eliminate the presence of noisy observations over time, while retaining a stable number of solutions in the archive. The REGEN benchmark also fails to improve on the L2-AURORA implementation's diversity score in Fig. 7.112.

This is again due to the selection mechanism's bias to choose solutions with long actual puck trajectories. These solutions will be regenerated repeatedly until the achieved loss drops significantly which can happen in one of two cases. Either, the regeneration of observations results in a new random trajectory that appears

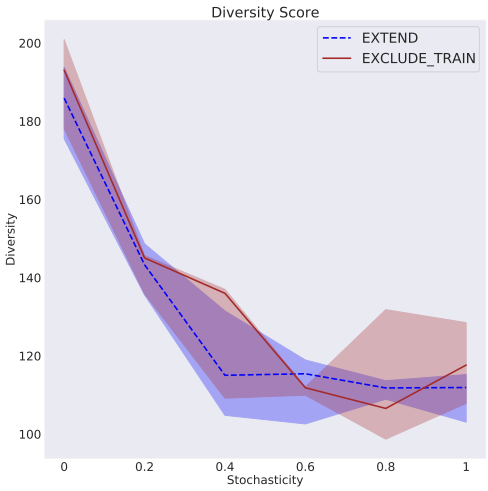


Figure 7.111: The Diversity Score.

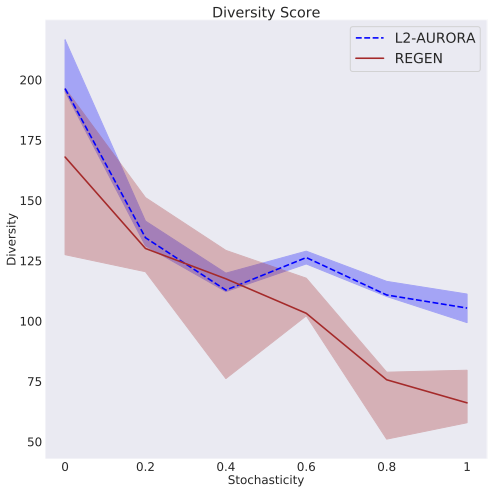


Figure 7.112: The Diversity Score.

frequently in the training set, or the regeneration results in no additional random trajectory at all. In both cases, the resulting BD is likely to be more similar to other existing solutions' BD, resulting in the rejection of these long-distance solutions after regeneration. Fig. 7.113 shows that REGEN produces solutions that have shorter trajectories than in the L2-AURORA implementation.

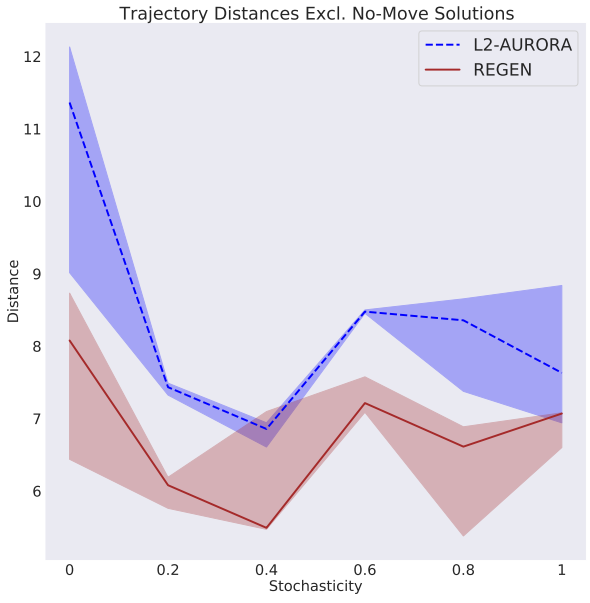


Figure 7.113: The trajectory distances of solutions that move the puck.

We conclude that the adaptations suggested in our benchmarks fail to reduce the AURORA algorithm's sensitivity to noise in the environment observations. Our deci-

sion to choose the solutions with the 30% largest losses was based on the fact that the algorithm would be able to regenerate roughly 30% of the target archive size between each exclusion or regeneration iteration. While a different choice in this hyperparameter value could perhaps have produced a better performance in our benchmarks, the task-specificity of any such improved result is almost guaranteed. With our overall research ambition being the elimination of any prior knowledge of task and environment, the existence of such a sensitive hyperparameter poses an additional significant downside in using one of these proposed mechanisms.

The final evaluation of our best performing architecture therefore features only our baselines. As demonstrated already at the start of Chapter 7.3, the AURORA baseline fails to provide the desired robustness to noise in this set of experiments. Fig. 7.114 shows a PCTMOVE score that declines as the stochasticity increases.

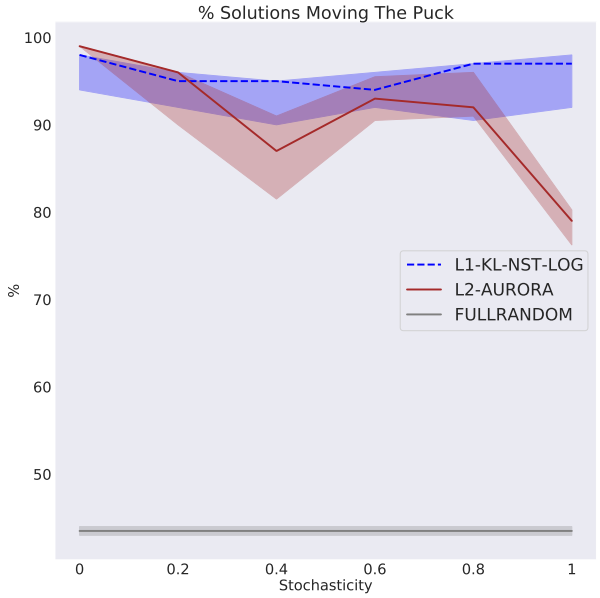


Figure 7.114: The proportion of solutions moving the puck.

Nonetheless, even at large stochasticity levels L2-AURORA still produces a significantly larger score than our RANDOM baseline, in which a similar-sized archive of solutions is randomly sampled from the solution space.
The AURORA algorithm also produces a marginally higher score in a noise-free environment than our best performing L1-KL-NST-LOG architecture. This again demonstrates the large BD accuracy in AURORA that we were not able to fully restore with our attempts to increase the compactness of the BD space.

As a result, at 0% stochasticity the AURORA baseline produces the largest diversity score among all variants evaluated in this experiment. We show the achieved scores for the two baselines and the L1-KL-NST-LOG variant in Fig. 7.115. However, rising

environment noise levels lead to a rapidly increasing inaccuracy in the BDs generated by AURORA. This leads to the frequent inclusion of similar puck trajectories into the archive. The influence of the noise and its impact on the trajectories' similarity appear quite sizable. L2-AURORA produces a rapidly declining diversity score and only a marginally stronger performance than the FULLRANDOM baseline at 100% stochasticity despite a much larger proportion of solutions moving the puck. Indeed, when removing the no-move solutions from consideration, the increasing similarity of puck trajectories produced by AURORA can be seen clearly in the entropy score in Fig. 7.116.

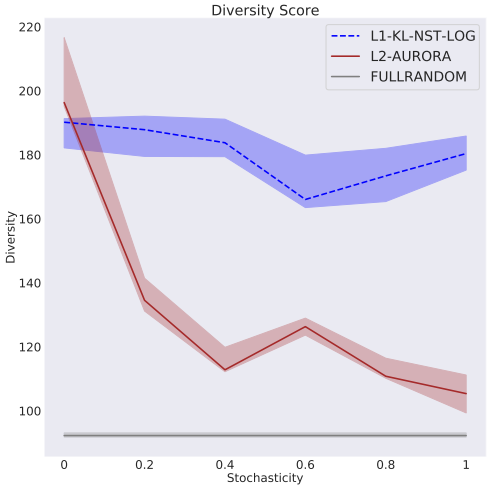


Figure 7.115: The Diversity Score.

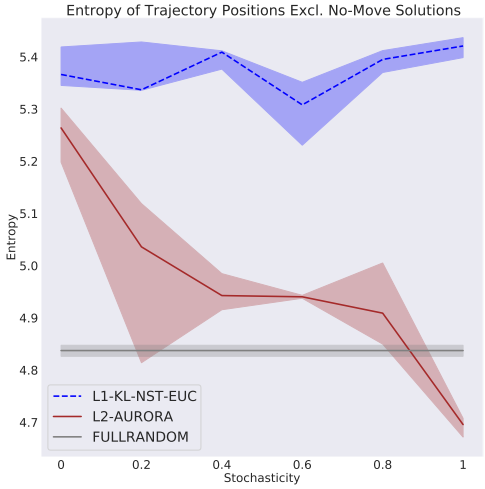


Figure 7.116: The Entropy Score.

Finally, in Fig. 7.117 we present another visualisation of the latent space constructed by the AURORA algorithm. It provides an interesting insight into how the differences in the types of input to the Encoder affect the shape of the space when we compare it to the BD spaces constructed by our best architecture and the most similar variant among our proposed architectures, the L2-AE implementation. Examples of the latter two are shown in Figs. 7.118 and 7.119 respectively. All visualisations are made at 0% stochasticity in the environment.

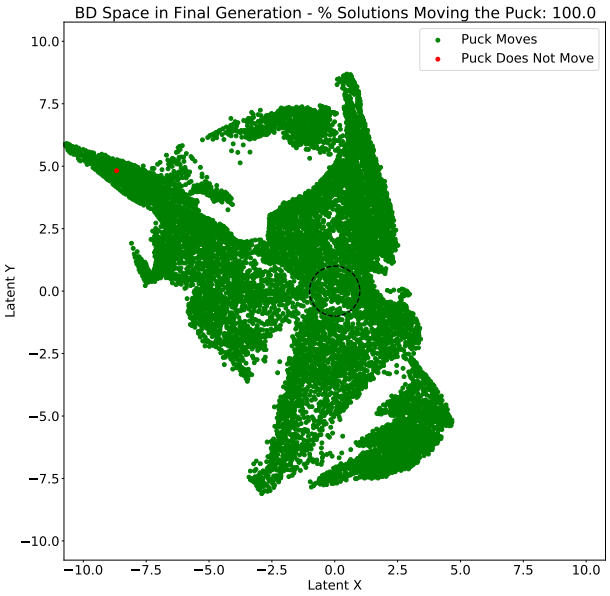


Figure 7.117: BD Space constructed by L2-AURORA at 0% stochasticity.

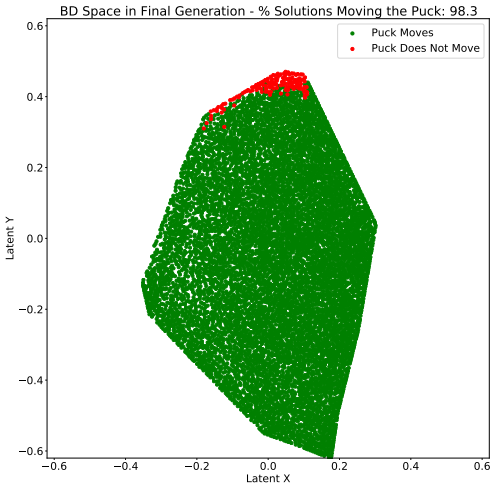


Figure 7.118: BD Space constructed by L1-KL-NST-LOG at 0% stochasticity.

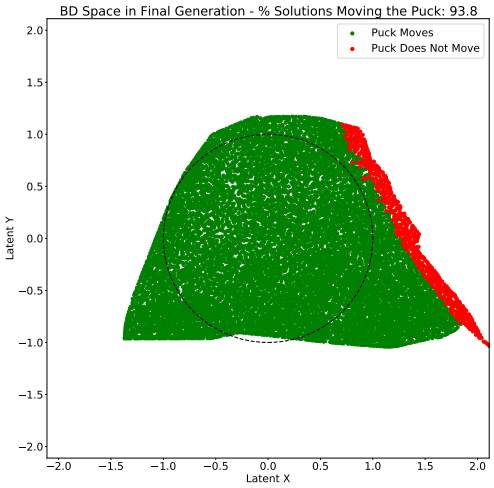


Figure 7.119: BD Space constructed by L2-AE at 0% stochasticity.

7.3.9 Improving the Performance

In addition to varying the architecture design and the composition of the loss function, we can attempt to achieve further performance gains by providing the VAE with more information.

Increasing the Batch Size

As discussed in Chapter 3, larger batch sizes have been shown helpful when datasets are noisy [43]. To understand this in the context of our application, suppose for instance we have a batch containing only one solution with one actual and one random trajectory. The resulting gradient update for the batch will be driven by the mean of the gradients associated to each trajectory's incurred loss. Now suppose we add more solutions with the exact same parameters. Each solution has the same actual trajectory but different random trajectories. The more solutions we add, the more likely will the gradient signals emitted by some of the random trajectories offset the gradients of other random trajectories. Meanwhile, the gradient for the actual trajectory will continue to accumulate. Compared to the case where we only have a single solution in our batch, this leads to a gradient for the batch that grows increasingly similar to the single gradient associated to the actual trajectory.

We initially chose a batch size of 64 in our first set of experiments. After reading the work referenced in the paragraph above we increased the batch size to 256 in all subsequent experiments. In this section, we evaluate the performance of our best architecture with batch sizes of 64, 256 and 1024.

Fig. 7.120 presents the Actual L2 errors achieved. All three implementations produce very similar median errors but the BATCH64 implementation exhibits a larger volatility in the obtained results. Since batches are sampled randomly from the dataset, the composition of samples is more likely to differ between the different batches when the size is smaller. This can increase the influence of noisy samples. However, we can also observe a large volatility in the PCTMOVE score for the BATCH1024 implementation in Fig. 7.121. It is therefore likely that parts of the observed differences in volatility between the implementations arise simply due to the small number of repetitions made to obtain our results.

Figs. 7.122 and 7.123 show the diversity and POSVAR scores respectively. While the BATCH64 variant seems to produce a marginally lower diversity score, no performance differences can be seen in the POSVAR score. We therefore conclude that our algorithm's performance is largely insensitive to changes in the batch size within our evaluated range of 64 - 1024.

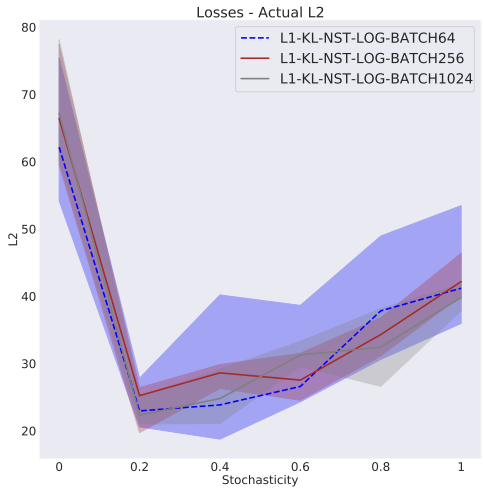


Figure 7.120: The Actual L2 Error.

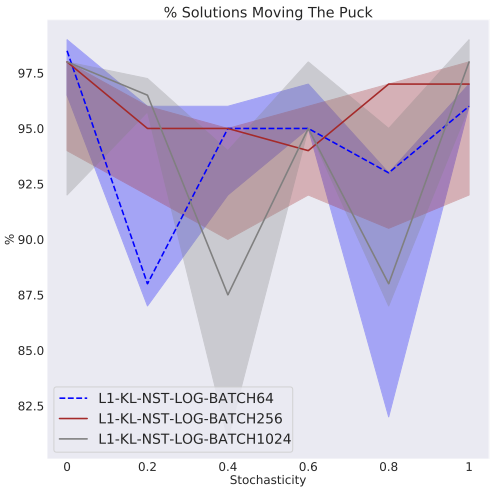


Figure 7.121: The proportion of solutions moving the puck.

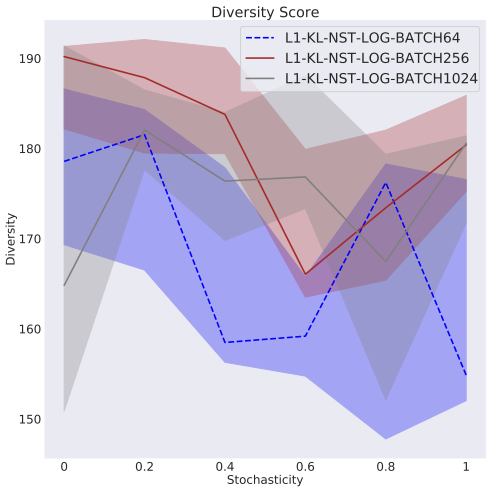


Figure 7.122: The Diversity Score.

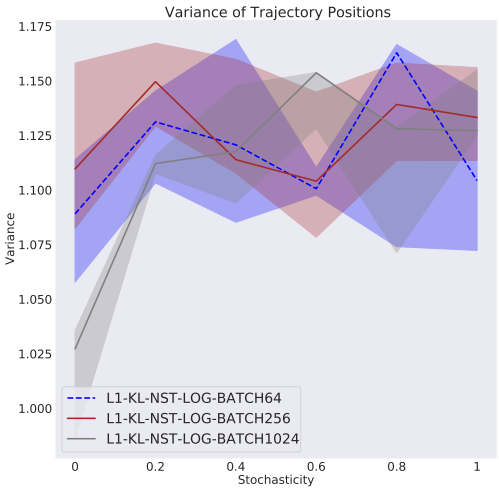


Figure 7.123: The variance in the puck trajectories.

Expanding the Dataset

In our hypothetical batch example, we assumed that we are adding solutions with identical parameters into the same batch. However, this duplication of solutions in the archive is what we explicitly seek to avoid in our search for diverse solutions. Indeed, the same solution cannot be added repeatedly when we do not sample the BD. This limits the frequency with which very similar solutions can be featured in the same batch.

At training time, we would like to have additional solutions available with the same parameters as solutions in the archive that have been influenced by noise. However, we do not want any of these duplicate solutions to be retained in our archive.

We propose to apply a mechanism similar to our EXTEND benchmark. At the start of each training iteration, we select the solutions in the archive with the 50% largest losses. In doing so, we are likely to select noisy individuals. Moreover, we are likely to select the solutions that are most affected by the noise as either significant differences in the actual and random trajectories, or a poor construction of the actual trajectory are causing the large loss. These solutions will therefore benefit the most from having additional samples available for training.
At the start of our training iteration, we create a copy and regenerate the observations for each of these solutions. This additional dataset of features and labels is added to our existing dataset and discarded after training.
In Fig. 7.124, we show that the selection of the solutions with the largest losses works as expected. The selected set contains approximately the maximum number of all available noisy solutions in the archive at each stochasticity level.

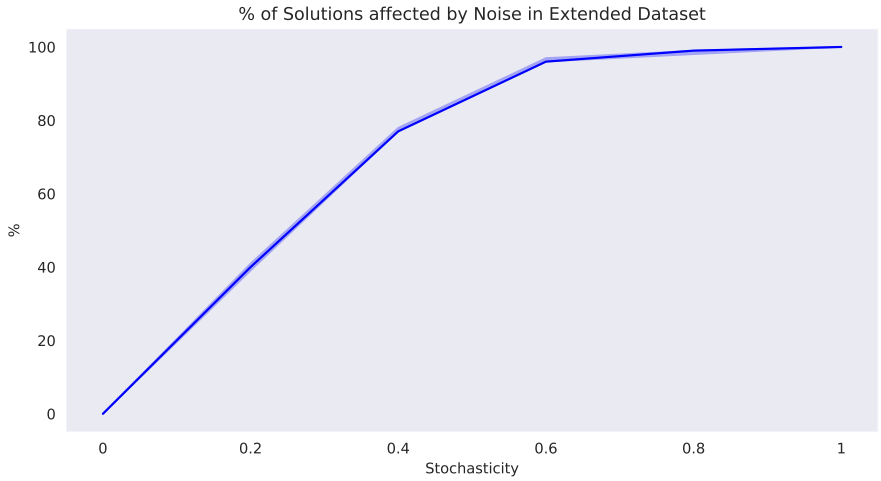


Figure 7.124: Proportion of solutions with noisy trajectories among the set selected to be re-added to the dataset.

However, our EXTEND adaptation does not appear to produce significant changes to the performance of the L1-KL-NST-LOG variant. Figs. 7.125 and 7.126 show a largely similar PCTMOVE and POSVAR score respectively.
Note, however, by using the memorisation effect we introduce the same bias towards the selection of longer solutions that we discussed in Chapter 7.3.8. This can lead to an imbalanced dataset. An improvement in the construction performance for the over-represented long-distance solutions in the dataset can come at the expense of a deterioration in the performance for solutions associated with shorter solutions.

To avoid creating an imbalance while still providing more samples to the dataset, we could copy the whole archive and regenerate all the solutions. Alternatively, we

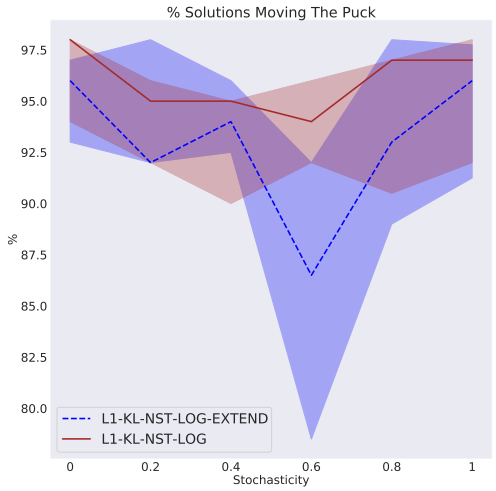


Figure 7.125: The proportion of solutions moving the puck.

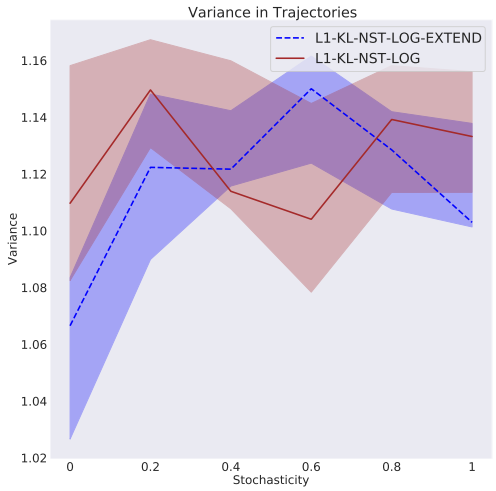


Figure 7.126: The variance in the puck trajectories.

propose the use of additional training archives. When a solution is rejected from the first archive due to its similarity to existing solutions, we attempt to add the solution to a different archive instead. This archive will have its own set of solutions and therefore a different threshold parameter L. If a solution fails to be accepted into any archive, we choose a training archive randomly and replace the solution in that archive that is most similar. At training time, we then extract the contents of all archives to generate the dataset. While this addresses the imbalance issues mentioned above, the approach also features an additional advantage over regenerating one single archive of solutions multiple times.

Doncieux et al. show that the utilisation of an archive leads to uniformly distributed solutions in the BD space [50]. We observed this in the BD space density plots we presented in Chapter 7.3. While a uniform BD space is likely to translate into a uniform distribution in the observation space, it can lead to certain subspaces of the solution space being under-represented in the dataset. Consider for instance the set of solutions that achieve no puck movement. If our architecture achieves its desired performance, our dataset will feature only a single solution from this set as all no-move solutions will have the same BD. More generally, the uniform distribution in the BD space likely leads to a non-uniform distribution in the feature space of our training dataset. This can have negative implications on our network training. We make the assumption of an independently and identically distributed dataset in our use of the mini-batch gradient to approximate the true gradient.
By maintaining separate archives, solutions with similar observations can now feature repeatedly in the dataset. This allows for a better coverage of the solution space in the training dataset. The regeneration of the solutions in one single archive would clearly leave this issue unaddressed.

In Figs. 7.127 and 7.128 we present the achieved Actual L2 errors and the PCTMOVE score.

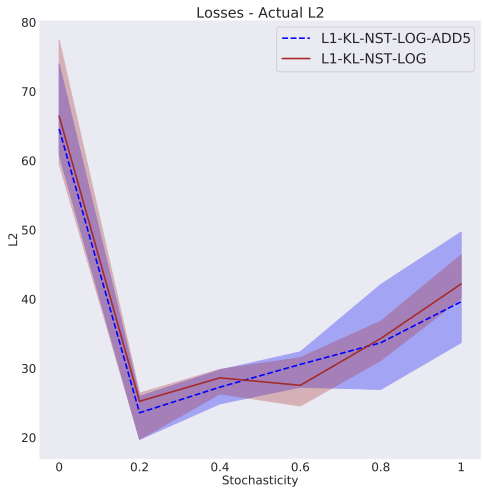


Figure 7.127: The Actual L2 Error.

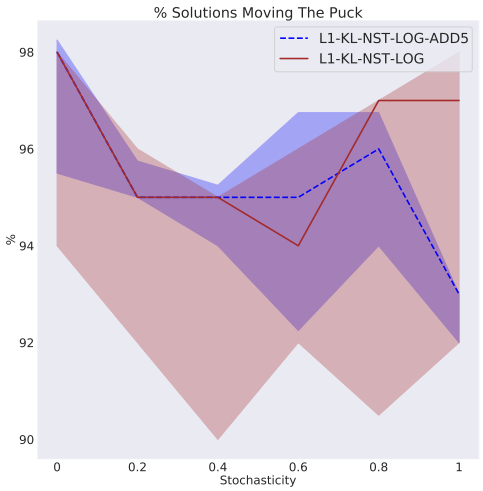


Figure 7.128: The proportion of solutions moving the puck.

The ADD5 adaptation no longer suffers from an imbalanced training dataset. However, the achieved Actual L2 and PCTMOVE scores are very similar to the scores produced by the L1-KL-NST-LOG variant without the additional training archives. The diversity metrics also remain largely unchanged. We show the diversity and entropy scores in Figs. 7.129 and 7.130.

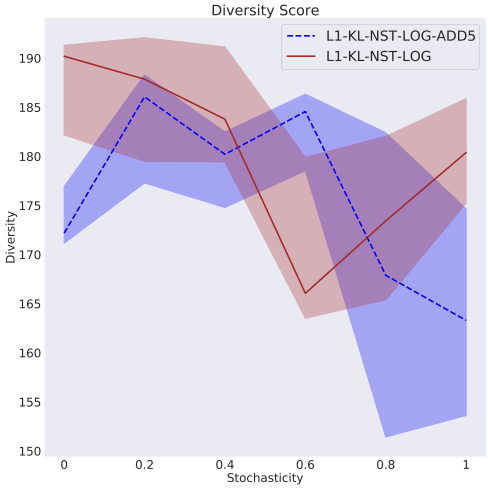


Figure 7.129: The Diversity Score.

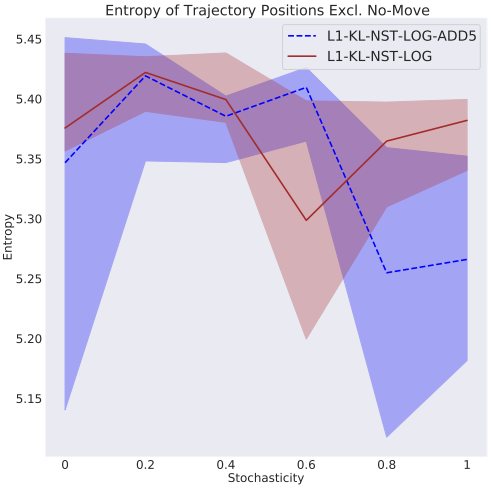


Figure 7.130: The Entropy Score.

The results shown in this section do not offer any evidence of an additional benefit

obtained from increasing the batch sizes or adding the training archives. However, this could be explained by our base architecture achieving a very good performance already such that these additional adaptations, while beneficial, fail to produce improvements of significant size. Indeed, we find that a larger batch size and the use of additional training archives does lead to further performance gains in Experiment 3 in Chapter 8.3.9.

7.4 Conclusion

When the environment is noise-free, the AURORA algorithm produces an archive containing solutions of a large diversity. However, when observations capture environment changes that arise independently of the robot's actions, the BDs no longer accurately reflect the actual behaviour of the robot. The environment stochasticity affects the AURORA algorithm's performance significantly and increased numbers of very similar solutions are accepted into the archive. The diversity of solutions that move the puck is lower in an archive created by AURORA than the diversity of the same set of solutions in an archive generated by randomly sampling in the solution space. We proposed benchmarks that attempt to adapt the algorithm to increase its robustness to noise. However, we determine that none of our suggested benchmarks improve on the performance of the original AURORA implementation.

The presence of a KL divergence criterion in the VAE's optimisation process results in larger Encoder variances. These exert a substantial negative influence on the accuracy of our Behavioural Descriptors and therefore on the diversity of the generated solutions. By assigning the Encoder means as the BDs, we can improve the BD accuracy and our diversity metrics significantly. However, the Encoder variances' influence persists when they are used to perform the reparameterisation trick in the VAE's training process. The algorithm's performance increases further when we eliminate the variance term altogether. With its influence fully removed, the application of the KL divergence criterion leads to a sharp contraction of the occupied BD space. While this produces substantial changes to the development of the value of our adaptive novelty distance threshold, the consequential increase in the BD accuracy still improves the diversity metrics substantially.

The SNE criterion achieves a further compression of the BD space. However, it also introduces a direct relationship between the length of a given solution's trajectory and its perceived similarity to other solutions. This leads to a discrimination of solutions and a preference for solutions with longer trajectories. The generated archives exhibit a poor diversity. The t-SNE criterion too is not able to improve the performance of the algorithm.

The addition of a Decoder variance term leads to a weaker influence of the construction criterion on the VAE's optimisation process. When a KL divergence criterion is applied, this results in a further increase in the compactness of the BD space and an improvement to our diversity metrics. However, we also find that the rapidly

changing composition of the training dataset can cause instability issues when the Decoder variance term is used. The impact is especially significant in a noise-free environment.

We achieve significant improvements in the VAE's noise discrimination ability by replacing the L2 distance metric in the construction loss with an L1 criterion. However, the diversity of the produced archives remains largely unchanged.
Similarly, changing the batch size and augmenting the training dataset does not produce any significant positive impacts on our algorithm's performance.

Nonetheless, we find that our proposed architecture successfully adapts the AURORA algorithm to demonstrate a robust performance in the presence of environment noise. The produced archives exhibit at all levels of stochasticity a diversity that approximates the diversity of an archive generated by the original AURORA implementation in a noise-free environment.

Chapter 8

Experiment 3: Synthetic Images

In the previous experiments, we utilised sequences of ground truth puck positions as environment observations. On one hand, the use of these positions assumes that we have access to these positions. This is not a realistic assumption to make in a real world environment. On the other hand, we are implicitly utilising our prior knowledge of the task, the environment and the desired behaviours in the decision to choose these trajectory positions as our observations. In this set of experiments, we use the ground truth puck position data to create synthetic images which we use as our observations instead.

All characteristics of the robot and the environment are exactly the same as in Experiment 2. We discretise the room into a 20 x 20 grid. For each solution, any bin in the grid visited by any puck in the simulation at any time step, takes a value of 1. All remaining bins are filled with values of zero. The created image can therefore be interpreted as a long-exposure shot taken from a camera mounted on the ceiling. The robot arm is not featured in the image. We present an example of an environment observation in Fig. 8.1. Note, that the distinction between the actual and the random puck in the visualisation is made for analysis purposes. This information is not available in the image data used by the VAE.

8.1 Code Implementation

In this experiment, each solution has exactly one associated observation. This allowed us to simplify the BD generation pipeline described in Chapter 6.1.

The AURORA Encoder is replaced with a Convolutional Neural Network. In both AURORA and our VAE, the previously used Decoder network is replaced with a Convolutional Neural Network composed of transposed convolution layers. The hyperparameters controlling the algorithm and network architecture are given in Appendix B.1 and B.4.

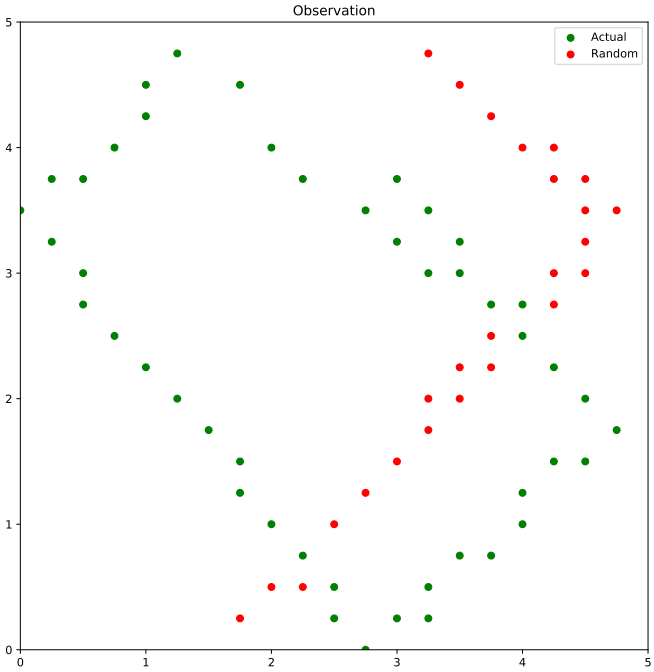


Figure 8.1: Synthetic image with two pucks present in the environment. Activations are thresholded at a value of 0.5.

8.2 Performance Measures

8.2.1 VAE Performance Measures

Undisturbed L2 Metric

Following the change in observation type, we now can no longer consider only the errors associated to the trajectory of the actual puck. To obtain a construction error metric that is not influenced by the random puck, we generate a new image in which only the actual puck is captured. This image is then used to calculate the L2 error. Note, this also removes the influence of any collisions. We refer to this metric as the "Undisturbed" L2 error.

8.3 Results and Discussion

The results obtained in this experiment largely mirror the findings made in Experiment 2 with some notable exceptions. Where applicable, we therefore omit any detailed explanations but briefly review our conclusions made previously and refer to the relevant previous sections.

Fig. 8.2 presents the diversity score achieved by our L2 variants and the AURORA implementation. Both VAE variants produce poor diversity scores. While the Euclidean variant exhibits some robustness to noise, the score is significantly lower than in the AURORA baseline at all stochasticities. The Log-Likelihood variant and the AURORA algorithm produce a similar performance with both appearing equally affected by the environment noise. On the other hand, the L2-AE variant demonstrates a strong and robust diversity performance. This suggests that the VAE variants are likely to benefit from a decision not to sample the BDs.

Fig. 8.3 presents the PCTMOVE scores. The performance of the AURORA baseline in this metric is very interesting, as it seems to indicate some robustness for low to moderate levels of stochasticity.

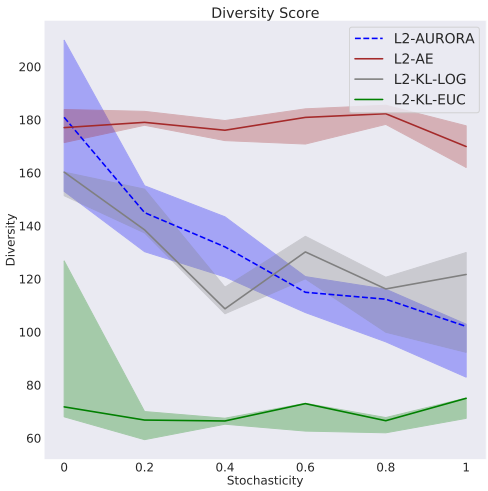


Figure 8.2: The Diversity Score.

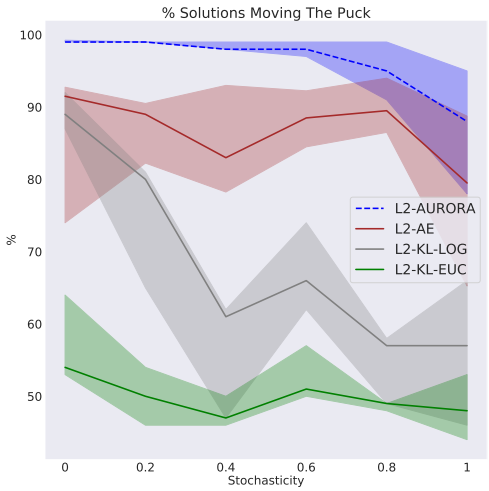


Figure 8.3: The proportion of solutions moving the puck.

At the start of each simulation, the robot arm is positioned in the same fully stretched neutral configuration and the actual puck is returned to its fixed starting position. Certain areas of the room will therefore rarely be visited by the actual puck. Random trajectories too will only appear infrequently in these areas, when the environment stochasticity is low. The network therefore tends to ignore any pixel activations in these areas of the image observations. Fig. 8.4 provides an example where this can be seen in the network's reconstructions. In the bottom box of the visualisation we show the pixel-wise L2 errors. Note the errors for the 3 pixels where the actual puck's trajectory would have crossed the random puck's trajectory if it had continued on its path in the bottom left quadrant of the room. In these pixels, the random trajectory crosses an area of the room that is frequently visited by the actual puck and the network is therefore much more inclined to activate them. This results in a relatively lower L2 error.

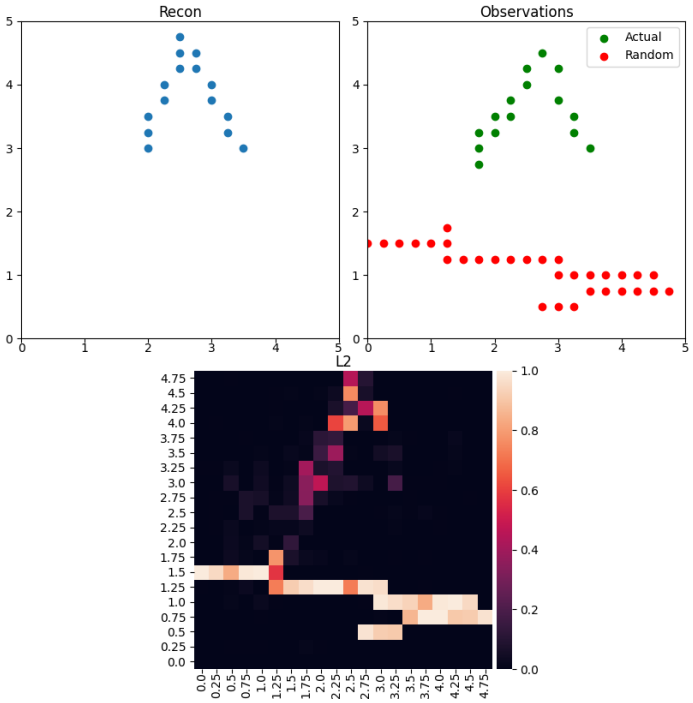


Figure 8.4: Top Left: Constructed image with pixel activations thresholded at a value of 0.5. Top Right: Actual and random puck trajectories in the original observation. Bottom: Pixel-wise L2 error. L2-AURORA at 40% stochasticity.

By failing to learn activations that occur too infrequently, the AURORA baseline implicitly eliminates some of the noise in the observations from its reconstructions without any modifications to the architecture. The algorithm therefore benefits from the limitations in the variability of the possible puck trajectories in this environment. However, this is clearly specific to our experiment setup. For instance, we could devise a similar experiment in which we expand the solution space to allow solutions to also control the shape of the robot arm at the start of the simulation. This would result in more possible directions in which the puck can be pushed and therefore a larger variety of possible trajectories. In such an experiment, we would expect the same noisy trajectories to have a much larger impact on the reconstructions made by the AURORA network.

Nonetheless, this observation alone seems unable to reconcile the strong and robust performance in the PCTMOVE metric in Fig. 8.3 and the rapidly deteriorating diversity score in Fig. 8.2. The former suggests a strong noise discrimination ability but the latter indicates the opposite.
While the frequency of activations for specific pixels is an important factor, the AURORA Autoencoder does not make its reconstruction decisions for each pixel in isolation. It additionally considers the overall shape of the trajectory in its constructed image. A random puck crossing an area of the room that is frequently observed to be occupied does not generate any isolated activations in the reconstruction. In Fig. 8.5, we can see no reconstruction activations in the bottom-left quadrant for any of the pixels occupied by the random puck's trajectory, despite being positioned

in an area of the room frequently visited by the actual puck, as demonstrated by an example in Fig. 8.6.

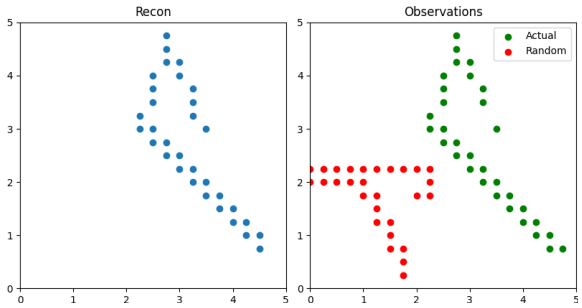


Figure 8.5: Construction (left) of the original observation (right). L2-AURORA at 40% stochasticity.

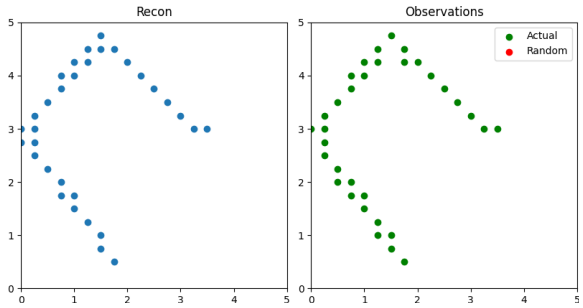


Figure 8.6: Construction (left) of the original observation (right). L2-AURORA at 40% stochasticity.

Observations in which the actual puck moves further and into more different areas of the room are therefore more susceptible to noise as they are more likely to be intersected by a random trajectory. These intersections can mislead the network to reconstruct alternative but plausibly shaped trajectories. Solutions with long trajectories are therefore most affected by the noise. At the same time they make the largest contributions to the diversity score. This explains the rapidly deteriorating diversity score in Fig. 8.2.
Conversely, no-move solutions are much more likely to avoid any intersections with random trajectories and the network can therefore discriminate the noisy elements in their observations very effectively. This results in the large PCTMOVE score in Fig. 8.3 at moderate levels of stochasticity. A growing environment stochasticity leads to a larger number of random trajectories intersecting the starting location of the puck. At large noise levels, this results in the observed deterioration of the PCTMOVE metric.

8.3.1 Effects of the KL Divergence Term

When we removed the KL criterion in Chapter 7.3.1, we found that the VAEs learned to produce a very small Encoder variance. This is no different in this experiment as illustrated by the variance values in Fig. 8.7. However, we can observe large differences between the experiments in the LOG and EUC variants' response to the addition of the KL criterion. In line with our previous results, the Log-Likelihood variant's variance values grow larger as the stochasticity increases. The Euclidean variant however, maintains the same variance value which is also significantly larger.

To understand this difference in behaviour, we need to consider both variants' L2 construction performances. Fig. 8.8 shows that the LOG variant incurs a larger Undisturbed L2 error. This is a consequence of the instability introduced by low Decoder log-variance values in the initial generations of the algorithm. We previ-

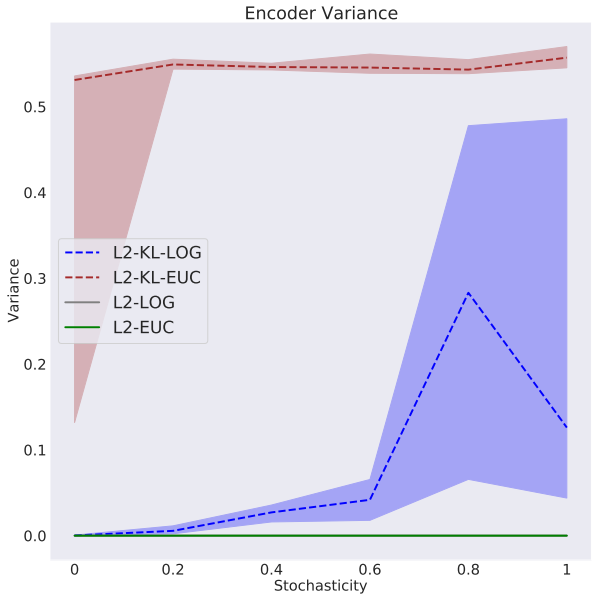


Figure 8.7: The Encoder variance.

ously analysed this effect in Chapter 7.3.5 and discuss it in the specific context of this experiment in Chapter 8.3.4.

The significantly increased L2 errors result in a magnitude of the construction loss' gradient that is larger for the LOG variant than for the Euclidean variant, despite the presence of a Decoder variance term in the former. In the VAE optimisation, the LOG variant therefore places a larger weight on the construction criterion than the Euclidean variant does. This leaves the latter with more capacity to reduce the KL loss. Fig. 8.9 shows a consistent low KL loss in the EUC variants which in turn yields large Encoder variances.

Large Encoder variance values lead to a reduction in the BD accuracy when we sample from the Encoder. The larger the variance, the more likely are similar solutions assigned dissimilar BDs. The KL variants' large Encoder variances therefore lead to a lower PCTMOVE score in Fig. 8.10. The LOG variant's gradually decreasing score matches its gradually increasing Encoder variances in Fig. 8.7. Conversely, the KL-Euclidean variant maintains the same large Encoder variance values in Fig. 8.7 and consequently the same low PCTMOVE score in Fig. 8.10. The KL variants' poor PCTMOVE scores also explain why they appear to produce a superior Undisturbed L2 performance in Fig. 8.8 compared to the no-KL implementations, despite the presence of an additional optimisation criterion. No-move solutions tend to incur the smallest L2 errors as their observations consist of a single pixel activation that is shared across all observations in the dataset. The large differences in the proportion of no-move solutions distort our L2 metric.

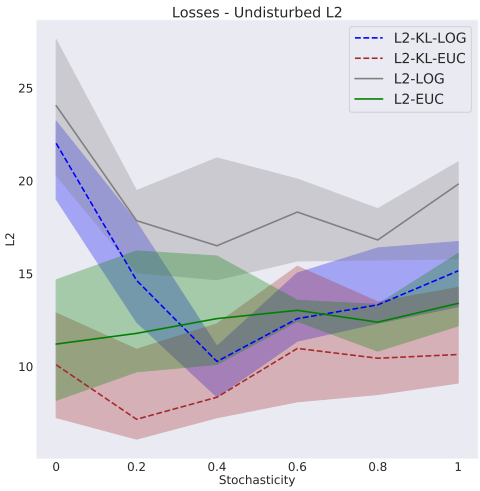


Figure 8.8: The Undisturbed L2 Error.

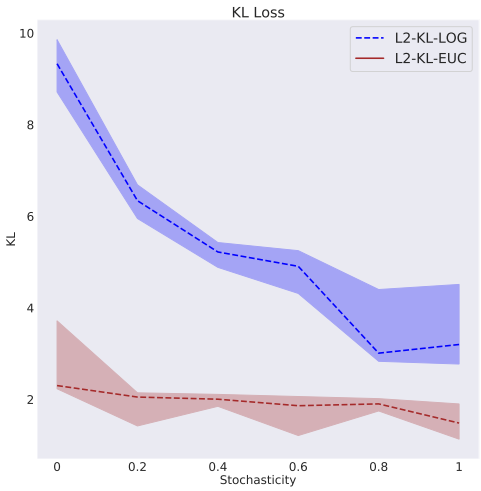


Figure 8.9: The KL Loss.

The large disparity in the achieved PCTMOVE scores is also reflected in our diversity measures. The KL variants produce a significantly lower diversity score in Fig. 8.11.

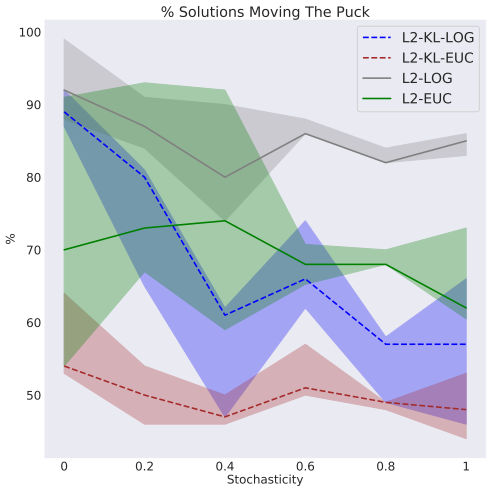


Figure 8.10: The proportion of solutions moving the puck.

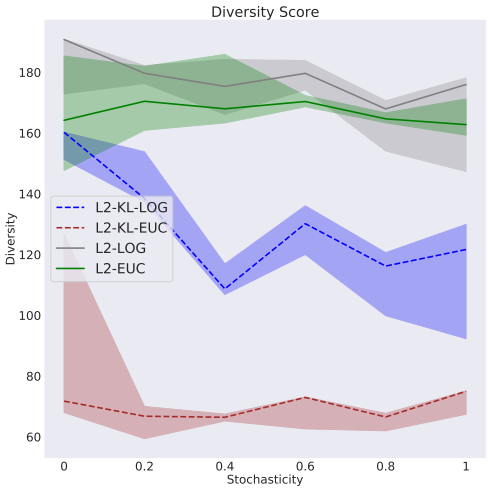


Figure 8.11: The Diversity Score.

8.3.2 Sampling from the Encoder during Inference

The L2 variants' minimal Encoder variance values render the decision whether to sample the BD from the Encoder inconsequential. The L2-NS and L2 variants produce the same variance in the no-move solutions' latent descriptors in Fig. 8.12.

This indicates that the density of the occupied BD space remains unaffected. Consequently, no changes in the diversity performance are observed. We report a largely unchanged POSVAR score in Fig. 8.13.

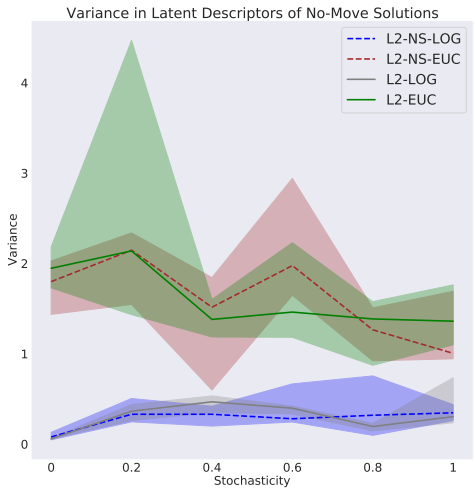


Figure 8.12: The variance in the no-move solutions' latent descriptors.

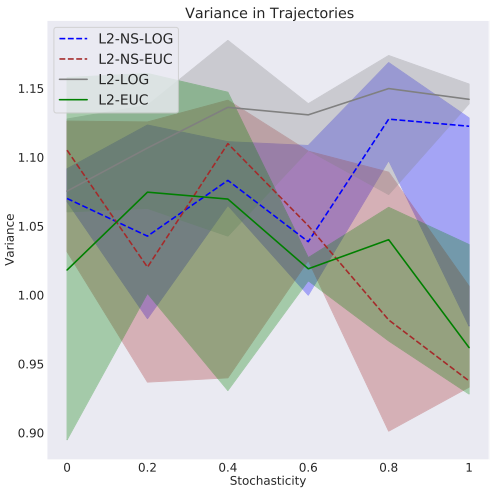


Figure 8.13: The variance in the puck trajectories.

In the KL variants, the decision to sample the BD is much more impactful. We saw in Fig. 8.7 that the Encoder variances are largest in our KL-Euclidean implementation. The consequences of sampling the BD are therefore especially pronounced for this variant. Figs. 8.14 and 8.15 illustrate the latent spaces constructed by the KL-Euclidean and KL-NS-Euclidean variants respectively. While the Encoder means are placed along the y-axis in both latent spaces, the subsequent sampling of the BDs leads to a huge expansion in the size of the occupied BD space in Fig. 8.14.
The Log-Likelihood variant produces significantly lower Encoder variance values in Fig. 8.7. In Fig. 8.16, we present a BD space constructed by the L2-KL-NS-LOG variant at 100% stochasticity when we do not sample from the Encoder. The Encoder means are now no longer placed along one of the latent space axes as they were for the Euclidean variant. Instead, despite no sampling of the BDs, the constructed space looks much more similar to a space obtained when we do sample. We present an example of such a space in Fig. 8.17 for the L2-KL-LOG variant.

Given the stark contrast observed for the Euclidean variant, we expect it to be affected much more heavily by the decision not to sample the BDs. Nonetheless, the LOG variant too still experiences an increase in the BD space compactness as shown by the reduced variances in the no-move solutions' latent descriptors in Fig. 8.18.
The increased compactness translates into larger PCTMOVE scores for both variants in Fig. 8.19. These in turn lead to substantial increases in the diversity score in Fig. 8.20.

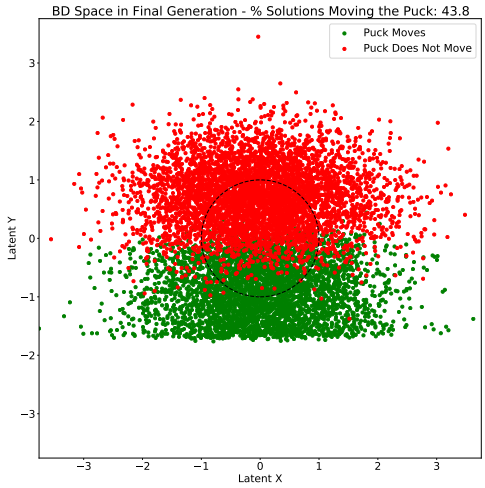


Figure 8.14: BD Space constructed by L2-KL-EUC at 100% stochasticity.

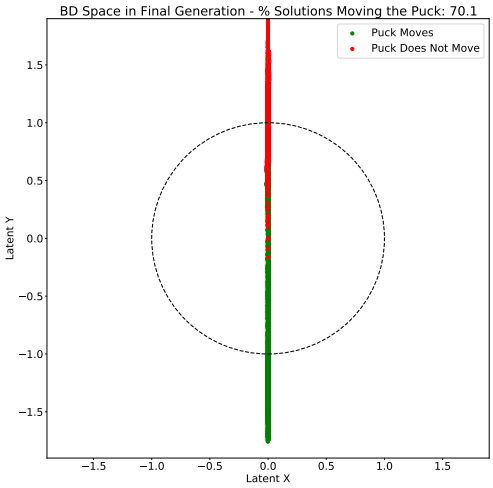


Figure 8.15: BD Space constructed by L2-KL-NS-EUC at 100% stochasticity.

Figure 8.16: BD Space constructed by L2-KL-NS-LOG at 100% stochasticity.

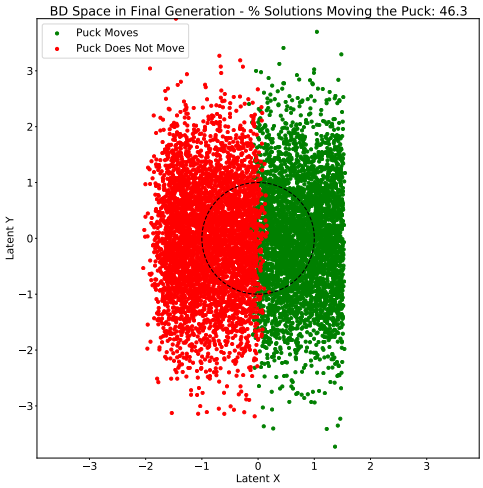


Figure 8.17: BD Space constructed by L2-KL-LOG at 100% stochasticity.

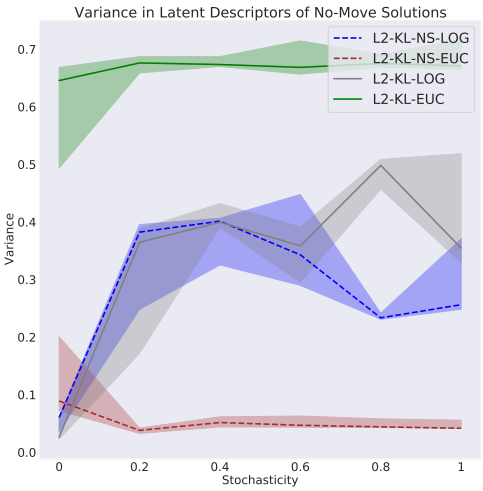


Figure 8.18: The variance in the no-move solutions' latent descriptors.

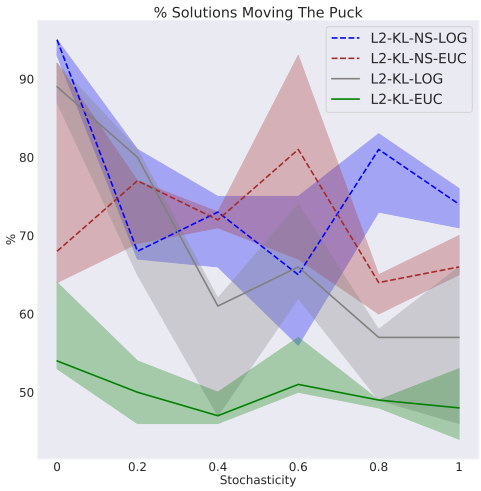


Figure 8.19: The proportion of solutions moving the puck.

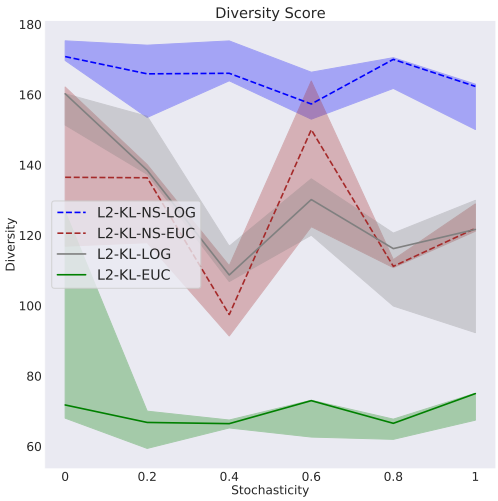


Figure 8.20: The Diversity Score.

Following the KL variants' significant performance gains, we see a much more similar diversity performance between three of our four variants when we do not sample the BD from the Encoder. We show this in the diversity score in Fig. 8.21. The L2-KL-NS-EUC variant however, still produces a much lower diversity score as its Encoder variance values are substantially larger in Fig. 8.22.

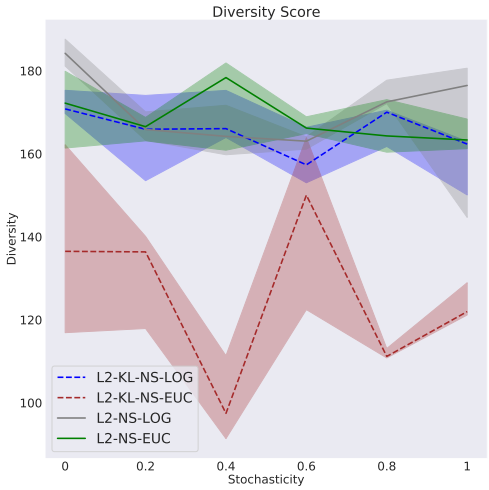


Figure 8.21: The Diversity Score.

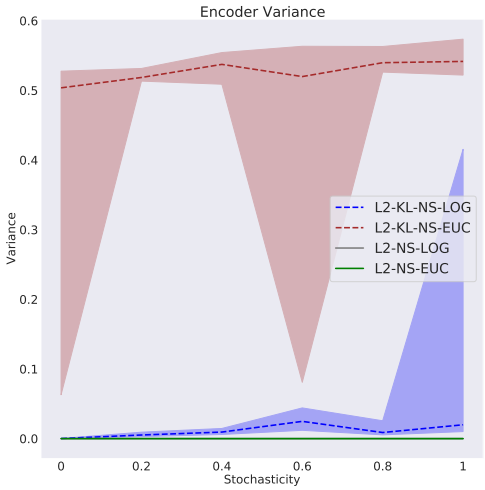


Figure 8.22: The Encoder variance.

The overall impact of imposing a KL divergence criterion is the result of a trade-off between two immediate consequences. On one hand, by pressuring the Encoder means to converge to the BD space origin we increase the compactness in the BD space. In Chapter 7.3, we determined that this has a large positive impact on our algorithm's diversity performance. On the other hand, by incentivising larger Encoder variance values, we reduce the accuracy with which similarities between solutions' BDs capture the similarities between their associated observations.
By assigning the Encoder means as the BDs, we are able to reduce a large portion of the variance values' influence. This results in the KL variants' large performance gains observed in this section. However, the L2-KL-NS-EUC variant's significantly lower diversity score shows that we have not fully eliminated the influence of the Encoder variances yet as they are still utilised to sample the descriptors during training.

8.3.3 Sampling from the Encoder during Training

Fig. 8.23 shows the latent space constructed by the L2-KL-NST-EUC variant. With the influence of the large Encoder variances removed, the Encoder means no longer need to be aligned along one of the latent axes as we saw previously in the L2-KL-NS-EUC variant in Fig. 8.15. This effectively adds another dimension for the Encoder to place its means. The variance in the no-move solutions' latent descriptors in Fig. 8.24 decreases for both KL variants. This indicates that the distance between the solutions' representations shrinks.

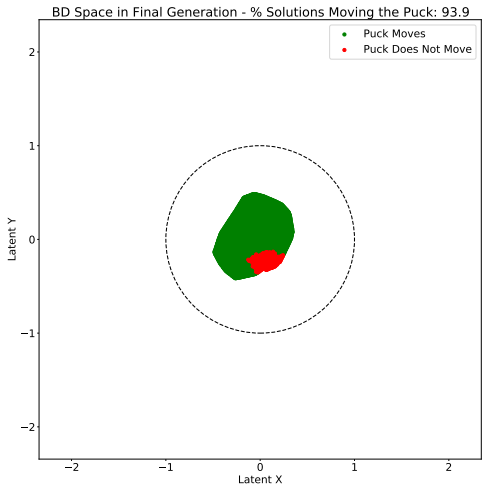


Figure 8.23: BD Space constructed by L2-KL-NST-LOG at 100% stochasticity.

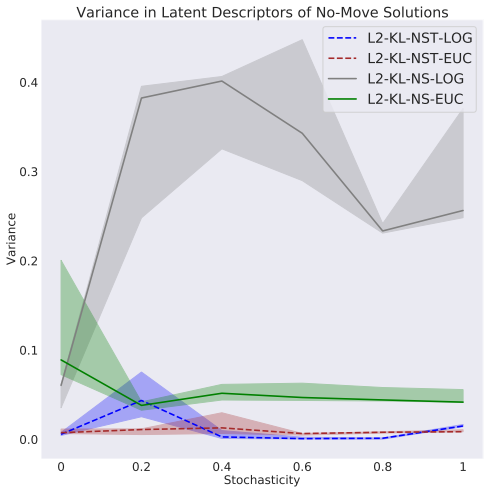


Figure 8.24: The variance in the no-move solutions' latent descriptors.

We therefore obtain a more compact BD space by choosing not to sample the representations when training the VAE. In both KL variants, this leads to an improved PCTMOVE score in Fig. 8.25.

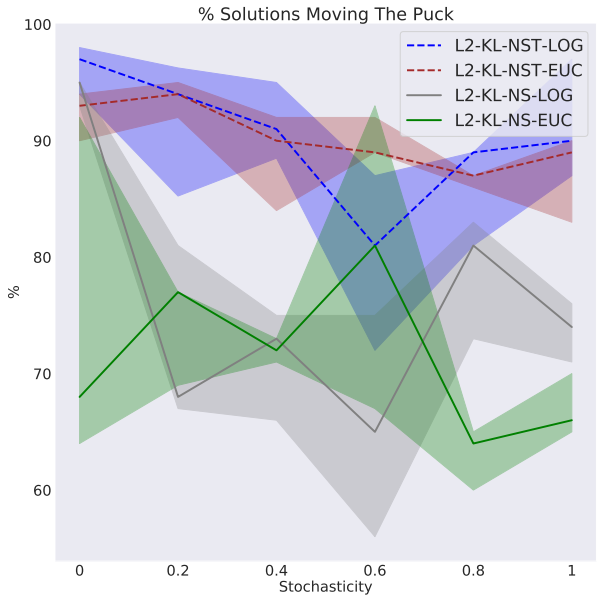


Figure 8.25: The proportion of solutions moving the puck.

These improved scores translate into a significantly stronger diversity performance, especially for our Euclidean variant. We show both KL variants' achieved POSVAR and diversity scores in Figs. 8.26 and 8.27.

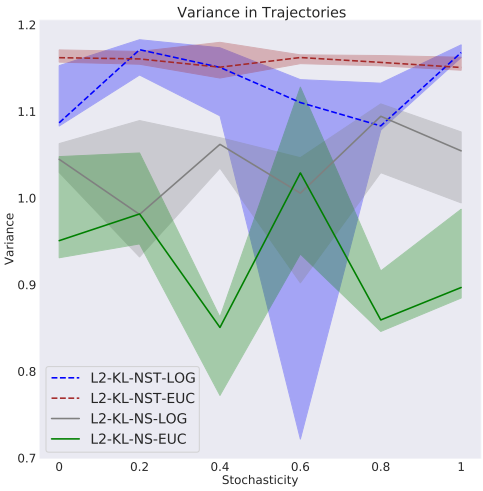


Figure 8.26: The variance in the puck trajectories.

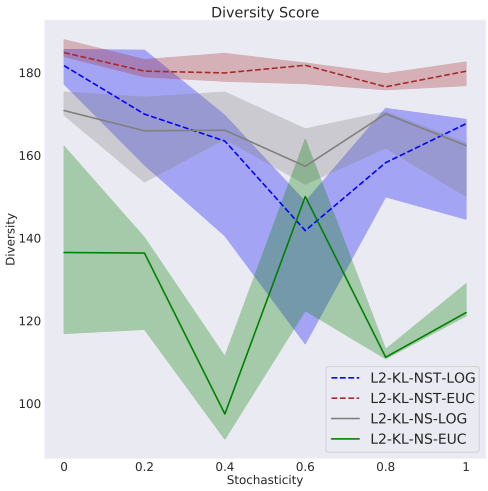


Figure 8.27: The Diversity Score.

The Encoder variances are substantially smaller in our L2-NS implementations compared to their KL counterparts. Nonetheless, we found in Chapter 7.3.3 that even variances of small magnitudes can still contribute to a significant dispersion in the Encoder means.

Figs. 8.28 and 8.29 present two BD spaces constructed by the L2-NST-EUC and L2-NS-EUC variants respectively. A significant reduction in the size of the occupied space is achieved in Fig. 8.28 when the representations are no longer sampled during training.

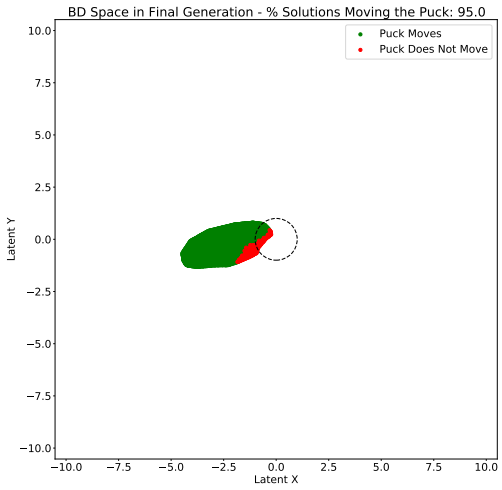


Figure 8.28: BD Space constructed by L2-NST-EUC at 100% stochasticity.

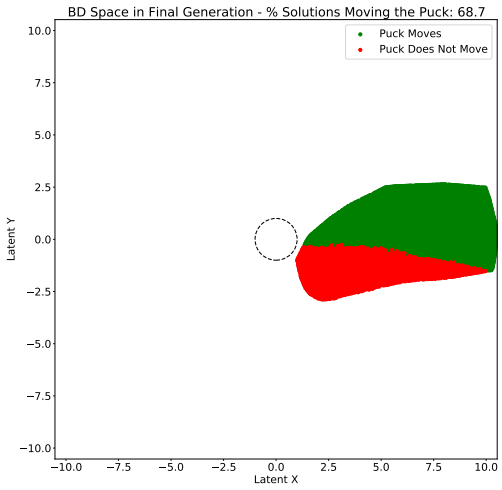


Figure 8.29: BD Space constructed by L2-NS-EUC at 100% stochasticity.

This increased compactness translates into a larger PCTMOVE score in Fig. 8.30 and a superior diversity performance as measured by the POSVAR and diversity scores in Figs. 8.31 and 8.32 for the EUC variant. The L2-NS-LOG variant's Encoder variance values are of even smaller magnitudes. The elimination of the variance term does not have a substantial impact. Its achieved scores in Figs. 8.30, 8.31 and 8.32 remain unchanged. The large volatility in the LOG variants' results is again caused by the instability introduced by the Decoder variance term. We discuss this in Chapter 8.3.4.

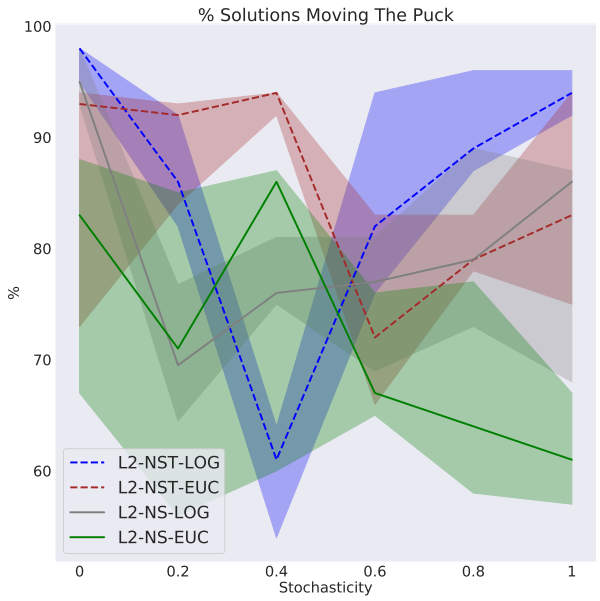


Figure 8.30: The proportion of solutions moving the puck.

When comparing the no-KL and KL variants, the latter now benefit from an increased compactness in the BD space without incurring any of the negative consequences of having a larger Encoder variance. The previously discussed trade-off when imposing the KL criterion is now no longer existent. Only its positive influence on the BD accuracy remains. The NST variants produce a PCTMOVE score in Fig. 8.33 that is significantly larger when the KL criterion is implemented. As expected, this yields a stronger diversity performance as shown by the POSVAR score in Fig. 8.34.

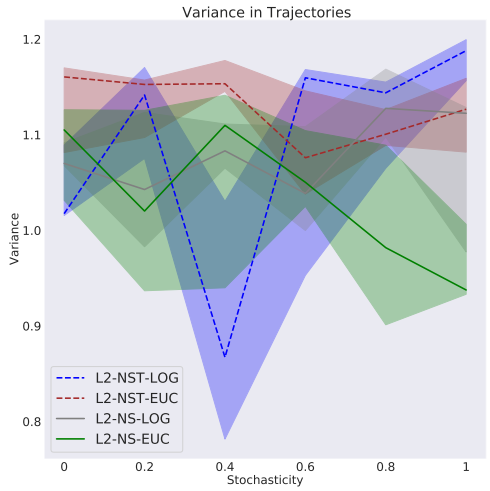


Figure 8.31: The variance in the puck trajectories.

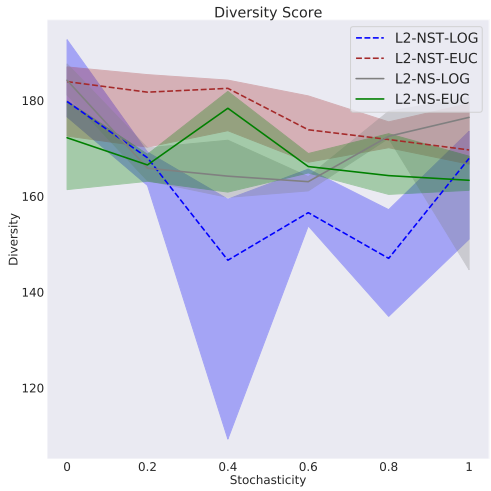


Figure 8.32: The Diversity Score.

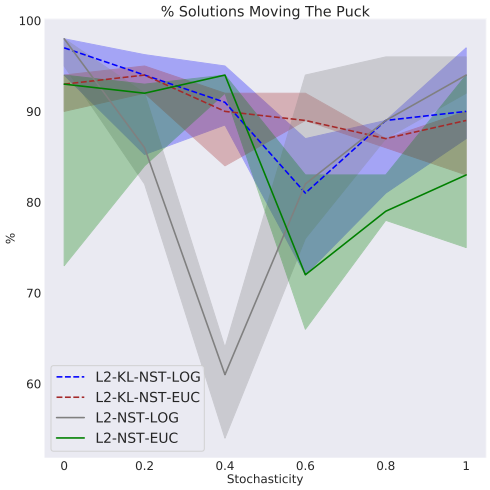


Figure 8.33: The proportion of solutions moving the puck.

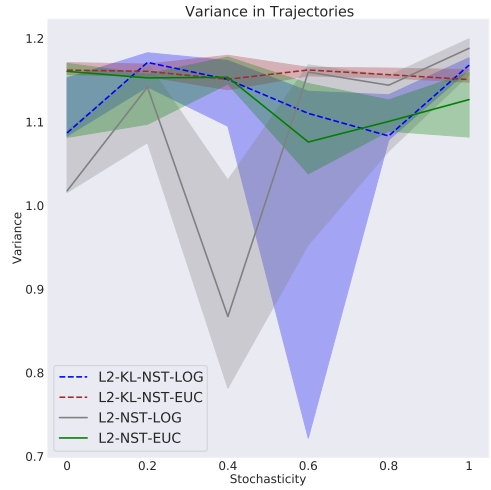


Figure 8.34: The variance in the puck trajectories.

8.3.4 Log-Likelihood Loss

In the majority of the Undisturbed L2 statistics shown in the previous sections, we observed the LOG variants incurring much larger construction errors than their Euclidean counterparts. This can be seen clearly in the L2 construction performances of our L2-KL-NST variants shown in Fig. 8.35.

The Euclidean variant behaves largely as expected. Its construction performance deteriorates as the stochasticity increases. On the other hand, the Log-Likelihood variant seems to achieve its best construction performance at 100% stochasticity

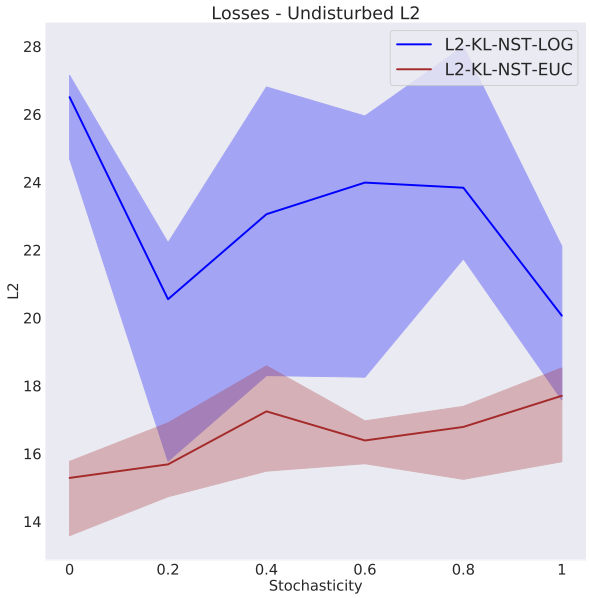


Figure 8.35: The Undisturbed L2 Error.

and its worst when the environment is noise-free.

The latter observation is the only parallel we can draw to our previous discussions on the impact of the Decoder variance term, but it is a crucial one in explaining the observed behaviour. In Chapter 7.3.5, we found that in Experiment 2 the LOG variants also incur an L2 construction error that is larger when the environment is noise-free than when it is stochastic. This is caused by an abrupt increase in the Decoder log-variances after having been reduced to large negative values in the first training iterations when the dataset is small and homogeneous. As a result, L2 errors on novel samples can be reduced only slowly due to the large variances lowering the magnitudes of the construction gradients.

However, this effect lost its impact almost entirely, when we introduced random trajectories to 20% of all solutions in the previous experiment. In constrast, the losses presented in Fig. 8.35 seem to indicate that this process is much more gradual in this set of experiments. The loss keeps decreasing well beyond the 20% stochasticity mark and reaches its lowest levels only at 100%.

The disparity in the observed behaviours is driven by the use of different observation types in each experiment. The image observations take on pixel-wise values between 0 and 1. Generally speaking, the L2 errors incurred are therefore of smaller magnitudes than for observations consisting of the puck trajectories' coordinates. As the additive term in the logvar gradient in Eq. 7.2 is constant, Decoder logvar levels are reduced much more consistently to low values even as random trajectories start

to feature in the image more frequently. In this experiment, the instability therefore dissipates at a much slower rate as the environment stochasticity increases. This leads to a significant volatility across all noise levels and a negative trend in the LOG variant's Undisturbed L2 error in Fig. 8.35.

Fig. 8.36 shows an illustration of this effect at 0% stochasticity.

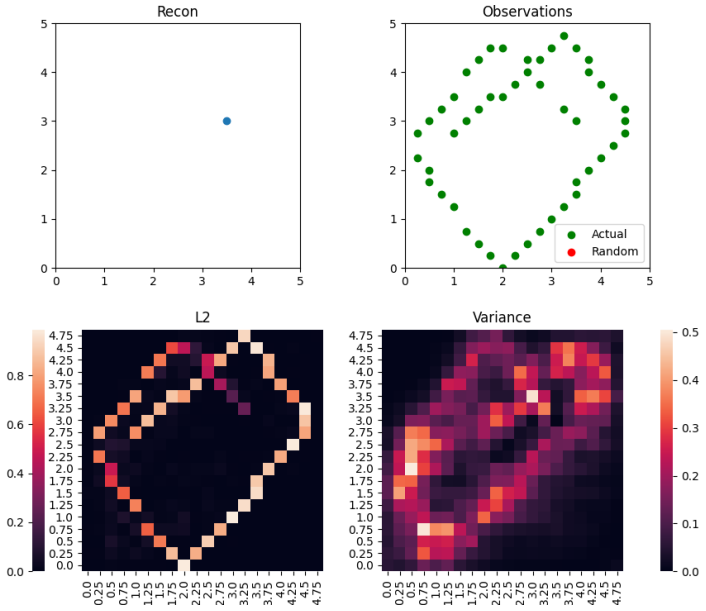


Figure 8.36: Top Left: Constructed image with pixel activations thresholded at a value of 0.5. Top Right: Actual and random puck trajectories in the original observation. Bottom Left: Pixel-wise L2 error. Bottom Right: Pixel-wise Decoder variance outputs. L2-KL-NST-LOG at 0% stochasticity.

The Decoder variances in the bottom-right box trace out the observation correctly. However, the L2 errors in the bottom-left remain very large as the network can only very slowly reduce them. These statistics are taken after the last generation of the algorithm. This shows that this effect persists for the 5000 generations following the archive's stabilisation as shown by an unchanging diversity score after 1000 generations in Fig. 8.37.

Conversely, at 100% stochasticity the variance outputs are much larger for many parts of the image in Fig. 8.38. The increased presence of random elements in the images leads to larger Decoder logvar values in the initial training iterations. When novel trajectories incur a large L2 error, the subsequent jump in the logvar gradient is significantly reduced. The L2 construction can therefore improve more rapidly which leads to a good construction accuracy at the end of 6000 generations.

The Log-Likelihood variant's large volatility in the construction accuracy affects its diversity performance. Fig. 8.39 shows the diversity score. The large fluctuations in this metric can be traced back to an equally large volatility in the distance travelled by the puck in Fig. 8.40.

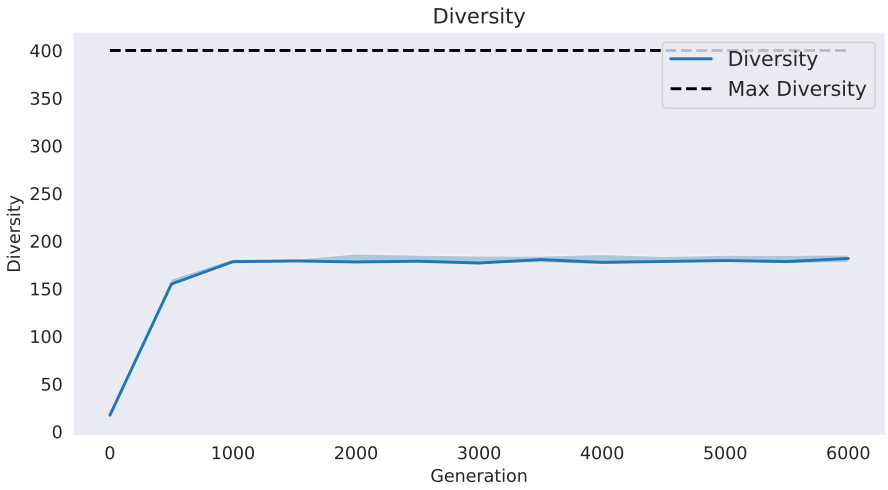


Figure 8.37: The Diversity Score.

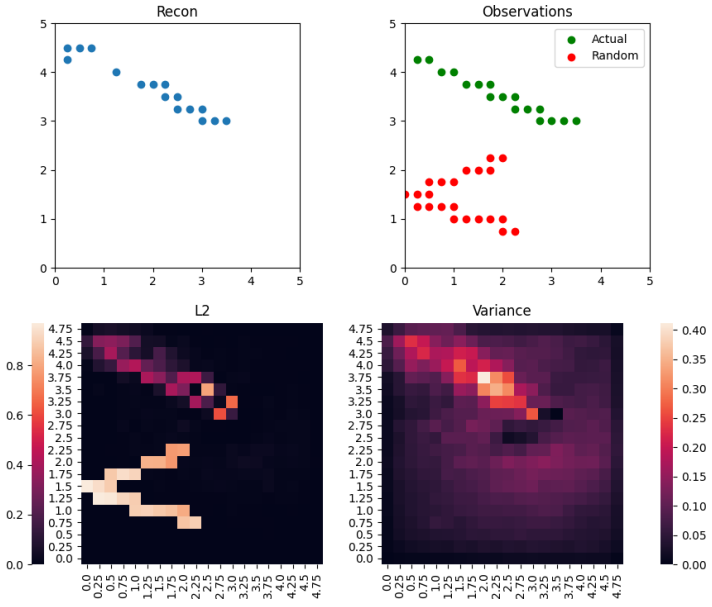


Figure 8.38: Top Left: Constructed image with pixel activations thresholded at a value of 0.5. Top Right: Actual and random puck trajectories in the original observation. Bottom Left: Pixel-wise L2 error. Bottom Right: Pixel-wise Decoder variance outputs.

When we remove the Decoder variance term, we obtain much more consistent performances across our metrics. The Euclidean variant achieves a low and stable Undisturbed L2 error as shown previously in Fig. 8.35. The diversity performance is robust and significantly improves on the Log-Likelihood variant's achieved diversity and entropy scores in Figs. 8.39 and 8.41.

In a reversal of the findings obtained for our previous set of experiments, the evidence therefore overwhelmingly suggests that the Euclidean variants provide a

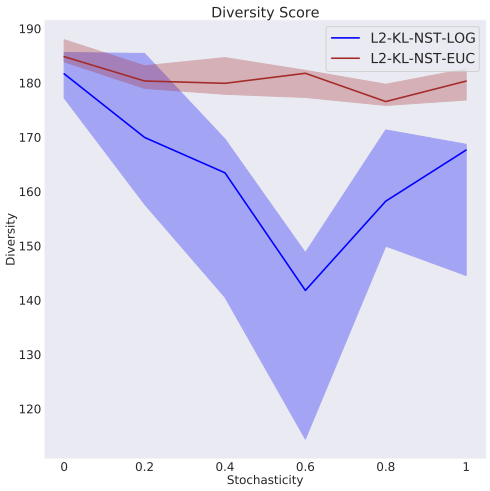


Figure 8.39: The Diversity Score.

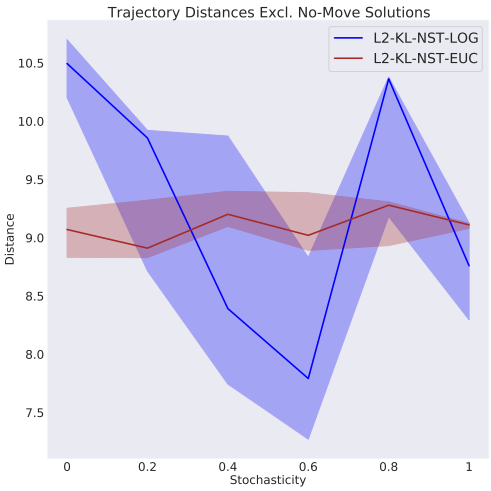


Figure 8.40: The trajectory distances of solutions that move the puck.

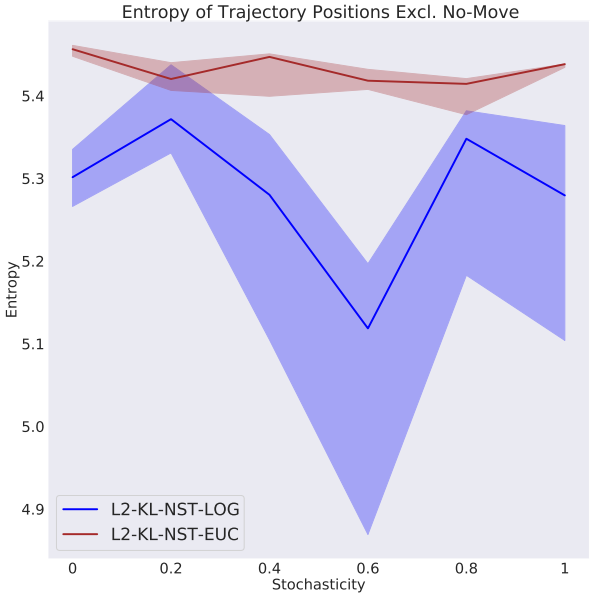


Figure 8.41: The Entropy Score.

stronger and more robust performance than the Log-Likelihood variants when we use images as the environment observations. Furthermore, even if puck trajectories are used as the observation type, the persistent volatility observed in this section can still occur. For instance, a normalisation of the trajectory coordinates can result in L2 errors of smaller magnitude. The limitation of this instability to a noise-free environment in Chapter 7.3.5 is therefore much more likely the exception to the general

applicability of our findings made in this section.

8.3.5 SNE and T-SNE

Both SNE variants produce a large variance in the latent descriptors of the no-move solutions in Fig. 8.42. Typically, no-move solutions are correctly recognised as such and clustered in the BD space. The variance in their descriptors therefore usually serves as an indication of the compactness of the BD space. However, given the SNE variants' large PCTMOVE scores in Fig. 8.43, the increased descriptor variance instead indicates a poor construction accuracy for the no-move solutions. The VAE's constructed images are used as the high-dimensional observations in the similarity calculations performed in the SNE algorithm. A misconstruction of the image therefore directly affects the position of a given solution's BD.

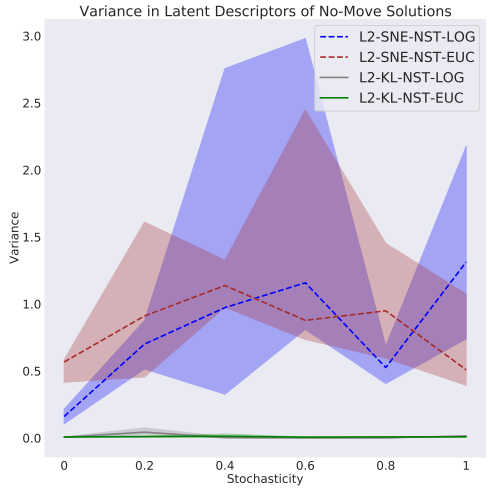


Figure 8.42: The variance in the no-move solutions' latent descriptors.

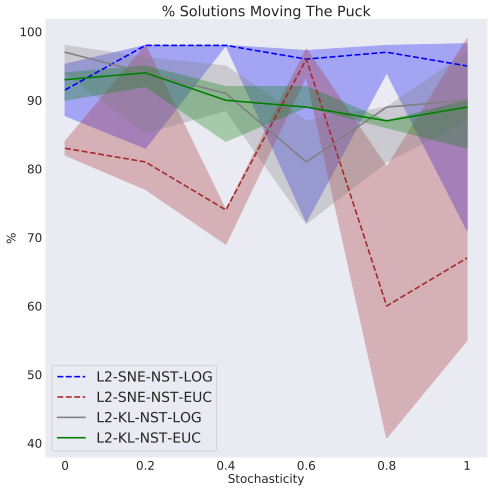


Figure 8.43: The proportion of solutions moving the puck.

We confirm these construction inaccuracies in Fig. 8.44, in which we present the BD space constructed at 100% stochasticity by the L2-SNE-NST-LOG variant. The no-move solutions are located in different areas of the occupied space, leading to a high recorded variance in their descriptors.

The L2-SNE-NST-LOG variant also produces a significantly larger PCTMOVE score at all noise levels in Fig. 8.43 than the EUC variant. As discussed in the previous section, all LOG variants experience a spike in their Decoder variances in the initial training iterations across all levels of stochasticity. These large variance values offset any large L2 errors. Not only does this lead to a slow reduction in the construction error, but it reduces the magnitude of the construction criterion's contribution to the overall gradient. In the LOG variant, the VAE therefore has a larger capacity to reduce the SNE loss further. This increases the criterion's influence.

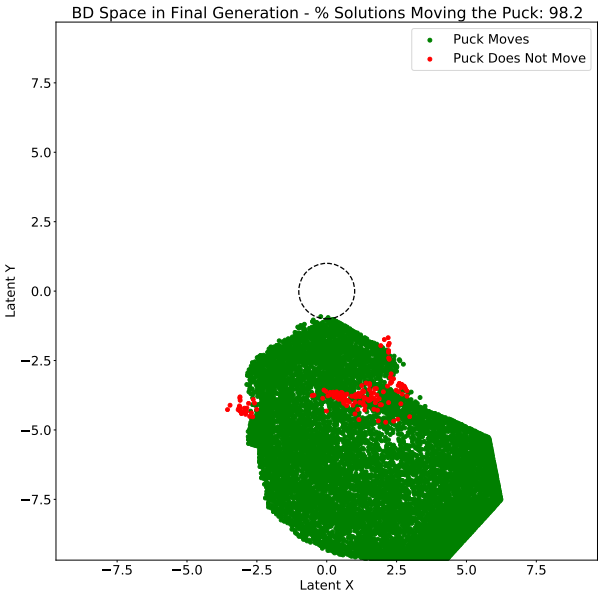


Figure 8.44: BD Space constructed by L2-SNE-NST-LOG at 100% stochasticity.

Figs. 8.45 and 8.46 present the diversity and entropy scores. The SNE variants' negative trend in both scores can again be attributed to their deteriorating construction accuracy as the environment noise increases. Despite a larger PCTMOVE score, the SNE-LOG implementation fails to outperform not only the KL but also the SNE-Euclidean variant in both diversity measures. While the SNE criterion is rather effective at reducing the number of no-move solutions in the archive, these results suggest that it fails to produce a set of diverse solutions that do move the puck.

We arrived at the same conclusion in Chapter 7.3.4, where we showed that a given pair of far-travelling solutions is more likely to be considered dissimilar than an "equivalent" pair of short trajectory solutions. In the previous experiment, this made it more likely for any given long trajectory solution to be accepted and retained in the archive. The resulting change in the characteristics of the solutions retained in the archive led to lower diversity measures.

By replacing the previously used trajectory coordinates with images as our high-dimensional data points, we implicitly changed the SNE algorithm's definition of the similarity between two given trajectories.

Consider the trajectories of three pucks travelling through the room as shown in Fig. 8.47. Most would likely consider a pair consisting of trajectories A and B to be more dissimilar than a pair made up of B and C. When observations consisted of the trajectories' coordinates, the SNE algorithm agreed with this view. However, when we use images any pair of these three trajectories is considered equally dissimilar, as the similarity is calculated as the sum of the squared element-wise pixel differences

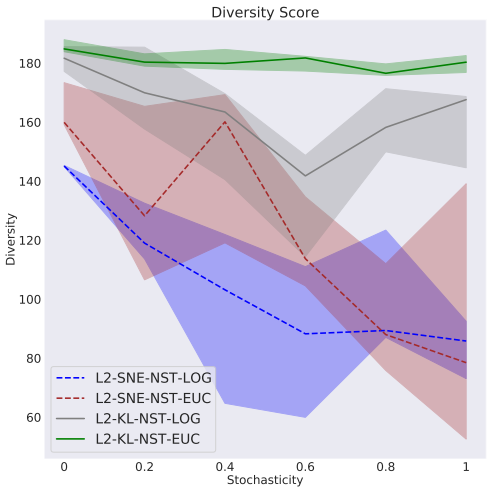


Figure 8.45: The Diversity Score.

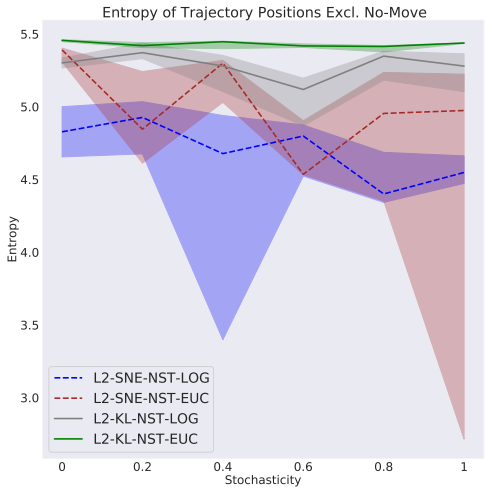


Figure 8.46: The Entropy Score.

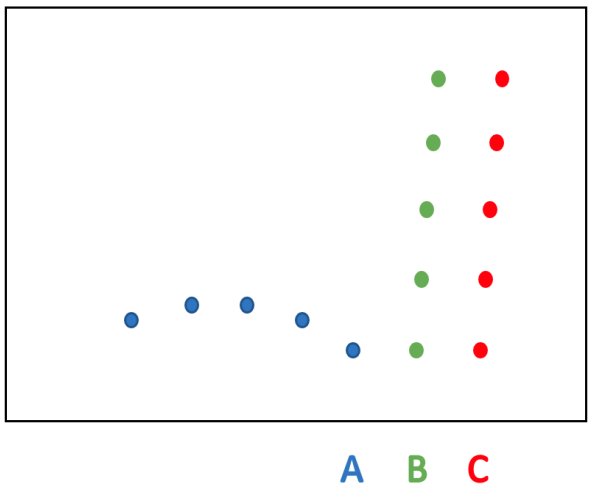


Figure 8.47: Three trajectories observed in the room.

between two images. The similarity of two given solutions now increases in the amount of overlap between their trajectories, even if their shapes and directions are anything but similar.
In our previous experiment, small differences in the initial direction or force applied to the puck could result in the accumulation of large differences towards the end of two long but rather similar trajectories. However in this experiment, a large proportion of the longest of trajectories share the same pixel activations due to a coarse discretisation of the room. The resulting overlap leads to only a small number of the longest solutions to be retained in the archive.

At the other end of the distance spectrum, any given pair of non-overlapping short trajectories is still perceived to be more similar than a non-overlapping pair of long trajectories, since the sum of pixel-wise differences is smaller for the two short tra-

jectories. Shorter solutions are therefore placed closer together in the BD space than longer solutions. Compared to the KL variants this results in our algorithm rejecting a larger proportion of the shortest solutions.

The consequences of imposing the SNE criterion can be seen quite clearly in the constructed BD spaces. Fig. 8.48 shows a BD space constructed by the L2-SNE-NST variant. We colour-code the descriptors according to the distances travelled by their associated actual pucks. The shortest solutions need to be placed in the center of the occupied space due to their large perceived similarity to each other and subsets of the other solutions with slightly longer trajectories. Moving away from this central cluster increases the distance of the trajectory at the same rate in every direction as we can see from the concentricity of the same-coloured rings. Non-overlapping long solutions are most dissimilar from each other. They can therefore be placed in the outermost ring of the BD space.

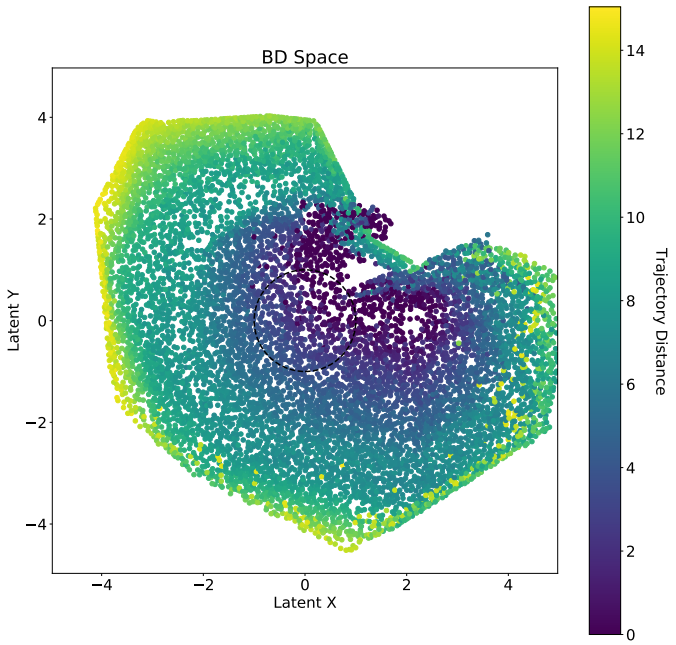


Figure 8.48: BD Space constructed by L2-SNE-NST-LOG at 100% stochasticity. Colour bar indicates the distance travelled by the trajectory.

With reductions to the proportions of both the longest and the shortest solutions, the median distance of trajectories in Fig. 8.49 does not differ significantly between the KL and SNE variants, but the latter exhibit a much smaller variance in these distance values in Fig. 8.50. This indicates an increased similarity between solutions in the SNE variants' archives. They therefore achieve a much lower diversity and entropy

score than the KL variants, as illustrated previously in Figs. 8.45 and 8.46. The SNE Log-Likelihood implementation produces the worst diversity performance measures out of all variants, and, in particular, lower scores than its Euclidean counterpart. This is due to the comparatively larger influence of the SNE criterion in the LOG implementation that also resulted in larger PCTMOVE score.

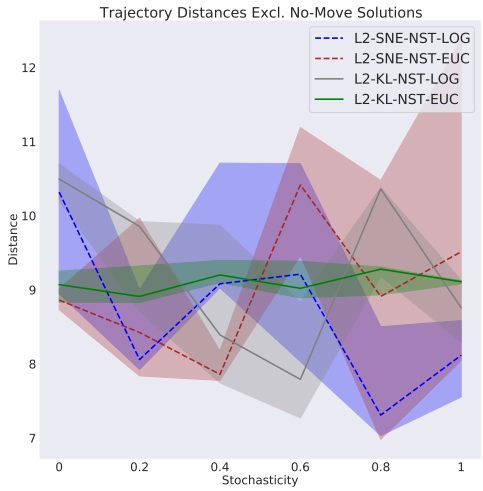


Figure 8.49: The trajectory distances of solutions that move the puck.

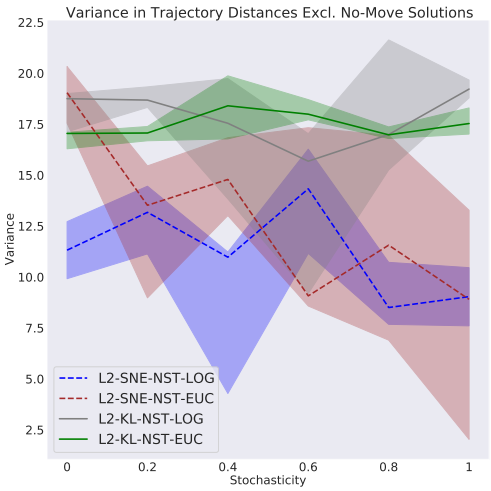


Figure 8.50: The variance in the trajectory distances of solutions that move the puck.

By replacing the Gaussian distribution applied to the latent representations with a Student t-distribution, the t-SNE algorithm alleviates what is described as the SNE's crowding problem. The heavier tails of the Student t-distribution allow the representations of a given pair of high-dimensional observations to be placed further apart than under the SNE criterion. The t-SNE criterion therefore exerts a weaker influence on the compactness of the BD space.
This leads to both t-SNE variants producing a lower PCTMOVE score than their SNE counterparts in Fig. 8.51. The t-SNE Log-Likelihood variant produces a larger score than the Euclidean implementation. This is again due to the presence of the Decoder variance term.

Conversely, the t-SNE's reduced influence also implies that any given pair among the shortest and longest trajectories is less likely to be rejected from the archive when we replace the SNE with a t-SNE criterion. The result is a reversal of the previously observed changes in the composition of the SNE variants' archives. The t-SNE variants produce a larger proportion of shorter and longer solutions. We show this in a larger variance in the trajectory distances in Fig. 8.52.

The consequences of the reduced pressure on the BD positions can also be seen when we visualise the latent spaces. None of the t-SNE BD spaces we inspected, beared any similarity in their shapes and arrangement of solutions to those constructed by

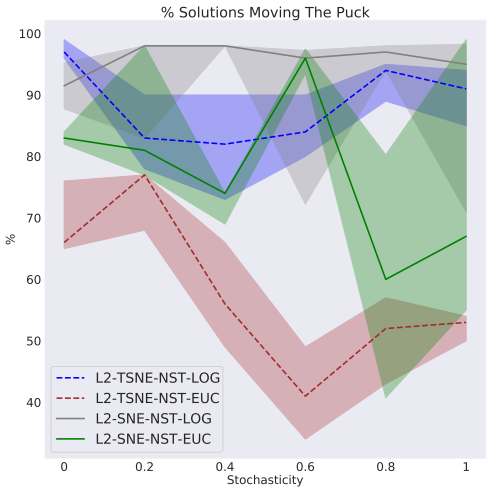


Figure 8.51: The proportion of solutions moving the puck.

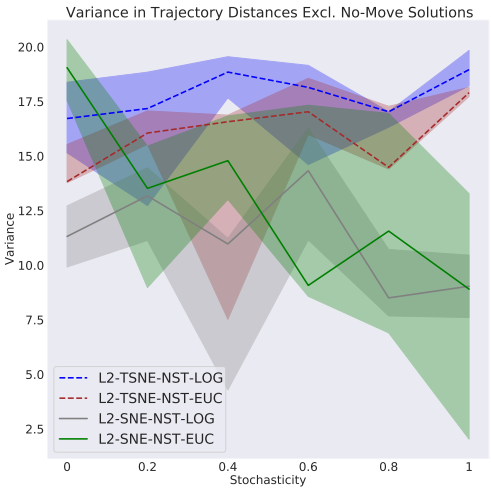


Figure 8.52: The variance in the trajectory distances of solutions that move the puck.

the SNE variant. Instead they match our KL variants' gradual progression in distances from one end of the occupied space to the other. We illustrate this in Fig. 8.53 for a BD space constructed by the L2-TSNE-NST-LOG variant.

The TSNE implementations therefore achieve a much more competitive diversity score in Fig. 8.54 than their SNE counterparts. The L2-TSNE-NST-LOG variant in particular benefits from a larger PCTMOVE score as there is no longer a corresponding decrease in the proportion of the longest and shortest solutions in the archive.

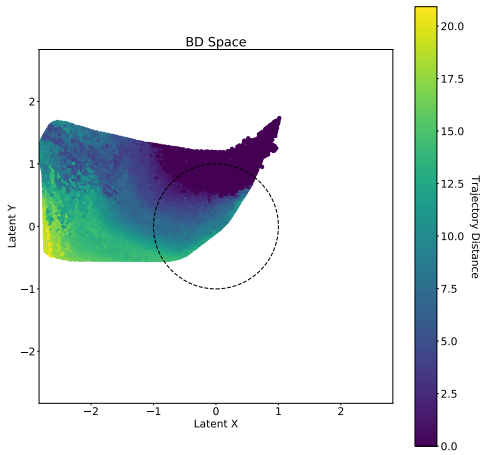


Figure 8.53: BD Space constructed by L2-TSNE-NST-LOG at 100% stochasticity. Colour bar indicates the distance travelled by the trajectory.

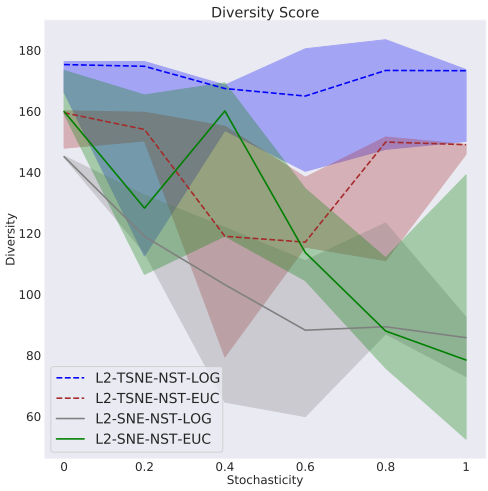


Figure 8.54: The Diversity Score.

Adding the KL criterion to our t-SNE variants produces little change across our metrics. Similar to our findings in Chapter 7.3.4, we can see that the contribution of the KL criterion is largely restricted to moving the occupied descriptor space as a whole closer to the origin. We illustrate this in two BD spaces constructed by the LOG variant with and without the added KL criterion in Figs. 8.55 and 8.56 respectively. This indicates that the KL criterion is only afforded little attention in the network's optimisation process due to the t-SNE's strong influence.
With the density of the BD spaces left unchanged, the PCTMOVE scores in Fig. 8.57 are left largely unaffected.
The addition of the KL criterion therefore does not yield any changes to the t-SNE's diversity metrics. Moreover, the t-SNE variants fail to produce an improvement on our L2-KL-NST-EUC implementation, which remains our best performing architecture. We illustrate this in the diversity and entropy scores in Figs. 8.58 and 8.59.

By design, the SNE criterion exerts an even stronger influence on the network's optimisation than the t-SNE criterion. Adding the KL divergence term to the SNE variants therefore also does not produce any significant changes to our metrics. The PCTMOVE and entropy scores in Figs. 8.60 and 8.61 are left unaffected. Despite a larger PCTMOVE score for the LOG variants, all SNE variants' entropy scores remain lower than the scores produced by the L2-KL-NST-EUC variant.

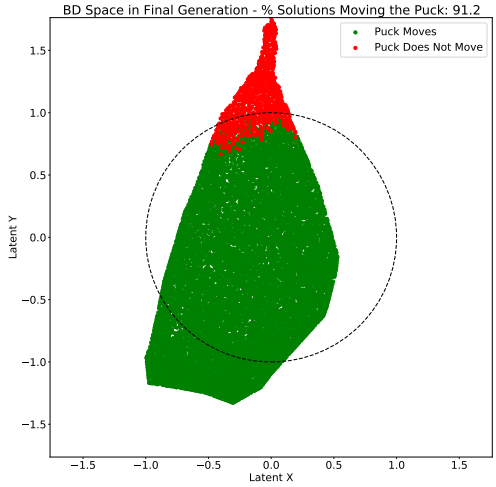


Figure 8.55: BD Space constructed by L2-KL-TSNE-NST-LOG at 100% stochasticity.

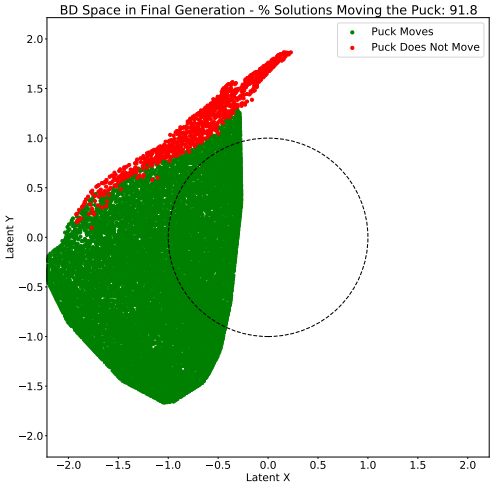


Figure 8.56: BD Space constructed by L2-TSNE-NST-LOG at 100% stochasticity.

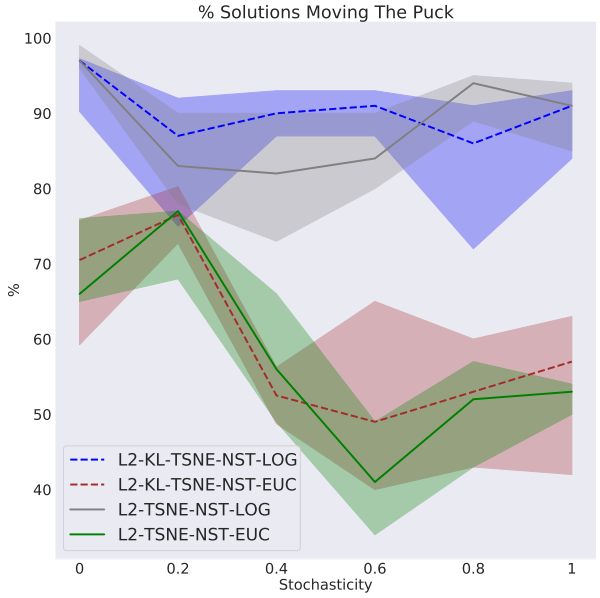


Figure 8.57: The proportion of solutions moving the puck.

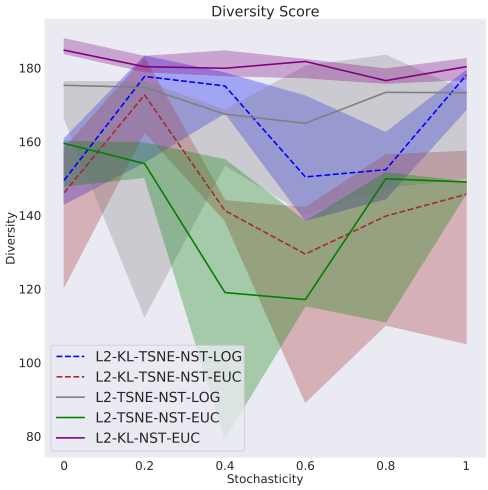


Figure 8.58: The Diversity Score.

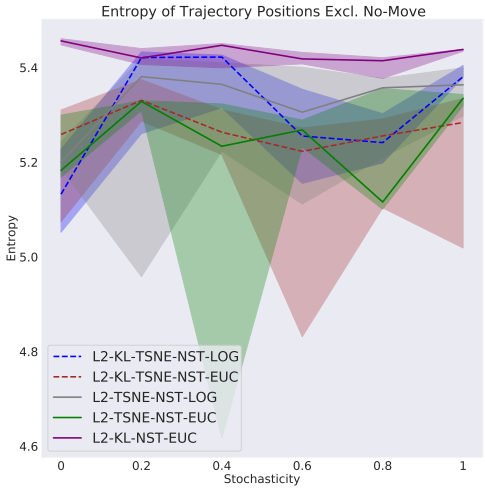


Figure 8.59: The Entropy Score.

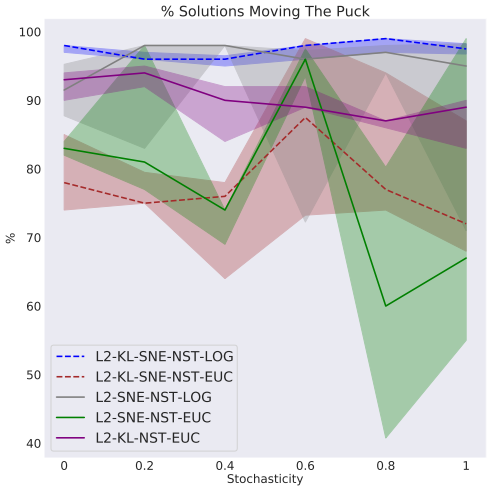


Figure 8.60: The proportion of solutions moving the puck.

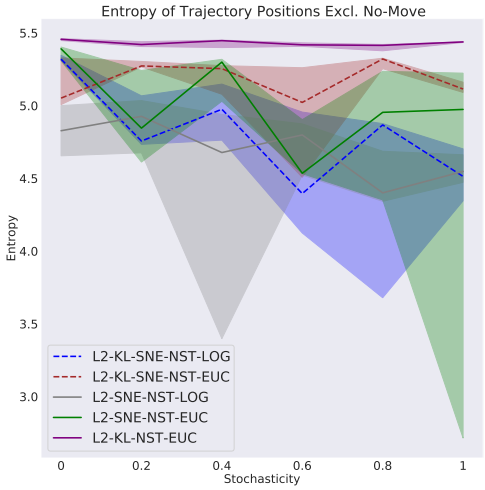


Figure 8.61: The Entropy Score.

8.3.6 Distance Measure In Loss

The L1-KL-NST-EUC implementation experiences a well-documented phenomenon which works in the literature have referred to as optimisation challenges of VAEs [51], the information preference property [52] and more recently, a posterior collapse [53]. When the latent descriptions formed by the Encoder network do not carry sufficient levels of information for an extended period, the Decoder network starts to treat them as noise. It disregards the inputs it receives and produces the same construction for all latent codes. This construction resembles the average of all the labels, in order to reduce the construction loss. When the latent descriptions are not used by the Decoder, any changes to their positions do not affect the construction loss. Consequently, the Encoder can now fully minimise the KL divergence loss. Using the notation from Eq. 2.1, we refer to this as the collapse of the posterior $q_\varphi(z|x)$ into the prior $p_\theta(z)$.

Since we use a standard Gaussian distribution as the prior, we expect the BD space constructed to accurately approximate such a distribution when the posterior is fully collapsed. However, since our implementation does not sample from the Encoder, the minimisation of the KL divergence loss instead results in all BDs being placed very close to the origin of the space. We show the minimal KL loss achieved and a visualisation of the constructed BD space at 0% stochasticity in Figs. 8.62 and 8.63. Note, the magnitude of the coordinates in Fig. 8.63.

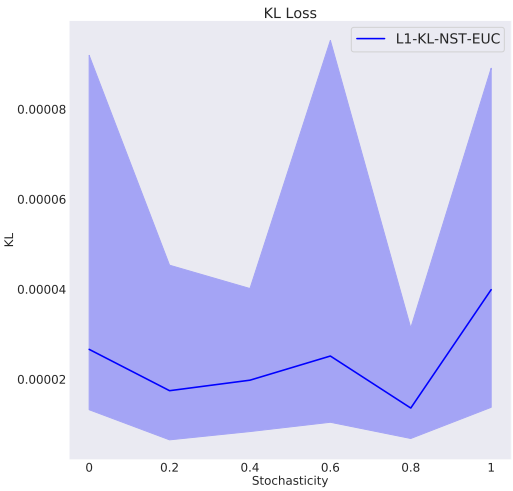


Figure 8.62: The KL Loss.

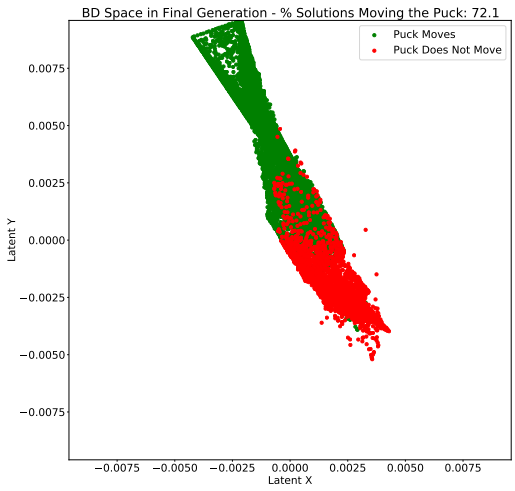


Figure 8.63: BD Space constructed by L1-KL-NST-EUC at 0% stochasticity.

We confirm the posterior collapse further by finding that the network makes the same image construction for every single solution. As the visualisation is rather anti-climactic we do not include it here.

Unsurprisingly, the produced performance measures are poor. Fig. 8.64 shows a sub-

stantially decreased PCTMOVE score and Fig. 8.65 shows a significantly lower and more volatile diversity score.

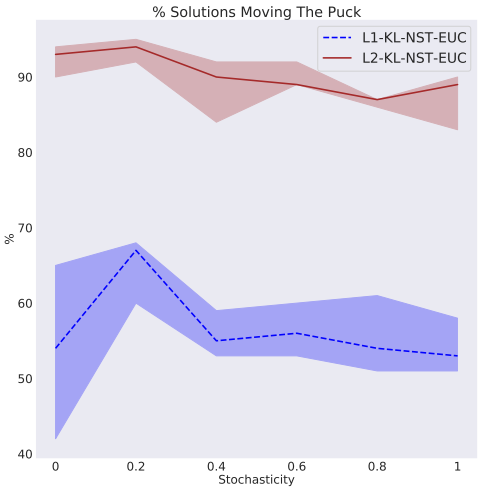


Figure 8.64: The proportion of solutions moving the puck.

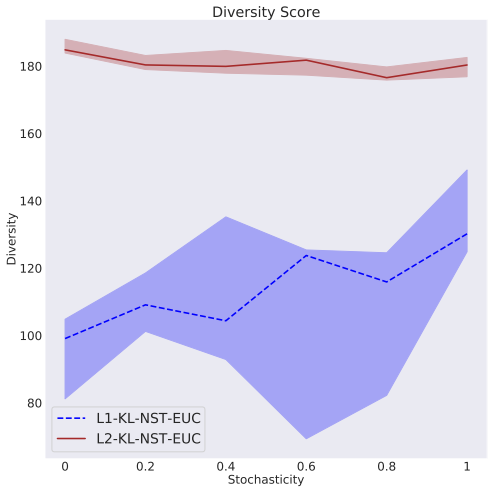


Figure 8.65: The Diversity Score.

We are uncertain as to the underlying causes leading to this posterior collapse. By replacing the L2 criterion with an L1 metric, we weaken the construction loss' gradient signal for large losses. Compared to the previous experiment, this is further exacerbated by our use of a deeper Decoder network which increases the likelihood of vanishing gradients. This could imply that the information content in the Encoder means increases too slowly in the initial generations and the Decoder therefore starts ignoring them. However, the VAE does not experience a posterior collapse when we replace the L2 metric with a Huber Loss, despite its gradient being of smaller magnitude than the L1's.

The Huber Loss is the more general formulation of the Smooth L1 criterion we tested in Chapter 7.3.6. While the Smooth L1 criterion starts applying an L2 metric below an absolute error of 1, the Huber Loss leaves the definition of the threshold to the user. Since images can be represented as collections of pixel-wise activations ranging from 0 and 1, the natural choice to provide some additional noise discrimination ability is a threshold of 0.5.

Fig. 8.66 shows that the Huber variant indeed achieves an improvement in the Undisturbed L2 error for larger levels of stochasticity. This indicates an enhanced noise discrimination ability. However, we noted in our previous discussion on the impact of different distance metrics in Chapter 7.3.6, that in isolation, small gains in construction accuracy are rather inconsequential to the BD accuracy and the achieved diversity performance. The more important result therefore is presented in Fig. 8.67,

which illustrates that the Huber variant also achieves a significantly lower KL divergence loss than the L2 implementation.

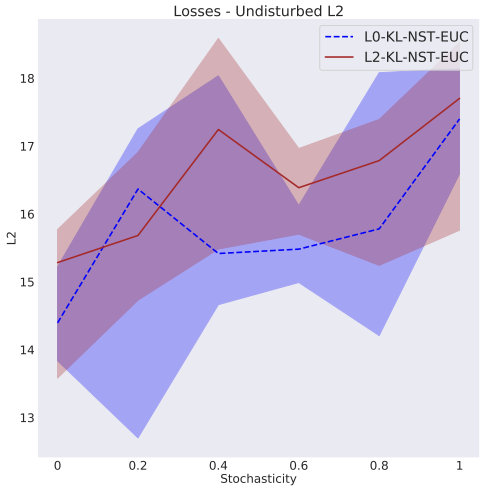


Figure 8.66: The Undisturbed L2 Error.

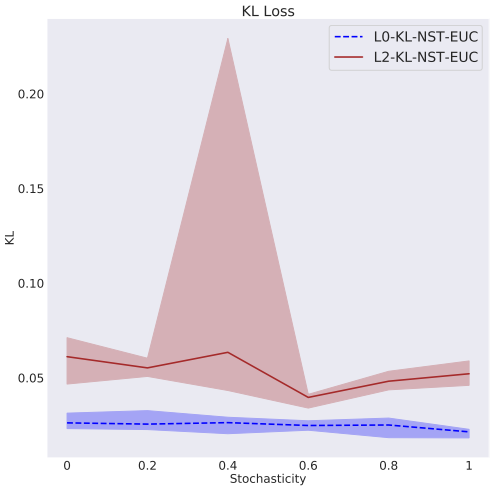


Figure 8.67: The KL Loss.

This is due to the Huber Loss producing a gradient of a lower magnitude than the L2 metric. We illustrate the gradients over output activations ranging from 0 to 1, assuming a label value of 1 in Fig. 8.68.

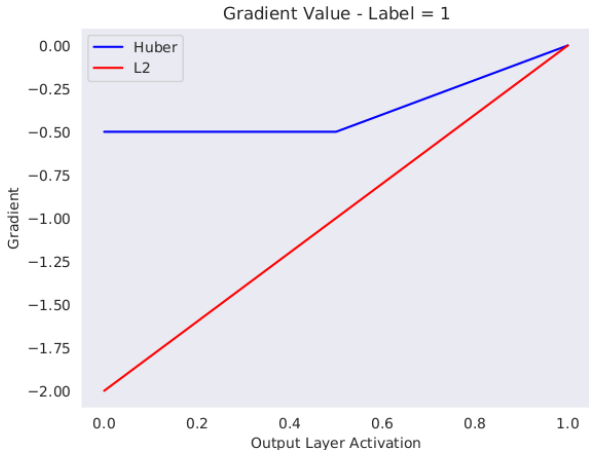


Figure 8.68: Values taken by the gradient of the construction loss with respect to the output layer activations assuming a label equal to 1.

Additionally, unlike our findings made on the L1 criterion in Chapter 7.3.6, the total L2 loss remains largely unchanged in Fig. 8.69.
Consequently, the application of the Huber Loss results in an overall smaller magnitude of the gradient of the construction loss. The network can therefore dedicate a larger capacity to reducing the KL loss. This increases the compactness of the BD space as indicated by a lower variance in the latent descriptors of no-move solutions in Fig. 8.70 and a larger PCTMOVE score in Fig. 8.71.

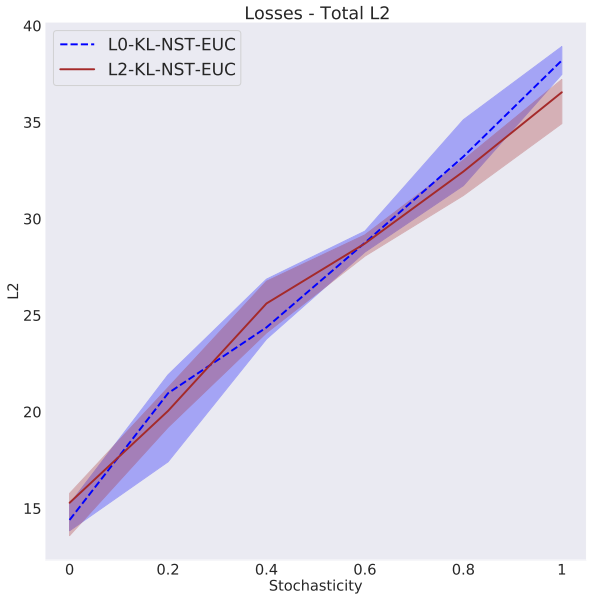


Figure 8.69: The Total L2 Error.

The Huber variant's robustness to noise and the L2 variant's lack thereof is especially noticeable at larger levels of stochasticities when the performances start diverging in both illustrations. At lower stochasticities, the L2 metric's larger gradient magnitude results in a faster convergence. The L2 variant therefore produces a larger PCTMOVE score when the noise levels are low. However, we can compensate for the Huber variant's slower convergence by increasing the number of generations or training iterations.

Finally, we present a comparison of the achieved diversity and POSVAR scores in Figs. 8.72 and 8.73. In these metrics, the Huber variant again shows an improved performance, albeit only marginally. Nonetheless, even if we considered both variants' scores to be equal, the lower number of no-move solutions and the increased construction accuracy make the Huber variant the more appropriate choice in our search for an algorithm that is robust to noise.

Finally, we consider an implementation with a sigmoid activation in the output layer and a binary cross-entropy loss criterion. In Fig. 8.74, we present the PCTMOVE scores. The Sigmoid implementation achieves a lower score than our Huber variant across all stochasticities indicating a weaker compression of the BD space. Indeed, the KL loss is significantly larger in Fig. 8.75. This is again the result of a difference in the magnitude of the gradients. The combination of a sigmoid activation and a binary cross-entropy loss results in a linear gradient of the construction criterion with respect to the output layer's activations. For the Huber Loss, the gradient is of smaller magnitude for any given error. We show a visualisation of the two gradients

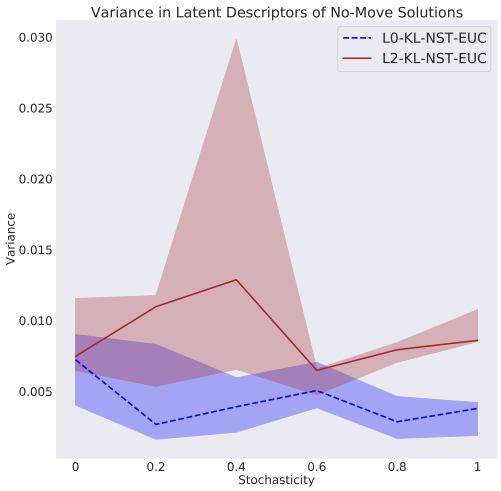


Figure 8.70: The variance in the no-move solutions' latent descriptors.

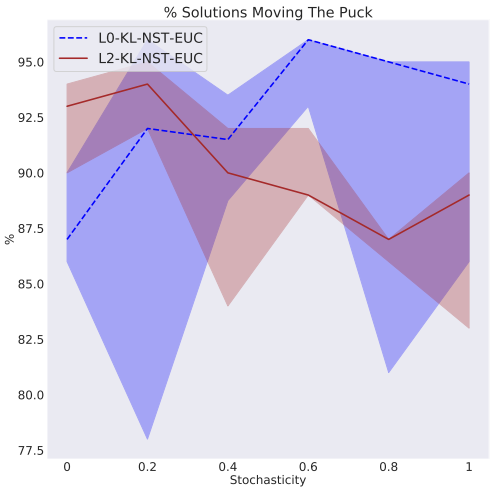


Figure 8.71: The proportion of solutions moving the puck.

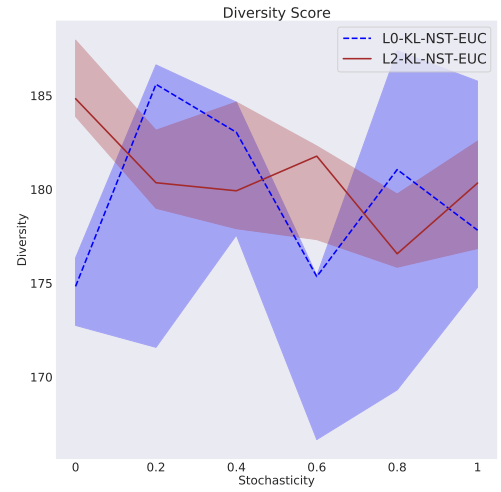


Figure 8.72: The Diversity Score.

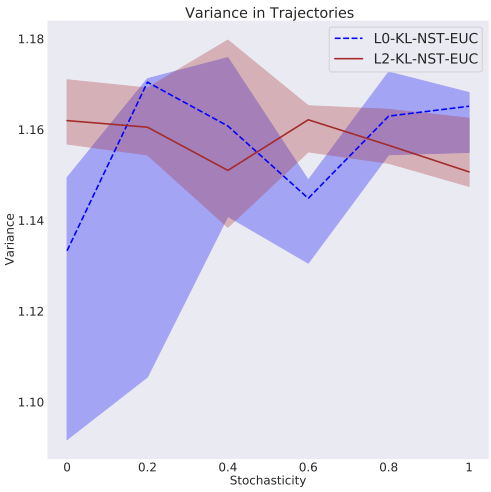


Figure 8.73: The variance in the puck trajectories.

over output activations ranging from 0 to 1, assuming a label value of 1 in Fig. 8.76.

The Huber variant is a its larger PCTMOVE score into a superior diversity score in Fig. 8.77 and therefore remains our best architecture.

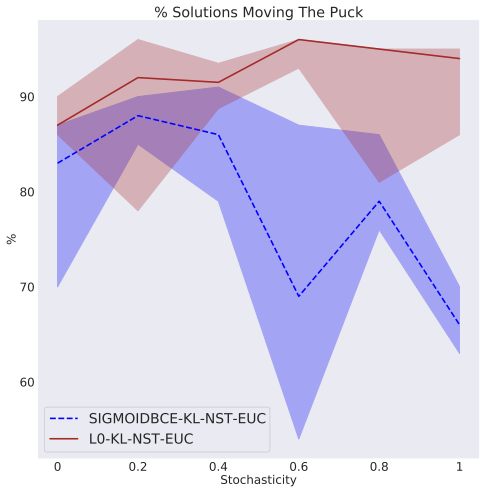


Figure 8.74: The proportion of solutions moving the puck.

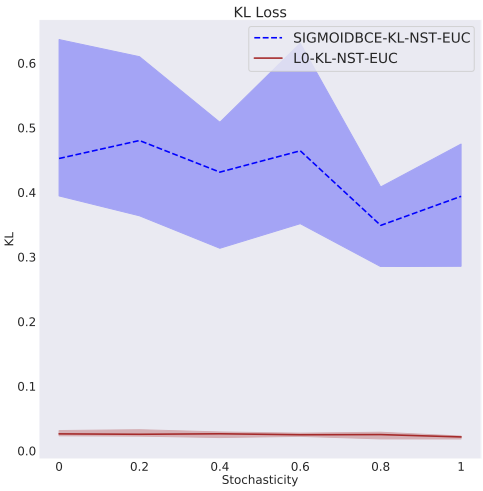


Figure 8.75: The KL Loss.

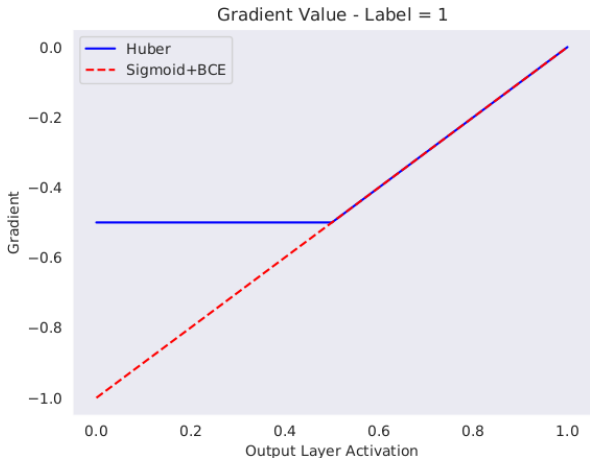


Figure 8.76: Values taken by the gradient of the construction loss with respect to the output layer activations assuming a label equal to 1.

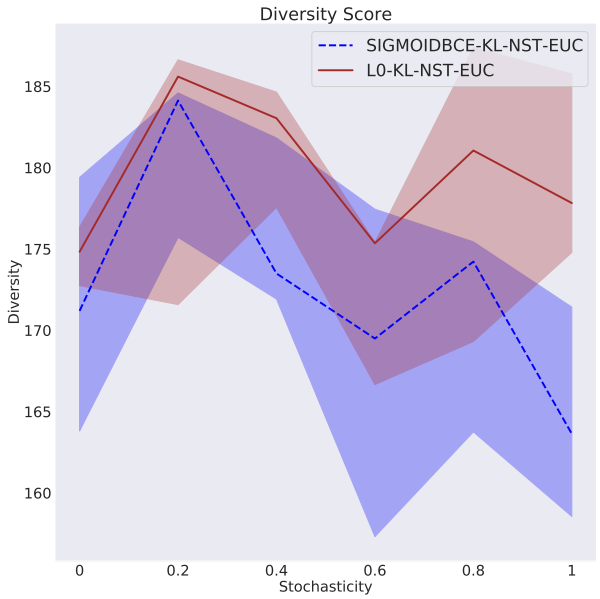


Figure 8.77: The Diversity Score.

8.3.7 Analysing the Best Architecture

The bulk of the L0-KL-NST-EUC variant's final performance scores are achieved again after only 500 generations. The metrics largely stabilise after 1000 generations but marginal increases throughout the remaining generations can still be observed. We present the progression in the POSVAR and PCTMOVE metrics in Figs. 8.78 and 8.79 at 0% and 100% stochasticity respectively.

Figure 8.78: Top: POSVAR score. Bottom: PCTMOVE score. L0-KL-NST-EUC at 0% stochasticity.

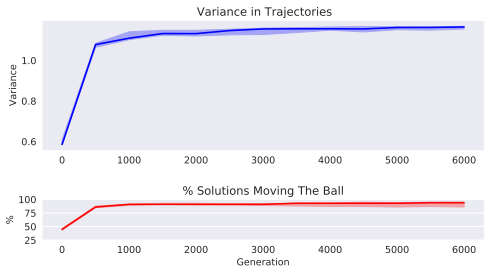


Figure 8.79: Top: POSVAR score. Bottom: PCTMOVE score. L0-KL-NST-EUC at 100% stochasticity.

The progression in the construction accuracy matches the same consistency. We show the Undisturbed L2 errors at 0% and 100% stochasticity in Figs. 8.80 and 8.81. The errors decrease sharply in the initial generations when the archive is still small and relatively less diverse. The following rapid increase in the error reflects the equally sharp increase in diversity and archive size, until losses finally stabilise around the 1000 generation mark again.

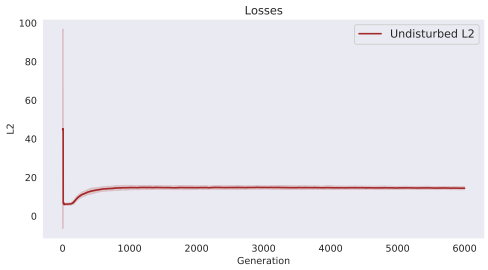


Figure 8.80: The Undisturbed L2 Error. L0-KL-NST-EUC at 0% stochasticity.

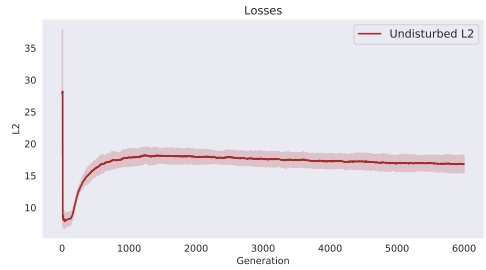


Figure 8.81: The Undisturbed L2 Error. L0-KL-NST-EUC at 100% stochasticity.

The shape of the constructed BD spaces remains stable too. We present the progression of two BD spaces over 6000 generations at 0% and 100% stochasticity in Figs. 8.82 and 8.83.

Fig 8.84 presents a visualisation of all trajectories in the generated archives of solutions across 4 repetitions at 100% stochasticity. The stability of the algorithm's performance leads to the generation of very similar archives in each repetition.

Figure 8.82: Progression of BD space constructed by L0-KL-NST-EUC at 0% stochasticity over 6000 generations. Snapshots taken in steps of 500 generations.

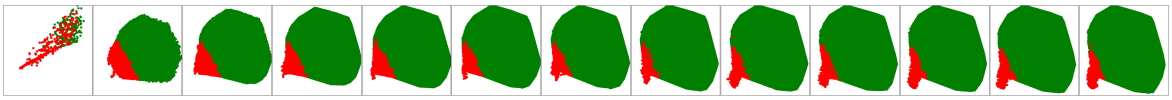


Figure 8.83: Progression of BD space constructed by L0-KL-NST-EUC at 100% stochasticity over 6000 generations. Snapshots taken in steps of 500 generations.

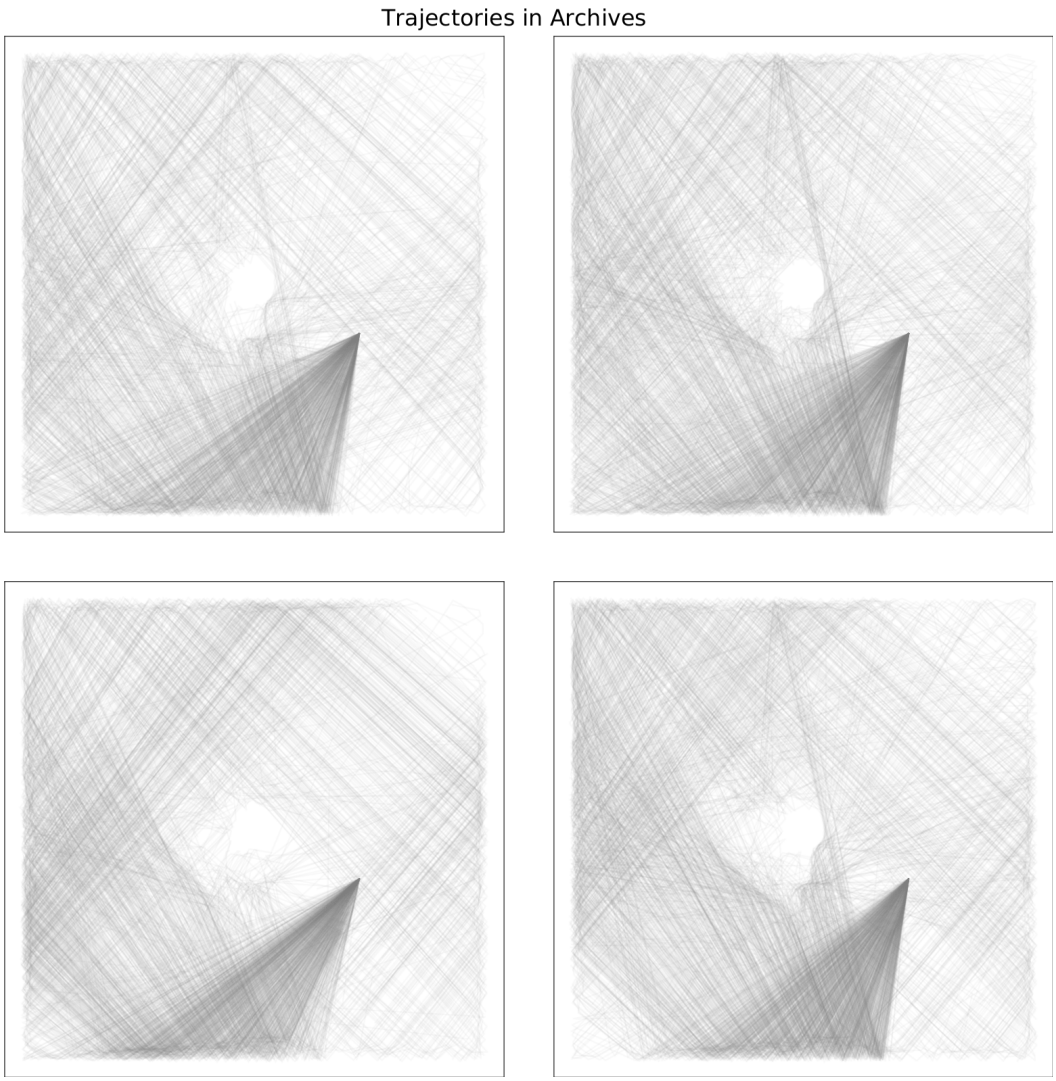


Figure 8.84: Visualisation of the actual trajectories of all solutions in 4 archives generated by L0-KL-NST-EUC at 100% stochasticity.

We conclude this section with a determination of whether our best performing architecture produces an archive that is suitable for the selection, mutation and cross-

over operations employed in the AURORA algorithm. To do so, we compare the L0-KL-NST-EUC variant's performance against an implementation that replicates its architecture but generates new solutions by sampling randomly in the solution space.

We confirm that a utilisation of these QD operators indeed results in larger PCT-MOVE, diversity and POSVAR scores in Figs. 8.85, 8.86 and 8.87 respectively.

Figure 8.85: The proportion of solutions moving the puck.

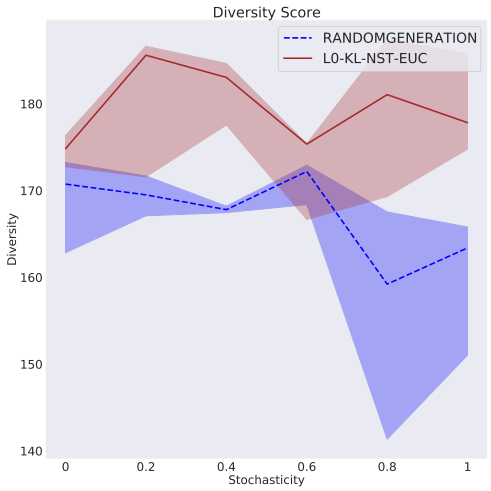


Figure 8.86: The Diversity Score.

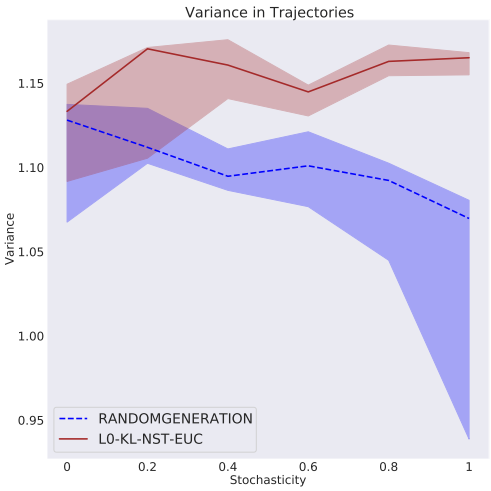


Figure 8.87: The variance in the puck trajectories.

8.3.8 Evaluation against Benchmarks and Baselines

In this section, we evaluate the same benchmark architectures that we introduced in Chapter 7.3.8.

In benchmark EXCLUDE_TRAIN we remove the solutions with the 30% largest reconstruction errors from the training dataset. In EXCLUDE_ARCHIVE we remove them from the archive altogether.

In Fig. 8.88, we present the PCTMOVE scores produced by the AURORA implementation and these two benchmarks.
The results obtained for the EXCLUDE_ARCHIVE benchmark mirror its performance in the previous set of experiments. This can be seen across all our metrics and is due to all the same reasons. We therefore refer to our detailed analysis made in Chapter 7.3.8 and omit this benchmark from our discussions in this section.

The EXCLUDE_TRAIN benchmark variant produces a similar number of no-move solutions as the AURORA algorithm. The diversity performance is largely similar but suggests some additional robustness to noise at larger stochasticity levels. We show a more stable POSVAR score in Fig. 8.89. We can also see indications of an improved diversity score in Fig. 8.90. However, the performance in both metrics is clearly still influenced by the environment noise and therefore does not provide the desired robustness.

The EXTEND benchmark pursues a similar idea as the EXCLUDE_TRAIN implementation. Both benchmarks try to lower the proportion of noisy solutions in the dataset. However, while EXCLUDE_TRAIN simply removes them from the training dataset,

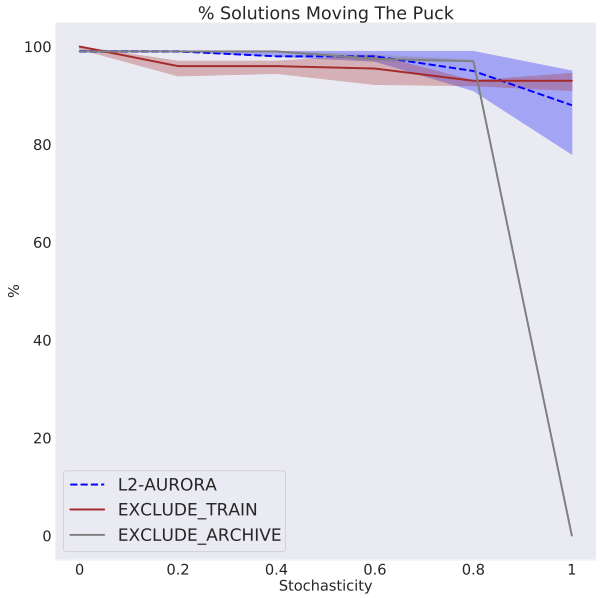


Figure 8.88: The proportion of solutions moving the puck.

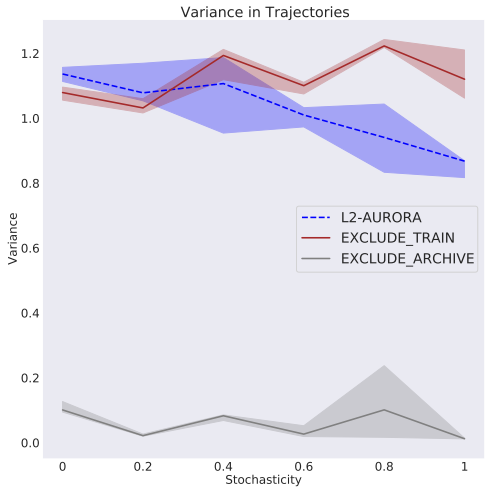


Figure 8.89: The variance in the puck trajectories.

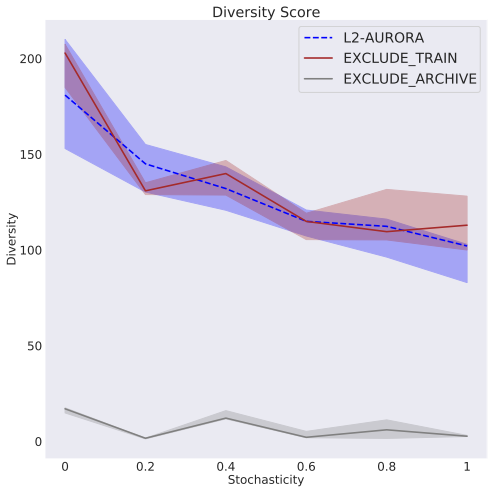


Figure 8.90: The Diversity Score.

EXTEND instead regenerates the environment observations for these solutions and adds these to the training dataset. As discussed in Chapter 7.3.8, this leads to a larger number of samples overall but in particular a larger number of noisy samples. This produced a worse performance in the previous set of experiments already, but the difference between these two approaches is much clearer in this set of experiments. Fig. 8.91 shows the EXTEND implementation's PCTMOVE score that is significantly

affected by the noise in the environment, and, in particular, much more so than for the AURORA implementation. As discussed in Chapter 8.3, the AURORA algorithm

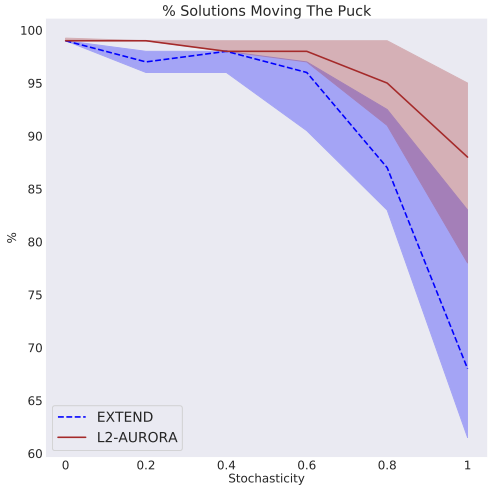


Figure 8.91: The proportion of solutions moving the puck.

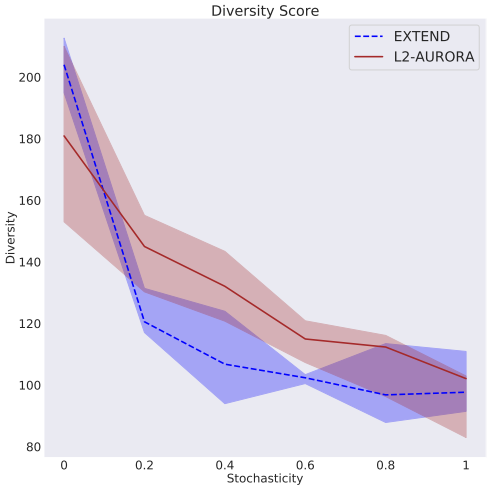


Figure 8.92: The Diversity Score.

produces a very large PCTMOVE score in this experiment due to the limited variability in the achievable puck trajectories. Only as the stochasticity increases to large levels, are there sufficiently many random trajectories to significantly impact the performance. However, whenever the environment is not noise-free, the EXTEND benchmark now adds additional random trajectories to the dataset. The result is the accelerated deterioration of the PCTMOVE score in the environment noise in Fig. 8.91. As expected, the diversity score in Fig. 8.92 reflects this increased susceptibility to noise.

At 0% environment noise the EXTEND benchmark shows an indication of a larger diversity score, albeit not significantly. This suggests a potential benefit in having additional training samples. We explore the impact of adding samples to the dataset of our architecture in the following section. Note however, that a substantial portion of this performance difference could arise from a large variability in the results that we can observe for the AURORA algorithm in Fig. 8.92. We discuss this volatility further along in this Chapter.

In the REGEN benchmark we also regenerate the observations but use them to replace the selected solutions' previously associated observations. We observe the same behaviour as in Chapter 7.3.8. By choosing the solutions with the largest loss, we bias our selection to solutions with longer trajectories. These solutions are regenerated repeatedly until the loss incurred on their construction decreases. This tends to happen in one of two cases. The regenerated random trajectory either resembles a frequently observed pattern of pixel activations and the network therefore produces a good construction, or the random trajectory is eliminated altogether. In both cases,

a reduction in the incurred loss is the result of a similarity between the regenerated solutions' observations and existing solutions' observations. This makes a rejection of these long-distance solutions from the archive very likely. Fig. 8.93 shows the large decrease in the distance travelled by the puck in the REGEN benchmark. This leads to the decreased diversity score shown in Fig. 8.94.

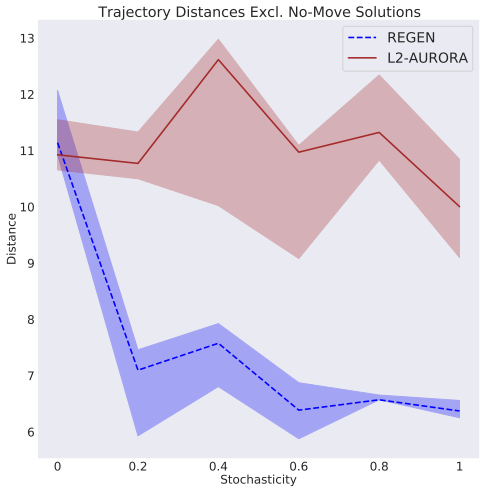


Figure 8.93: The trajectory distances of solutions that move the puck.

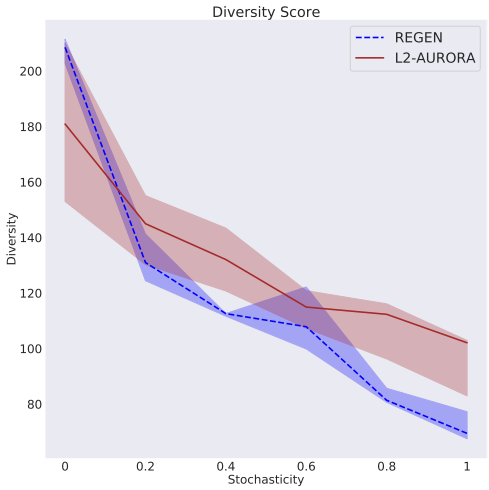


Figure 8.94: The Diversity Score.

Interestingly, at 0% stochasticity there is a large difference between the scores achieved by the AURORA baseline and the REGEN benchmark. This is not caused by the regeneration mechanism, as the regeneration of observations does not produce any changes when the environment is noise-free. Instead, the difference in the score suggests that repeated executions of the AURORA algorithm can produce very different archives. In Fig. 8.95, we visualise the trajectories in an archive created by a typical run of the AURORA algorithm at 0% stochasticity. In contrast, in Fig. 8.96, we show an archive containing a very different set of trajectories. In creating the latter, the robot bounces the puck off the robot arm in the center of the room. This creates dissimilar solutions. The diversity score is much lower due to trajectories travelling a shorter distance. Nonetheless, this creates very interesting behaviours. Our proposed architecture variants are not able to replicate this consistently as the collisions with the arm are largely ignored in the constructions.

While the EXCLUDE_ARCHIVE, REGEN and EXTEND benchmarks fail to match the performance of the AURORA baseline, the EXCLUDE_TRAIN implementation produces a marginal improvement to the achived diversity. Nonetheless, the AURORA algorithm's susceptibility to noise persists in this benchmark. The EXCLUDE_TRAIN's produced solutions achieve a better diversity score in Fig. 8.97 than a randomly sampled set of solutions. However, when we consider only the solutions that move the puck, we find that the archive generated by FULLRANDOM produces a larger en-

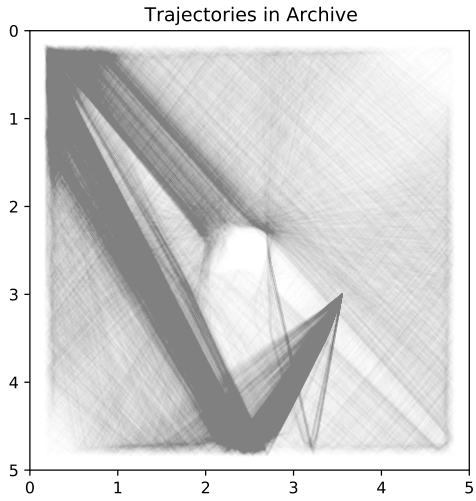


Figure 8.95: Visualisation of the actual trajectories of all solutions generated in a typical run of L2-AURORA at 0% stochasticity.

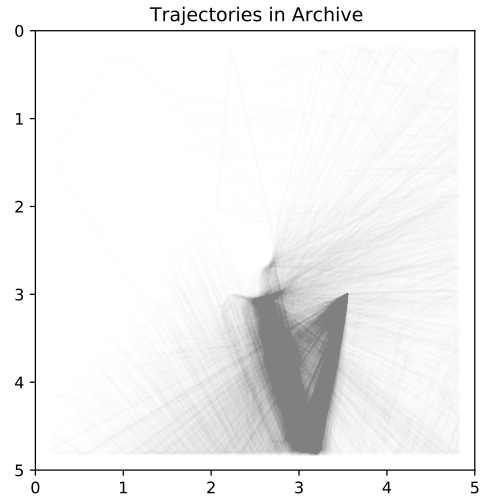


Figure 8.96: Visualisation of the actual trajectories of all solutions generated in an atypical run of L2-AURORA at 0% stochasticity.

tropy score in Fig. 8.98 at the largest stochasticity levels. The performance of both baselines is significantly inferior to that of our best performing architecture when the environment is stochastic. Nevertheless, EXCLUDE_TRAIN, and by extension the AURORA baseline, remains the best performing architecture in a noise-free environment. This is the result of a larger BD accuracy obtained by extracting the descriptors directly from the environment observations.

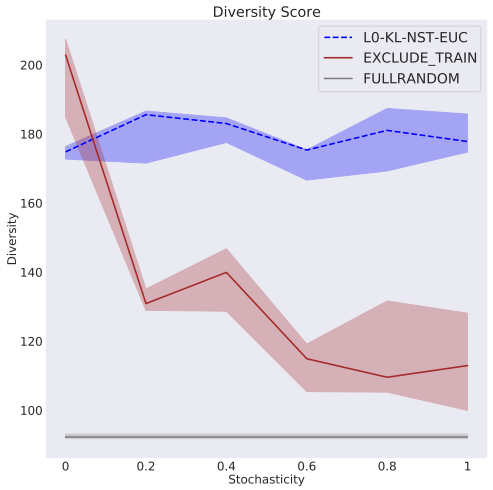


Figure 8.97: The Diversity Score.

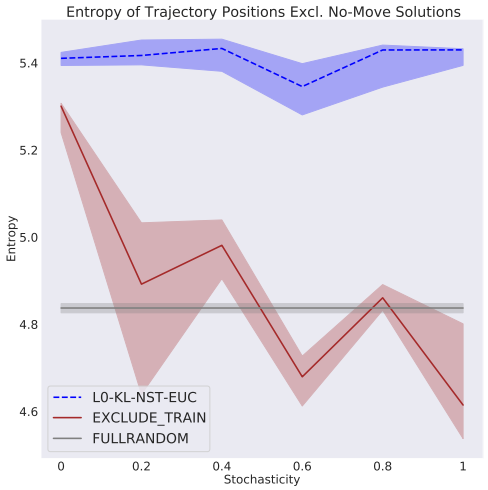


Figure 8.98: The Entropy Score.

8.3.9 Improving the Performance

In this section, we evaluate the performance of our best architecture with a batch size of 1024 and 5 additional training archives. As discussed in Chapter 7.3.9, by increasing the batch size, we attempt to reduce the influence of the noisy samples on the overall batch gradient. By adding additional training archives, we increase the size of the training dataset while retaining its balanced composition.

The results indicate that these additions lead to a more stable performance that is less impacted by the environment noise. Fig. 8.99 illustrates that the adapted variant has a lower KL loss. As we have seen in our previous analysis, this is likely the consequence of an improved construction performance that allows for a larger focus on the KL divergence criterion in the network optimisation. The improved BD accuracy decreases the variance of latent descriptors of no-move solutions further, as shown in Fig. 8.100.

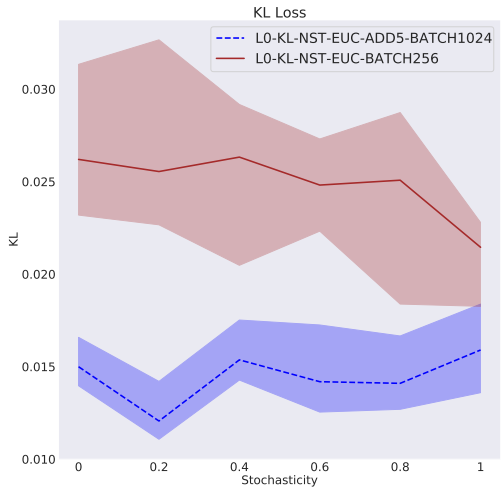


Figure 8.99: The KL Loss.

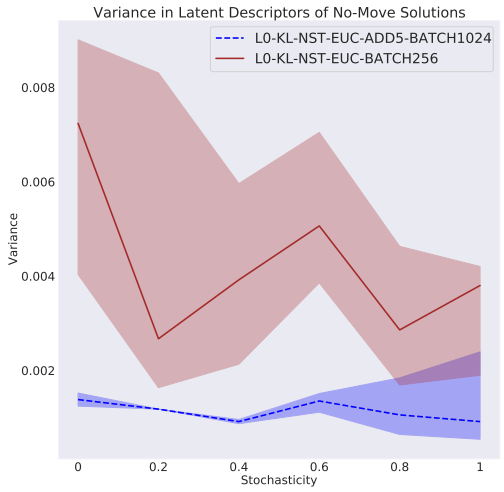


Figure 8.100: The variance in the no-move solutions' latent descriptors.

Despite these increases being rather marginal in absolute terms, the PCTMOVE metric in Fig. 8.101 does benefit. We can observe a significantly improved score when the environment noise levels are low, and an increased stability across stochasticities. The performance gains made at low stochasticities are reflected in our diversity measures. We show this in the POSVAR score in Fig. 8.102.
The addition of additional training archives and a larger batch size therefore lead to further improvements in the performance of our architecture.

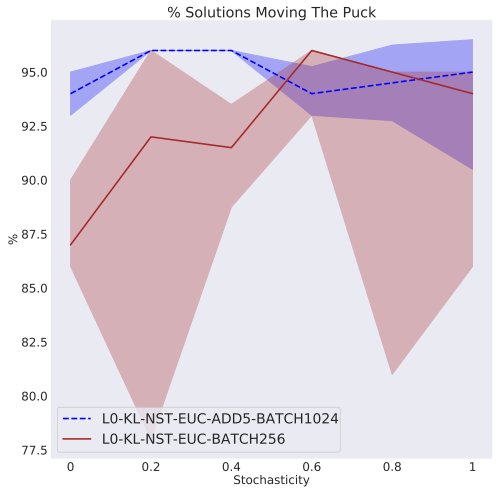


Figure 8.101: The proportion of solutions moving the puck.

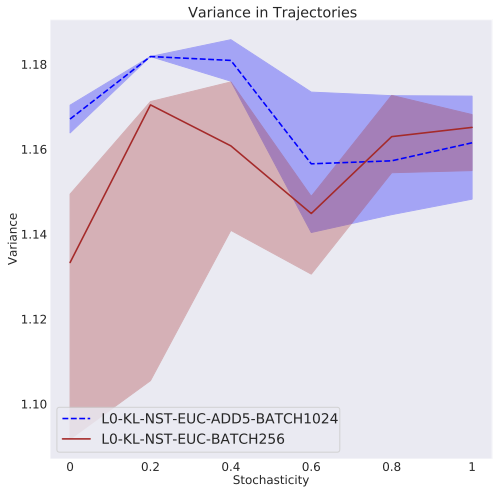


Figure 8.102: The variance in the puck trajectories.

8.4 Conclusion

In this experiment, we change the environment observations from sequences of puck position coordinates to synthetically created images. Despite this significant change, our main findings largely mirror the results of Experiment 2.

The decision not to sample the BDs from the Encoder leads to an increase in the compactness of the BD spaces constructed by all our evaluated variants. The diversity performance gains that follow are extended even further when we eliminate the Encoder variance term altogether and add a KL criterion to our architecture.

The addition of a SNE criterion leads to a discrimination of solutions based on their trajectory lengths. While the shortest of trajectories are yet again more likely to be rejected from the archive, the proportion of solutions with the longest trajectories also shrinks significantly. The resulting increased similarity between solutions yields inferior diversity metrics compared to the KL variants.

The TSNE variants avoid these drastic changes to the archives' composition of solutions. However, this comes at the expense of a significantly less compact BD space. Paired with an increased reliance on the construction accuracy of the VAE, this results in an uncompetitive diversity performance.

The Log-Likelihood variants' construction performances suffer from an instability introduced by the presence of the Decoder variance term. While the variants experienced these issues largely only in a noise-free environment in Experiment 2, the impact is clearly visible across stochasticities in this set of experiments. The performance stabilises when we remove the variance term and increases further when we

increase the size of the mini-batches and the training dataset.

Of our proposed 4 adaptations to the AURORA baseline, one benchmark improves the performance, albeit only marginally. It remains highly sensitive to the environment noise and its diversity performance deteriorates significantly as the environment stochasticity increases. At the largest noise levels, it fails to match the diversity of a randomly generated archive when we do not consider the no-move solutions. However, when the environment is non-stochastic, AURORA generates the archives with the largest diversity yet again. Our proposed architecture produces diversity metrics that approximate the AURORA algorithm's strongest performance scores and remains unaffected by changing noise levels.

Chapter 9

Experiment 4: Multi-modal Sensor Fusion

In our previous experiments, we relied on the KL divergence criterion to ensure that the VAE's strong construction performance translates into accurate BDs. In the original AURORA implementation, the environment observations serve as the inputs to the Autoencoder. The KL criterion is therefore not required to ensure accurate BDs. When the environment is non-stochastic, a change in the observation can only arise as a result of a change in the robot's behaviour. Any set of solutions producing the same behaviour therefore produces the same associated observations, which in turn generate the same Behavioural Descriptor.

In this set of experiments, we explore a multi-modal VAE that takes both the solution parameters and the synthetic images used in Experiment 3 as its inputs. By additionally providing the image observations, we try to inject a direct relationship between the environment observation and the BD into our architecture. However, an over-reliance on the image modality can lead to a re-introduction of the AURORA implementation's undesired sensitivity to noise.

9.1 The Variants

We consider two architectures that differ in the mechanism that creates the solution's BD from the two input modalities.

Architecture LC is illustrated in Fig. 9.1. We replicate the same Encoder network structure used in our AURORA implementation to create a latent representation of the image. We call this the reconstruction sub-network. Similarly, we replicate the Encoder network used in our VAE to produce a latent representation of the solution parameters. This is the construction sub-network. Additionally, each sub-network produces a scalar output. We concatenate these two outputs and apply a soft-max activation to obtain two weights summing to 1. The BD is then obtained as the weighted linear combination of the two produced representations.

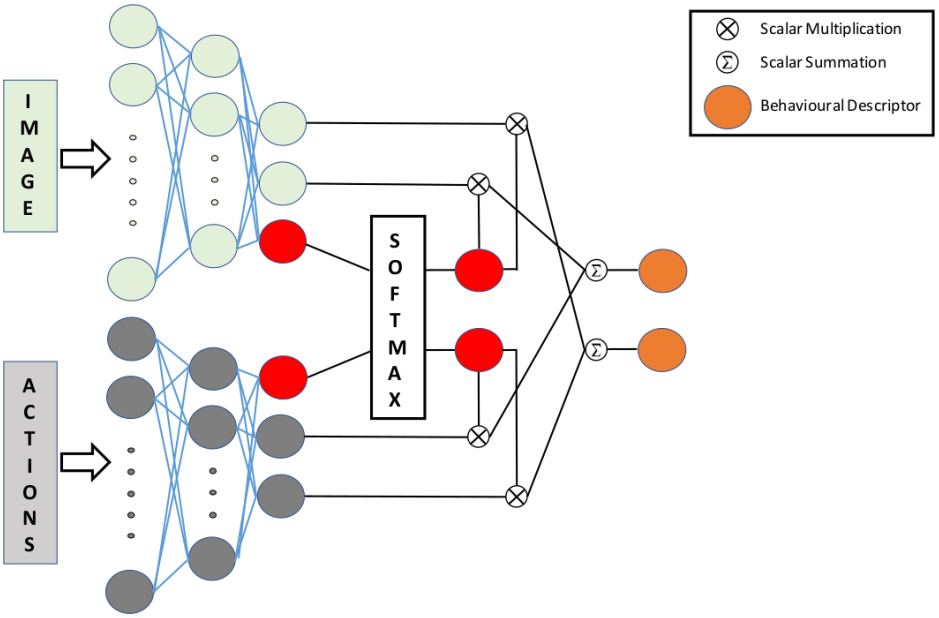


Figure 9.1: Illustration of the Encoder architecture used in LC.

In the FC architecture, we instead merge the two sub-networks' final hidden layer to produce a single latent representation. This is illustrated in Fig. 9.2.

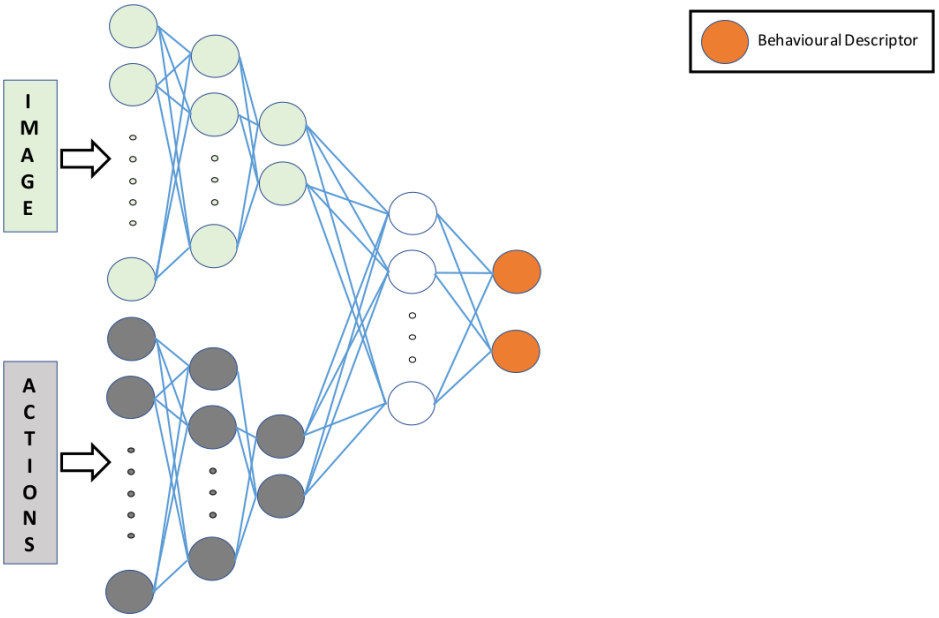


Figure 9.2: Illustration of the Encoder architecture used in FC.

9.2 Code Implementation

We copy the AURORA and VAE Encoder implementations from Experiment 3. These are combined to construct the FC and LC Encoder compositions described in Chap-

ter 9.1. The hyperparameters controlling the algorithm and Neural Network architecture are given in Appendix B.1 and B.5.

9.3 Results and Discussion

The LOG variants consistently achieve a large PCTMOVE score in Fig. 9.3 in both the FC and LC architectures. In previous experiments, large PCTMOVE scores of magnitudes similar to those achieved by the LOG variants were usually driven by a SNE or KL divergence criterion generating a large compactness in the BD space. Interestingly however, the KL loss in Fig. 9.4 is significantly larger for the LOG variants than for their Euclidean counterparts.

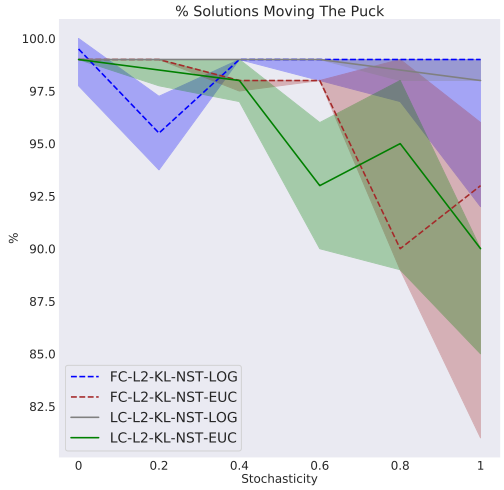


Figure 9.3: The proportion of solutions moving the puck.

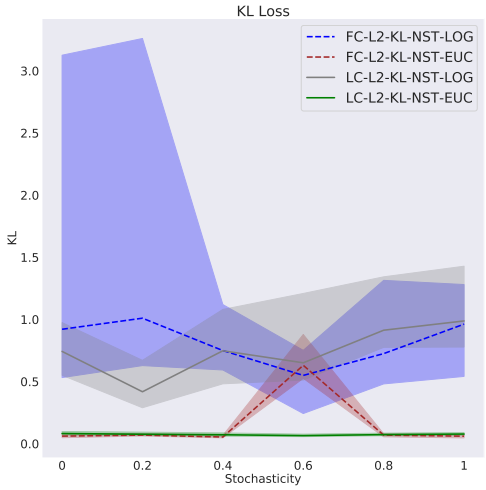


Figure 9.4: The KL Loss.

Moreover, despite the large PCTMOVE scores, the variance in the latent descriptors of no-move solutions in Fig. 9.5 is significantly larger for the LOG variants than their Euclidean counterparts. This indicates a poor construction accuracy which leads to a scattering of the no-move solutions' BDs across the BD space. Fig. 9.6 shows a BD space constructed at 100% stochasticity by the LC Log-Likelihood variant where this is clearly visible.

The deterioration in the construction accuracy and the lack of compactness in the BD space indicate that the LOG variants are likely relying on the image inputs to some extent to produce the large PCTMOVE scores in Fig 9.3.

For the FC variant, we qualitatively confirmed this by visualising the constructions made by the network. Many contain accurate reconstructions of the noisy trajectories. These were not made by any of our architectures when they did not have access

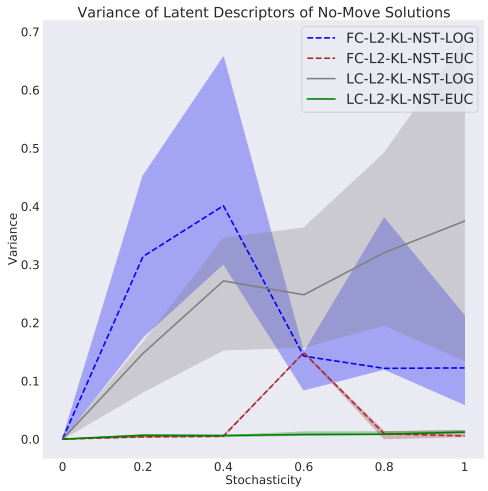


Figure 9.5: The variance in the no-move solutions' latent descriptors.

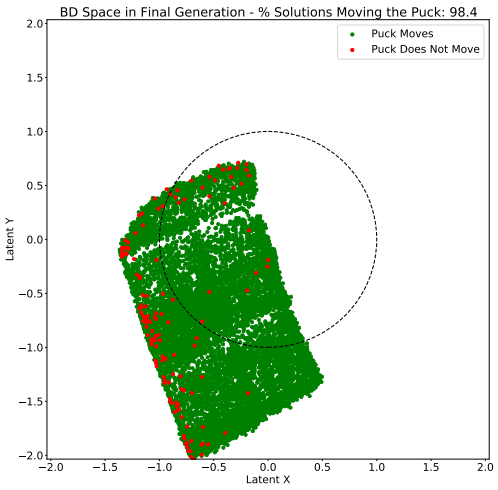


Figure 9.6: BD Space constructed by LC-L2-KL-NST-LOG at 100% stochasticity.

to the image modality. We present an example in Fig. 9.7.

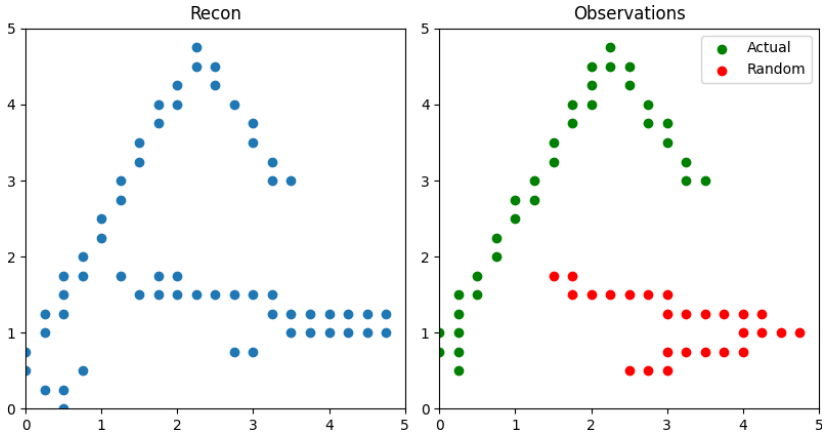


Figure 9.7: Left: Constructed image with pixel activations thresholded at a value of 0.5. Right: Actual and random puck trajectories in the original observation.

For the LC variant, we can attempt to confirm a reliance on the image inputs quantitatively. In Fig. 9.8, we visualise the learned scalar weights assigned to the latent representation of the image modality. While the difference between the LOG and EUC variants does not seem statistically significant, we can observe a trend in the LOG variant's weights that suggests that the magnitude of the Decoder variance term might have an impact on these weights. The Decoder variance increases towards both ends of the stochasticity spectrum. In the lower stochasticity ranges, a large spike in the initial generations leaves the Decoder variance at larger values throughout the remaining generations. This effect weakens as the stochasticity increases. We discussed this in detail in Chapter 8.3.4. Conversely, when the environment stochas-

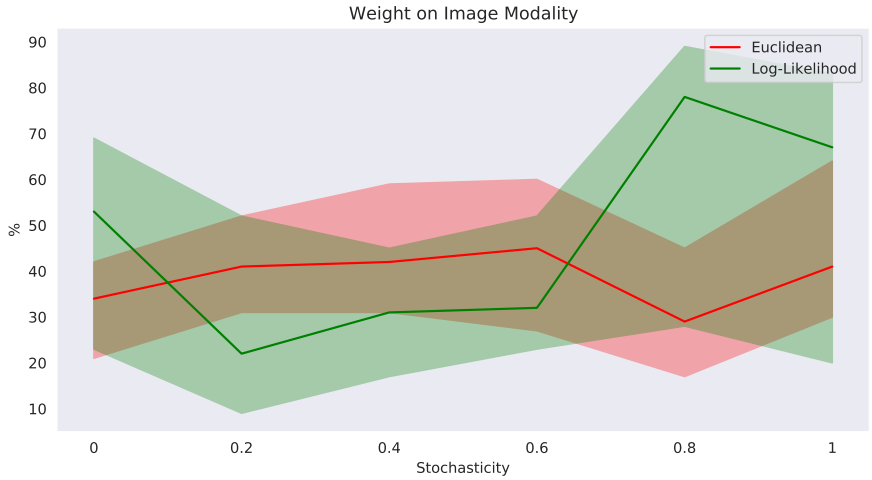


Figure 9.8: Weight assigned to the image representation when forming the BD in LC architectures.

ticity is large, the Total L2 losses increase. This yields larger Decoder variances. The shape of the magnitudes of the variances across the stochasticities could therefore bear some similarities to the LOG variant's weights on the image modality in Fig. 9.8.

When the variance increases, the magnitude of the overall construction loss gradient tends to decrease. We saw this in our previous experiments, when we observed decreasing KL losses as the environment stochasticity increased. A smaller construction loss gradient leads to a less rapid development of both Encoder sub-networks. However, the construction of the image from the solution parameters is a relationship more complex to learn than the reconstruction of the image. Consequently, the weight of the image modality remains at a larger level at both extremes of the stochasticity range.

When the Decoder variance is at lower levels, the magnitude of the construction loss gradient increases. The construction sub-network will learn the relationship between the solution parameters and the label image at a faster rate. Its increased accuracy results in a lower weight on the image modality at moderate stochasticities.

The Euclidean variant does not have a Decoder variance term that can change its magnitude. Consequently, the weight on the image modality remains largely the same across stochasticities.

We support this further with visualisations of the progression of the weights over the algorithm generations. Fig. 9.9 shows this for the Euclidean LC variant. The weights emerge from the first training iteration at a value around 50% and decrease over the following generations, as the performances of both sub-networks improve.
Conversely, in the LOG variant the weights in Fig. 9.10 emerge from the first training iteration at a higher value than in the Euclidean variant. The subsequent decrease in

Figure 9.9: Progression of learned weights on the image modality in the LC-KL-NST-EUC at 0% stochasticity.

the weight is also significantly slower than for the LOG variants as both sub-networks are slower to improve.

Figure 9.10: Progression of learned weights on the image modality in the LC-KL-NST-LOG variant at 0% stochasticity.

The final question that remains, is why the network seems to increasingly prefer the construction over the reconstruction sub-network as both networks' performances improve.
In our experiments, the VAE is not faced with a standard reconstruction task. Novel samples are added to the dataset throughout the algorithm generations.
An accurately learned relationship between the solution parameters that control the arm and the resulting puck trajectory, is likely to generalise better than a reconstruction ability of the image inputs. When a given novel sample is first added to the dataset, a strong construction sub-network can therefore still produce an informative latent description. On the other hand, the reconstruction sub-network is much less likely to produce a reconstruction that resembles an observation that is has never

seen before, even if it produces good reconstructions for the existing samples. We have seen indications of this in Chapter 8.3 when we observed that the reconstructions produced by the AURORA baseline tended to ignore infrequently occurring activations in the environment observations.
As a result, whenever novel samples are added, the network finds that the construction loss incurred can be decreased further by increasing its reliance on the more informative latent representations produced by the construction sub-network. This results in a decrease of the weights on the image modality.

We inspected the LC variants' construction outputs at 80% stochasticity where the difference between the LOG and EUC variants' weights is the largest in Fig. 9.8. In the Log-Likelihood variant, the constructions produced are very similar across solutions and only few pixels are activated. We provide a representative example in Fig. 9.11.

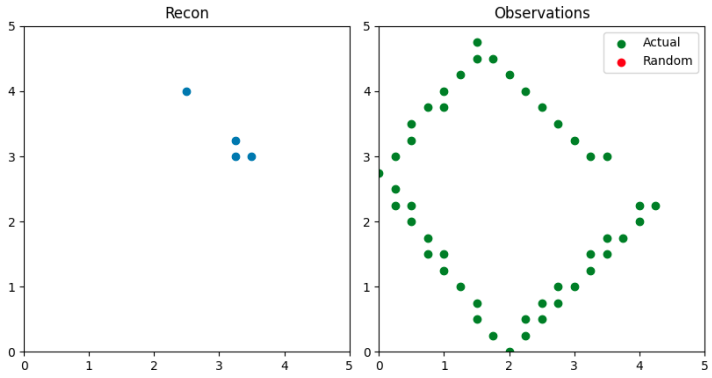


Figure 9.11: Left: Constructed image with pixel activations thresholded at a value of 0.5. Right: Actual and random puck trajectories in the original observation. LC LOG variant at 80% stochasticity.

On the contrary, the Euclidean variant produces very accurate constructions that largely exclude any noisy elements in the environment due to its lower reliance on the images. We present an example in Fig. 9.12.

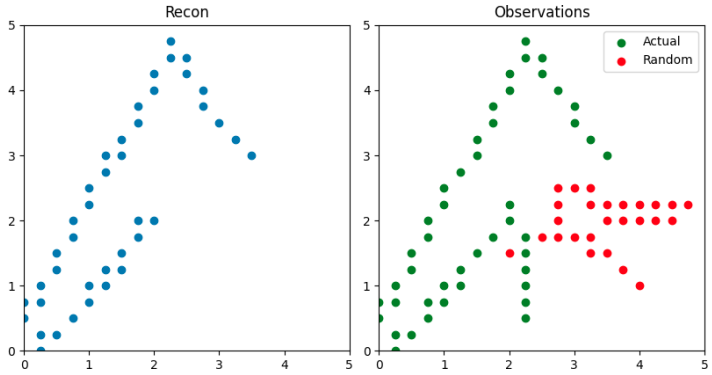


Figure 9.12: Left: Constructed image with pixel activations thresholded at a value of 0.5. Right: Actual and random puck trajectories in the original observation. LC EUC variant at 80% stochasticity.

We do not have the option to directly inspect the weights in the FC implementations. However, both FC variants shows largely similar performances to the LC implementations in the metrics presented thus far. This indicates that the FC Log-Likelihood implementation also relies on the image modality to a larger extent than the Euclidean implementation. Indeed, for the FC architecture, we find the same poor construction quality in the LOG and very accurate and noise-discriminating constructions in the EUC variant. We present two representative examples of each variant respectively in Figs. 9.13 and 9.14.

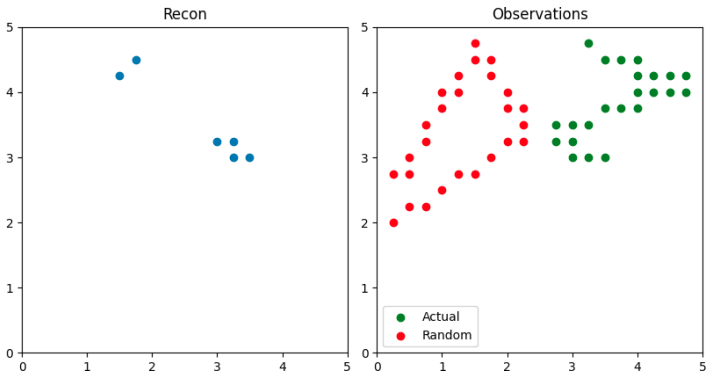


Figure 9.13: Left: Constructed image with pixel activations thresholded at a value of 0.5. Right: Actual and random puck trajectories in the original observation. FC LOG variant at 80% stochasticity.

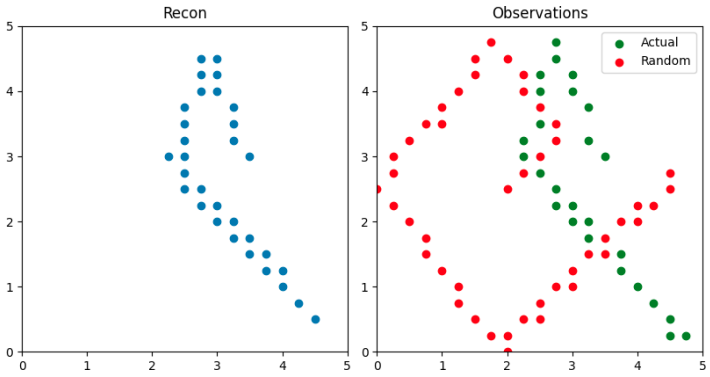


Figure 9.14: Left: Constructed image with pixel activations thresholded at a value of 0.5. Right: Actual and random puck trajectories in the original observation. FC EUC variant at 80% stochasticity.

Finally, we investigate whether the networks adjust the weight on the image modality based on the inputs they receive. The Encoder reconstruction sub-network for instance could produce a larger weight when it receives an input that it believes to contain noisy elements. However, this does not seem to be the case. In Fig. 9.15 we visualise the LC Euclidean's weights on the image modality for the samples that contain noisy observation elements and for the samples that do not. Fig. 9.16 shows the same visualisation but for the LC LOG variant. In both illustrations, the weights are largely the same for both sets of samples.

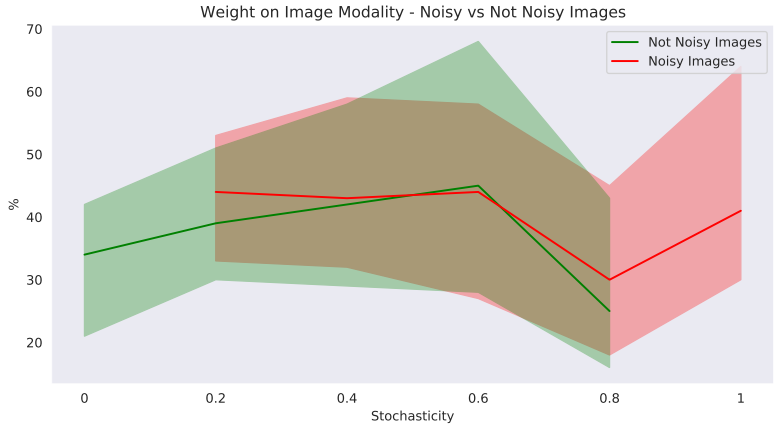


Figure 9.15: Learned weights on the image modality in the LC-KL-NST-EUC variant for noisy samples and for clean samples.

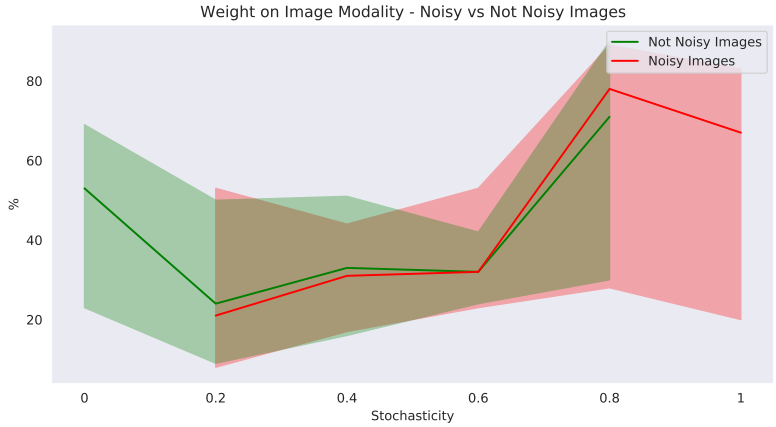


Figure 9.16: Learned weights on the image modality in the LC-KL-NST-LOG variant for noisy samples and for clean samples.

In both architectures, the LOG variants produce better diversity metrics but only marginally. We present the diversity score in Fig. 9.17.
The LOG variants' stronger performance follows its significantly larger PCTMOVE scores that result in the acceptance of more diverse solutions even if these are misconstructed under the influence of noise. All variants seem influenced by the noise albeit less so than the AURORA baseline. This indicates again that both architectures utilise a mixture of both modalities.

All performance metrics remained unchanged when we replaced the L2 criterion with the Huber loss that produced our best architecture in Experiment 3. We demonstrate this using the diversity score for the LC and FC variants in Figs. 9.18 and 9.19 respectively.

When comparing the two architectures, the LC variants seem to produce a larger

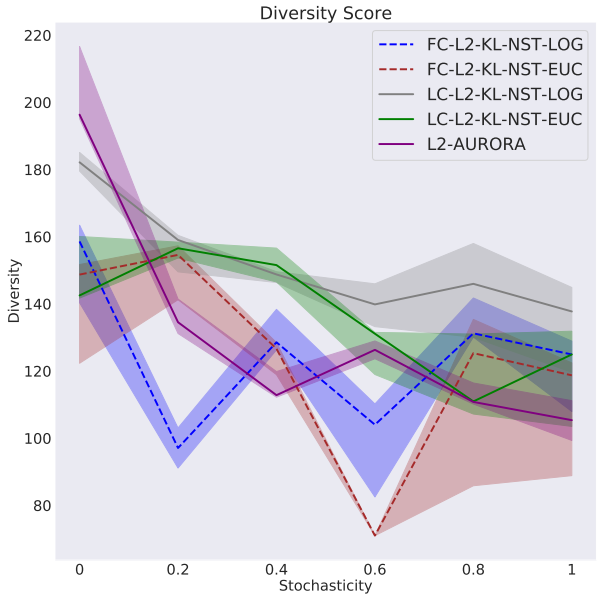


Figure 9.17: The Diversity Score.

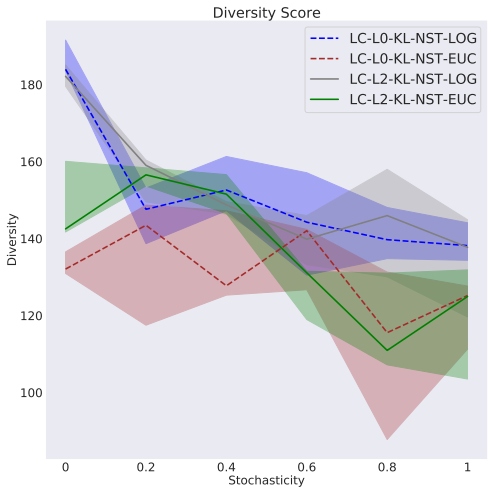


Figure 9.18: The Diversity Score.

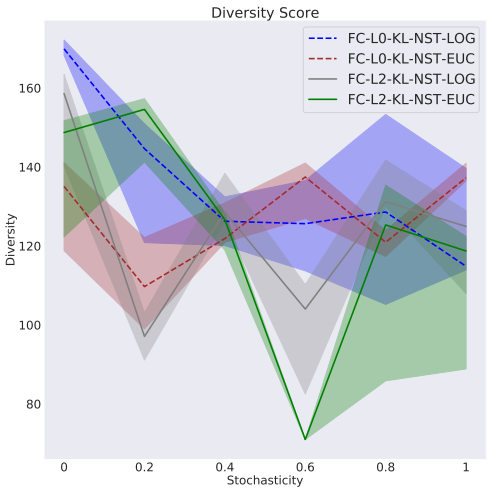


Figure 9.19: The Diversity Score.

diversity score in Fig. 9.17 than their FC counterparts. However, they are unable to match the performance or the robustness of our L0-KL-NST-EUC variant from Experiment 3. We demonstrate this in a comparison of the diversity and entropy scores in Figs. 9.20 and 9.21.

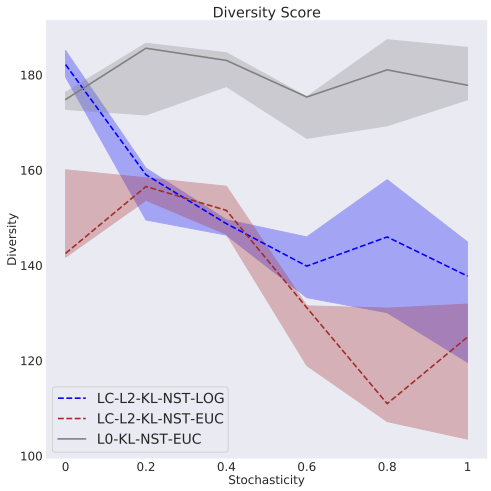


Figure 9.20: The Diversity Score.

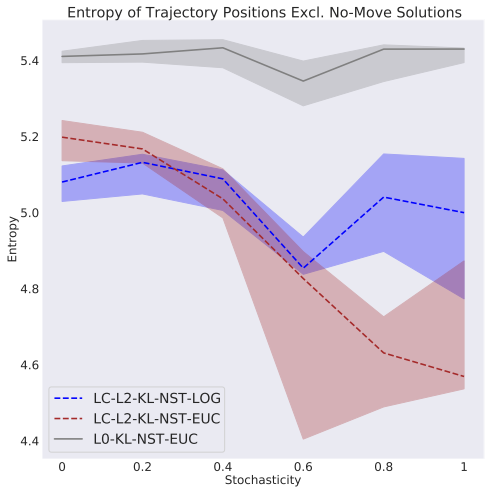


Figure 9.21: The Entropy Score.

As a final note, we present some of the most interesting latent spaces we inspected in our analysis.

In both architectures' LOG implementations, a typical BD space is presented in Fig. 9.22. However, repetitions of the same experiment can produce different and rather peculiar shapes that resemble the example presented in Fig. 9.23. Note, that the no-move solutions are plotted last and therefore cover up a large portion of the green descriptors in Fig. 9.23. We observe these strange shapes regularly across stochasticities.

In the LC-LOG variant, we observe many more irregular shapes. Two examples are shown in Figs. 9.24 and 9.25. These were not observed in the FC architecture, as the sub-networks' merged final layer likely produces a smoother latent space than the linear combination used in the LC architecture.

Both variants' Euclidean implementations produced latent spaces without any such intense irregularities but in a large variety of shapes. We provide two examples in Figs. 9.26 and 9.27.

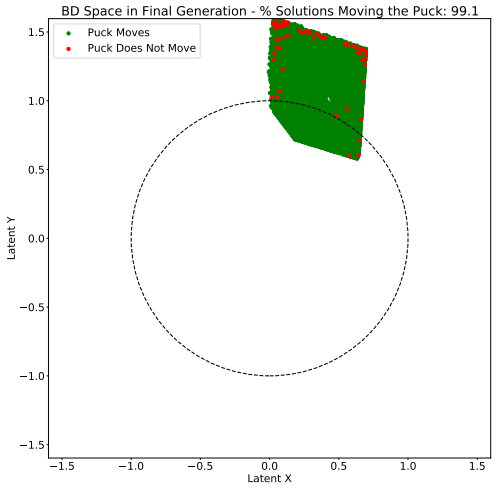


Figure 9.22: BD Space constructed by FC-L2-KL-NST-LOG at 100% stochasticity.

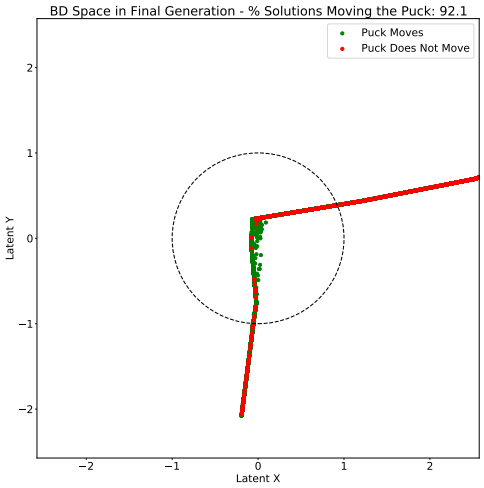


Figure 9.23: BD Space constructed by FC-L2-KL-NST-LOG at 100% stochasticity.

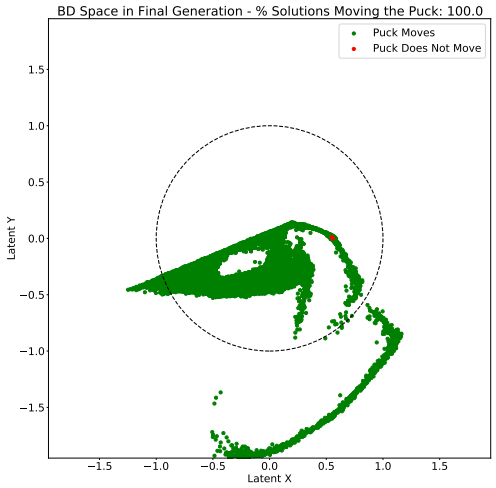


Figure 9.24: BD Space constructed by LC-L2-KL-NST-LOG at 0% stochasticity.

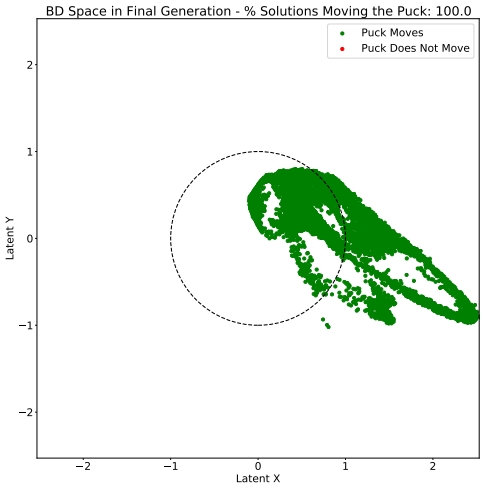


Figure 9.25: BD Space constructed by LC-L2-KL-NST-LOG at 0% stochasticity.

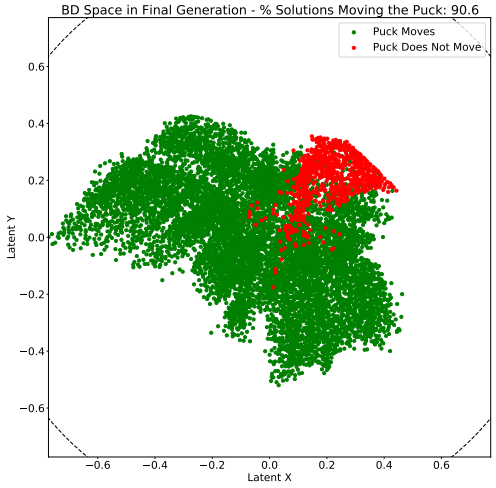


Figure 9.26: BD Space constructed by LC-L2-KL-NST-EUC at 100% stochasticity.

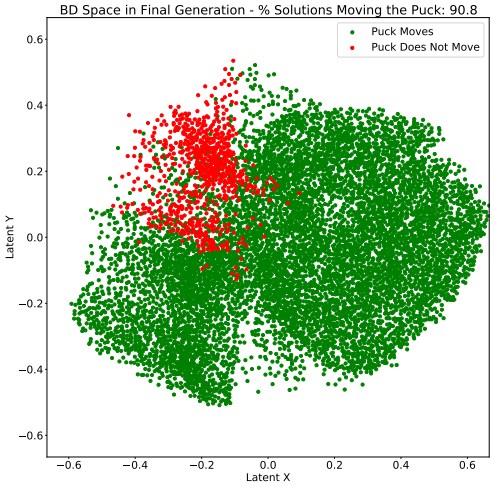


Figure 9.27: BD Space constructed by LC-L2-KL-NST-EUC at 100% stochasticity.

9.4 Conclusion

The addition of the image modality has the desired effect of increasing the BD accuracy when describing solutions that are perceived as similar or equal. This was demonstrated by a low proportion of no-move solutions. However, the use of these images also leads to a distortion in this perception when the environment is stochastic. The tested implementations therefore do not provide the desired robustness to noise. Nevertheless, the Euclidean variants' small reliance on the image modality is both surprising and promising for the further development of this multi-modal VAE.

Chapter 10

Conclusion and Future Work

In this work, we introduced a VAE architecture to establish an insensitivity to the presence of noise in environment observations used by the AURORA algorithm. We achieved this by conditioning the construction of the observations on the solution parameters that dictate the robot's actions. We validated this approach in experiments using two different types of environment observations. The first type consists of the ground truth puck position data generated by a robot arm's interaction with an air hockey puck. This is the same type of sensory data that is used in the original AURORA paper. The second type consists of synthetic images of the room and the puck trajectories.

The AURORA algorithm is severely impacted by environment noise in both scenarios. The diversity in the generated archives continues to decline as the stochasticity increases.

The use of our proposed VAE leads to a robust diversity performance. Our algorithm produces at all levels of noise the same values in our diversity metrics that the AURORA algorithm produces in a noise-free environment. In our best performing architecture we find that using the Encoder means as the solutions' BDs proves more effective in achieving a large diversity than producing latent descriptions using the reparameterisation trick. In this architecture, the application of a KL divergence term in the VAE optimisation avoids the negative impact of larger Encoder variances. Instead, the addition of the criterion leads to further improvements to the diversity of the generated archives by contracting the occupied latent space.

Conditioning the network on the solution parameters and the environment observations resulted in a lower diversity performance. However, we obtained an interesting result in the low magnitudes of the image modality's contribution to the final Behavioural Descriptor. This suggests that a conditioning of the VAE on the robot's actions is not only effective in discriminating noisy observation elements but beneficial to the network's construction accuracy when faced with a continuously changing dataset composition.

We propose three broad directions that future work can take. The first investigates

the performance of this architecture when faced with more complex challenges. For instance, we can increase the size of the solution space and simulate a robot arm with more degrees of freedom. Alternatively, we can use a larger solution space to create a task in which the solution parameters dictate the shape of the arm at the start and at the end of each simulation.

Experiments using environment observations that also capture the movement of the robot arm itself would likely produce very interesting results. In such a configuration, every solution generates a different behaviour in the observations. The set of no-move solutions would no longer produce an identical behaviour. However, solutions that generate non-still standing puck trajectories might still be perceived as more dissimilar from each other. The composition of solutions in the archive and the progression of behaviours over the algorithm's generations will likely provide interesting results too. For instance, we would expect the robot to focus on moving the arm in different ways in the initial generations and use these learned behaviours to produce a variety of puck trajectories subsequently.

Finally, further work can explore the use of sequential image data as the environment observations. An initial experiment could evaluate the application of a Convolutional Neural Network similar to Experiment 3's Decoder if we simply concatenate the sequence of images to form one large image array. If this architecture proves to be inadequate, we can investigate the use of a recurrent Neural Network architecture such as a LSTM [54].

Works in the second direction could investigate further performance improvements to our algorithm. We determined the two main drivers of performance to be the construction accuracy and the compression in the BD space. In isolation, improvements to the former did not produce any significant diversity changes. However, we found they can indirectly result in a further contraction of the occupied BD space by freeing up the network's capacity to reduce the KL divergence loss further. While the weight of the KL term in the network's optimisation can directly be influenced by setting a larger value of β, this comes at the expense of a decreased construction accuracy. Improving the performance in one experiment will therefore largely amount to hyperparameter tuning and produce an undesired sensitivity to the specifics of each experiment.

Instead, we propose to explore if variations to our multi-modal architecture can improve the performance further. For instance, we could attempt to use the image modality only as an auxiliary input. In such a configuration, the network first constructs an observation based on the solution parameters and uses the image inputs to fill in the activations caused by the noisy trajectories. A smart penalisation of the different outputs is required to avoid a degenerate output construction that is fully supplemented by the input modality. If successful, this would lead to significant increases in the accuracy of the constructions and the BDs. Furthermore, the total loss could be reduced to minimal magnitudes which can free up further capacity in the network optimisation and yield an increased compression of the BD space.

The execution wall-time of the algorithm is currently neither a limiting constraint nor a critical attribute of the algorithm. Nonetheless, we note that our proposed

architecture allows for the rejection of solutions without requiring a simulation of their corresponding environment observations. The BDs can be produced directly from the solutions' parameters. This is not possible in the AURORA implementation as the BDs are derived from the environment observations. The immediate rejection of solutions that are not sufficiently novel would result in immense increases in the execution speed and data efficiency. While these attributes are currently not of large importance to the performance, they are certainly required in any potential real world application of this algorithm. Further work could implement this feature and attempt to further exploit the VAE's resemblance to a forward model.

Finally, the third direction considers the addition of capabilities to any AURORA-based algorithm. Further work can attempt the generation of more complex behaviours, for instance, by creating a hierarchical behavioural repertoire [12].
The addition of a task-specific fitness objective to the AURORA algorithm will likely provide some interesting results. Using these results, we can attempt to develop a mechanism that creates an adequate fitness function automatically for any chosen task.

Appendix A

Legal, Social and Ethical Considerations

In discussion with the project supervisor, we determined that this project does not require any special considerations. We therefore answered “No” to all items in the Ethics checklist which we provide in Tab. A.1.
In this checklist, sections 1 - 5 do not apply to this project, as we generated our own datasets which consisted of physical simulations involving a robot arm and an air hockey puck. No humans or animals were simulated at any point in this project, nor was data gathered from any such individuals. Sections 6 and 7 are not applicable as our project is geography independent and the experiments were conducted exclusively using simulations.
Since this is a project in developing robot autonomy, a potential for military applications had to be considered carefully in section 8. However, rather than developing an autonomy in decision-making, our project aims to develop an algorithm that allows a robot to explore its own capabilities and skills. This technology is therefore highly unlikely to be applied in a military application, where robot capabilities will be well-known in advance and any autonomy introduced will likely aim to improve the robot’s decision-making process.
While we expect a further development of the work presented in this project to have large potential applicability to civilian applications, the technology is very versatile and there are likely many more possible applications than we realise. We therefore do not limit its applicability exclusively to civilian applications in the checklist.
Due to the reasons already given we also consider section 9 inapplicable.
With regards to section 10, all software used in this project is open-source and available to the public. Our developed code will also be made available to the public and distributed under permissive open-source licenses following a potential publication of our findings.
Finally, with regards to section 11, we developed the technology in this project with a large focus on creating a fast and light-weight application. All experiments are able to run to completion in a feasible time on consumer-grade hardware and in particular do not require any GPU access. We believe that this will make our algorithms more accessible to individuals who do not have access to any such resources.

	Y/N
Section 1: HUMAN EMBRYOS/FOETUSES	
Does your project involve Human Embryonic Stem Cells?	No
Does your project involve the use of human embryos?	No
Does your project involve the use of human foetal tissues / cells?	No
Section 2: HUMANS	
Does your project involve human participants?	No
Section 3: HUMAN CELLS / TISSUES	
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?	No
Section 4: PROTECTION OF PERSONAL DATA	
Does your project involve personal data collection and/or processing?	No
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?	No
Does it involve processing of genetic information?	No
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.	No
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?	No
Section 5: ANIMALS	
Does your project involve animals?	No
Section 6: DEVELOPING COUNTRIES	
Does your project involve developing countries?	No
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?	No
Could the situation in the country put the individuals taking part in the project at risk?	No
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY	
Does your project involve the use of elements that may cause harm to the environment, animals or plants?	No
Does your project deal with endangered fauna and/or flora /protected areas?	No
Does your project involve the use of elements that may cause harm to humans, including project staff?	No
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?	No
Section 8: DUAL USE	
Does your project have the potential for military applications?	No
Does your project have an exclusive civilian application focus?	No
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?	No

Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?	No
Section 9: MISUSE	
Does your project have the potential for malevolent/criminal/terrorist abuse?	No
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?	No
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?	No
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?	No
SECTION 10: LEGAL ISSUES	
Will your project use or produce software for which there are copyright licensing implications?	No
Will your project use or produce goods or information for which there are data protection, or other legal implications?	No
SECTION 11: OTHER ETHICS ISSUES	
Are there any other ethics issues that should be taken into consideration?	No

Table A.1: Ethics and Legal Considerations Checklist from `https://www.doc.ic.ac.uk/lab/msc-projects/ethics-checklist.xlsx`

Appendix B

Experiment Configuration Details

B.1 Common Configuration Elements Across Experiments

In Tab. B.1, we record configuration parameter values that are used in all experiments. In each implementation, we employ an early-stopping criterion in the Neural Network training. We stop the training if the error incurred on the validation set is larger than the running mean of validation errors over 5 epochs and the loss on the training set is smaller than before the start of training. This early-stopping criterion is only active after the initial 100 training epochs.

	Parameter	**Value**
Simulation	Room Size	5 x 5
	Puck Start Location	(3.55, 3)
	Trajectory Length	50
QD	Mutation Rate	0.1
	Cross Rate	0.1
	Mutation Operator	Polynomial
	Cross-Over Operator	SBX
	η_m (Mutation Parameter)	15
	η_c (Cross-Over Parameter)	15
	Archive Target Size	8000
	Solution Parameter Range of Possible Values	[0, 1]
	Behavioural Descriptor Size	2
	Tolerance on L distance threshold	10%
	Number of Offspring Solutions each Generation	256
VAE Training	First VAE training iteration	10th Generation
	Frequency of VAE training	Every 10 Generations
	Optimisation Algorithm	Adam [55]
	Adam Betas	(0.9, 0.999)
	Validation Set Size (% of total dataset)	80%
	Maximum Number of Training Epochs	6000

Table B.1: Configuration parameters used in all experiments.

B.1.1 SNE and t-SNE Parameters

We use a target perplexity of 25.6 in the calculation of the high-dimensional variances. This is derived as 0.1 * (batch size) as recommended in [25]. In the approximation of these variances, we perform binary search with a maximum number of 100 iterations and a tolerance of 0.00001. The SNE or t-SNE losses are added to the network's incurred training loss with a factor equal to the observation dimensionality (100 in Experiment 2 and 400 in Experiment 3), as recommended in [25].

B.2 Experiment 1

B.2.1 Simulation and QD Parameters

In this experiment, we simulate the puck trajectories manually. We assume no gravity or friction. Solutions consist of two elements. The first solution parameter is scaled to a range of [-π, π] and dictates the angle at which the puck leaves its starting trajectory. The second solution parameter is scaled to a range of [0, MAX_DPF] and sets the distance the puck travels between subsequent positions in the trajectory.

B.2.2 Neural Network Parameters

In both the AURORA implementation and our proposed architecture, the Encoder and Decoder network contain two hidden layers each. ReLU activations are applied after each layer except for the output layers. The number of neurons and other training configuration parameters are given in Tab. B.2.

Parameter / Layer	**Value / Number of Neurons**
Batch Size	64
Learning Rate	0.001
Encoder #1	10
Encoder #2	20
Decoder #1	40
Decoder #2	60

Table B.2: Neural Network Training Parameters and Architecture Details.

B.3 Experiment 2

B.3.1 Simulation and QD Parameters

In this experiment, we utilise the Box2D physics simulation engine to model a robot arm mounted in the center of the room. Solutions consist of 4 parameters, dictating the angle to be reached in each of four arm joints. The arm is controlled with servos

setting the motor speed at each joint to reduce the angle error. We specify the Box2D specific simulation parameters in Tab. B.3.

Note, we additionally had to adapt the Box2D source files to change a *#define* macro, to allow for our simulated pucks to bounce instead of gliding along any walls after impact. The change was made in file *box2d/include/box2d/b2_settings.h* to update the value of *#define b2_velocityThreshold* from *1.0f* to *0.0f*.

Configuration Element	Dimensions (in half sizes as per Box2D convention) / Value
Robot Base	0.0375 x 0.0375
Arm Segment	0.1875 x 0.015
Arm Segment #1 Density	1
Arm Segment #2 Density	0.666666
Arm Segment #3 Density	0.444444
Arm Segment #4 Density	0.296296
Arm Friction (All components)	0.8
Servo Maximum Motor Torque	1
Puck Radius	0.15
Puck Friction	0.8
Puck Restitution	0.8
Random Puck Maximum Force	1.5
Wall Friction	0.8
Simulation Duration	10secs

Table B.3: Box2D configuration parameters.

B.3.2 Neural Network Parameters

We use the same network architecture as in the previous experiment. The number of neurons used in the Encoder are the same as given in Tab. B.2. The number of neurons and other parameter values are given in Tab. B.4.

Parameter / Layer	Value / Number of Neurons
Batch Size	256
Learning Rate	0.0001
Encoder #1	10
Encoder #2	20
Decoder #1	40
Decoder #2	70

Table B.4: Neural Network Training Parameters and Architecture Details.

B.4 Experiment 3

B.4.1 Simulation and QD Parameters

We use the same QD and simulation configuration parameters as in Experiment 2. The synthetic images are created as 20 x 20 arrays. We use a 20 x 20 discretisation and set the value of an array element to 1 if any puck crosses the corresponding bin in the discretised grid at any time during the simulation. All other array elements take a value of 0.

B.4.2 Neural Network Parameters

In this experiment, we have two different Encoder networks. The Encoder network in our proposed architecture has the same architecture and number of neurons in each layer as in the previous Experiment. The AURORA Encoder network consists of six hidden convolutional layers each followed by a ReLU activation. The Decoder network is consists of 6 hidden transposed convolutional layers each followed by a ReLU activation. We provide the number of convolutional filters, kernel size and stride for each layer in Tab. B.5. No padding is applied. The batch size and learning rate are the same as in the previous Experiment.

Layer	Number of Filters	Kernel Size	Stride
AURORA Encoder #1	10	4 x 4	1
AURORA Encoder #2	10	3 x 3	2
AURORA Encoder #3	20	4 x 4	1
AURORA Encoder #4	20	3 x 3	2
AURORA Encoder #5	30	2 x 2	1
AURORA Encoder #6	30	1 x 1	1
Decoder #1	10	1 x 1	1
Decoder #2	10	2 x 2	1
Decoder #3	20	3 x 3	2
Decoder #4	20	4 x 4	1
Decoder #5	40	3 x 3	2
Decoder #6	40	4 x 4	1

Table B.5: Neural Network architecture details.

B.5 Experiment 4

B.5.1 Simulation and QD Parameters

We use the same QD and simulation configuration parameters as in Experiment 3.

B.5.2 Neural Network Parameters

The Encoder used in the FC and LC variant consists of two sub-networks. These sub-networks mirror the architecture and configuration details used for the AURORA Encoder and our proposed architecture's Encoder network from Experiment 3. In the FC variant, we add an additional fully connected layer with 6 neurons that takes the concatenated outputs of each sub-network as its inputs.

Appendix C

Execution Guide

Our containerised applications are built on machines with Ubuntu 18.04 and Singularity version 3.5.2. We therefore recommend this workspace configuration to guarantee a successful build of the applications.

In the code submission we provide separate files for each experiment. The following steps to build the applications are the same for each experiment. We use `experiment1` in our examples below.

In the instructions that follow, the user requires member permissions to access the source code stored in the AIRL GitLab group[1]. This group is owned and maintained by the project supervisor.

To build the applications, enter in the terminal shell of your machine:

```
cd experiment1/singularity
./build_final_image.sh
```

Once this build is complete, you will find a new file in the directory with a naming convention of

```
final_EXPERIMENT_YYYY-MM-DD_HH_MM_SS.sif
```

`EXPERIMENT` will take the values `balltrajectorysd`, `imagesd` and `dualconditioningsd` for Experiments 1-2, 3 and 4 respectively.

We abbreviate the name of this *.sif file as `FINAL.SIF` in the following execution instructions.

Note, that the applications in Experiment 2 - 4 offer GPU support. This requires a CUDA-enabled GPU on the machine and a CUDA driver version of 10.1. By replacing `./FINAL.sif` with `singularity run --nv FINAL.sif` in the following example commands, the applications will automatically detect a compatible GPU and use it for the Neural Network operations.

[1] `https://gitlab.doc.ic.ac.uk/AIRL`

C.1 Experiment 1

Each experiment configuration is driven by several command line arguments passed to the `FINAL.SIF` file. These are listed for Experiment 1 in Tab. C.1, along with their possible values and functions.

Argument	Possible Values	Function
number-gen	INT	Number of Generations
number-cpus	INT	Number of CPUs for Parallelisation
pct-random	FLOAT - [0, 1]	Environment Stochasticity
beta	FLOAT	KL Criterion Coefficient
full-loss	BOOL	Log-Likelihood (true) or Euclidean (false) Loss

Table C.1: Possible values and function of available command line arguments.

The command to perform an execution of our proposed architecture with a Log-Likelihood Construction Loss, 60% stochasticity in the environment, 6001 generations of the algorithm while using all available CPUs is given by,

```
./FINAL.sif balltrajectorysd_vae \
    --number-gen=6001 \
    --number-cpus=-1 \
    --pct-random=0.6 \
    --beta=1 \
    --full-loss=true
```

Note, that we implemented the L2-AE variant as a stand-alone application. It can be executed using `balltrajectorysd_ae` as the first argument, and with `full-loss` set to `false`. For instance,

```
./FINAL.sif balltrajectorysd_ae \
    --number-gen=6001 \
    --number-cpus=-1 \
    --pct-random=0.6 \
    --beta=0 \
    --full-loss=false
```

The L2-AURORA application is also implemented separately. It can be executed using `balltrajectorysd_aurora` as the first argument. For instance,

```
./FINAL.sif balltrajectorysd_aurora \
    --number-gen=6001 \
    --number-cpus=-1 \
    --pct-random=0.6 \
    --beta=0 \
    --full-loss=false
```

C.2 Experiment 2

The available command line arguments are given in Tab. C.2. All other configuration settings (e.g. the number of additional training archives) can be set in `experiment2/src/params.hpp`.

Argument	Possible Values	Function
number-gen	INT	Number of Generations
number-cpus	INT	Number of CPUs to use
pct-random	FLOAT - [0, 1]	Environment Stochasticity
full-loss	BOOL	Log-Likelihood (true) or Euclidean (false) Loss
beta	FLOAT	KL Criterion Coefficient
pct-extension	FLOAT	EXTEND adaptation, % of Dataset to re-add
loss-func	0, 1, 2	0: Smooth L1, 1: L1, 2: L2
sample-train	BOOL	Sampling from Encoder during Training
sample	BOOL	Sampling BD during Inference
sne	0, 1, 2	0: No SNE or TSNE, 1: SNE, 2: TSNE

Table C.2: Possible values and function of available command line arguments.

A sample command for the execution of an experiment using our VAE is given by,

```
./FINAL.sif balltrajectorysd_vae \
    --number-gen=6001 \
    --number-cpus=-1 \
    --pct-random=0.6 \
    --full-loss=true \
    --beta=1 \
    --pct-extension=0 \
    --loss-func=2 \
    --sample-train=false \
    --sample=false \
    --sne=2
```

The AE implementation can be achieved by setting

```
    --beta=0
    --full-loss=true
    --sample-train=false
    --sample=false
    --sne=0
```

The AURORA implementation can be executed using `balltrajectorysd_aurora` as the first argument. For instance,

```
./FINAL.sif balltrajectorysd_aurora \
```

```
    --number-gen=6001 \
    --number-cpus=-1 \
    --pct-random=0.6 \
    --full-loss=false \
    --beta=0 \
    --pct-extension=0 \
    --loss-func=2 \
    --sample-train=false \
    --sample=false \
    --sne=0
```

C.3 Experiment 3

The available command line arguments are given in Tab. C.3. All other configuration settings can be set in `experiment3/src/params.hpp`.

Argument	Possible Values	Function
number-gen	INT	Number of Generations
number-cpus	INT	Number of CPUs to use
pct-random	FLOAT - [0, 1]	Environment Stochasticity
full-loss	BOOL	Log-Likelihood (true) or Euclidean (false) Loss
beta	FLOAT	KL Criterion Coefficient
pct-extension	FLOAT	EXTEND adaptation, % of Dataset to re-add
loss-func	0, 1, 2	0: Smooth L1, 1: L1, 2: L2
sample-train	BOOL	Sampling from Encoder during Training
sample	BOOL	Sampling BD during Inference
sne	0, 1, 2	0: No SNE or TSNE, 1: SNE, 2: TSNE
sigmoid	BOOL	Sigmoid Activation in Output Layer

Table C.3: Possible values and function of available command line arguments.

A sample command for the execution of an experiment using our VAE is given by,

```
./FINAL.sif imagesd_vae \
    --number-gen=6001 \
    --number-cpus=-1 \
    --pct-random=0.6 \
    --full-loss=true \
    --beta=1 \
    --pct-extension=0 \
    --loss-func=2 \
    --sample-train=false \
    --sample=false \
    --sne=2 \
    --sigmoid=false
```

The AE implementation can be achieved by setting

```
--beta=0
--full-loss=true
--sample-train=false
--sample=false
--sne=0
--sigmoid=false
```

The AURORA implementation can be executed using `imagesd_aurora` as the first argument. For instance,

```
./FINAL.sif imagesd_aurora \
    --number-gen=6001 \
    --number-cpus=-1 \
    --pct-random=0.6 \
    --full-loss=false \
    --beta=0 \
    --pct-extension=0 \
    --loss-func=2 \
    --sample-train=false \
    --sample=false \
    --sne=0 \
    --sigmoid=false
```

C.4 Experiment 4

The available command line arguments are given in Tab. C.4. All other configuration settings can be set in `experiment4/src/params.hpp`.

Argument	Possible Values	Function
number-gen	INT	Number of Generations
number-cpus	INT	Number of CPUs to use
pct-random	FLOAT - [0, 1]	Environment Stochasticity
full-loss	BOOL	Log-Likelihood (true) or Euclidean (false) Loss
beta	FLOAT	KL Criterion Coefficient
loss-func	0, 1, 2	0: Smooth L1, 1: L1, 2: L2
sample-train	BOOL	Sampling from Encoder during Training
sample	BOOL	Sampling BD during Inference
sigmoid	BOOL	Sigmoid Activation in Output Layer

Table C.4: Possible values and function of available command line arguments.

A sample command for the FC architecture is given by,

```
./FINAL.sif dualconditioningsd_fc \
    --number-gen=6001 \
    --number-cpus=-1 \
    --pct-random=0.6 \
    --full-loss=true \
    --beta=1 \
    --loss-func=2 \
    --sample-train=false \
    --sample=false \
    --sigmoid=false
```

Experiments for the LC architecture can be executed by replacing `dualconditioningsd_fc` with `dualconditioningsd_lc` .

C.5 The Benchmarks

The code to build our benchmarks can be found in `experiment2_benchmark` and `experiment3_benchmark`.
A sample command is given by,

```
./FINAL.sif balltrajectorysd_benchmarks_EXP \
    --number-gen=6001 \
    --number-cpus=-1 \
    --pct-random=0.6 \
    --full-loss=false \
    --beta=0 \
    --pct-extension=0.4 \
    --loss-func=2
```

where the value of `EXP` in `balltrajectorysd_benchmarks_EXP` determines the benchmark. Possible values are `regen, extend, excl_train, excl_algo` for benchmarks REGEN, EXTEND, EXCLUDE_TRAIN and EXCLUDE_ARCHIVE respectively.

`--pct-extension` determines the proportion of the dataset that is regenerated or excluded.

Bibliography

[1] Antoine Cully. Autonomous skill discovery with Quality-Diversity and Unsupervised Descriptors. 9, 2019. pages 1, 2, 5, 23

[2] John Howard Long, Andrés Faíña Rodríguez-Vila, Kenneth O Stanley, Justin K Pugh, and Lisa B Soros. Quality Diversity: a new Frontier for evolutionary computation. 3:12, 2016. pages 1, 4

[3] Justin K Pugh, L B Soros, Paul A Szerlip, and Kenneth O Stanley. Confronting the Challenge of Quality Diversity. pages 1, 4

[4] Antoine Cully and Jean-Baptiste Mouret. *Behavioral Repertoire Learning in Robotics*. 2013. pages 1, 2, 5

[5] Antoine Cully and Yiannis Demiris. Quality and Diversity Optimization: A Unifying Modular Framework. Technical report. pages 1, 5, 18

[6] Joel Lehman and Kenneth O Stanley. *Evolving a Diversity of Creatures through Novelty Search and Local Competition*. 2011. pages 1, 5

[7] Joel Lehman and Kenneth O Stanley. Abandoning Objectives: Evolution through the Search for Novelty Alone. Technical Report 19, 2011. pages 1, 4

[8] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. pages 2, 4

[9] Sylvain Koos, Jean Baptiste Mouret, and Stéphane Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 17(1):122–145, 2013. pages 4

[10] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. apr 2015. pages 5

[11] Elliot Meyerson, Joel Lehman, and Risto Miikkulainen. Learning behavior characterizations for novelty search. In *GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, pages 149–156, New York, New York, USA, jul 2016. Association for Computing Machinery, Inc. pages 5

[12] Antoine Cully and Yiannis Demiris. Hierarchical Behavioral Repertoires with Unsupervised Descriptors. 2018. pages 5, 158

[13] Giuseppe Paolo, Alban Laflaquì Ere, Alexandre Coninx, and Stephane Doncieux. Unsupervised Learning and Exploration of Reachable Outcome Space. Technical report. pages 6

[14] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, jul 2006. pages 6

[15] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, dec 2014. pages 6, 7, 15

[16] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *31st International Conference on Machine Learning, ICML 2014*, volume 4, pages 3057–3070. International Machine Learning Society (IMLS), jan 2014. pages 6

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. -VAE: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019. pages 7

[18] Martina Zambelli, Antoine Cully, and Yiannis Demiris. Multimodal representation models for prediction and control from partial information. *Robotics and Autonomous Systems*, 123, jan 2020. pages 7

[19] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, jun 2013. pages 8

[20] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *Proceedings - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA 2018*, pages 80–89. Institute of Electrical and Electronics Engineers Inc., may 2019. pages 8

[21] Finale Doshi-Velez and Been Kim. Towards A Rigorous Science of Interpretable Machine Learning. feb 2017. pages 8

[22] Zachary C. Lipton. The mythos of model interpretability. *Communications of the ACM*, 61(10):35–43, jun 2018. pages 8

[23] Geoffrey Hinton and Sam Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, 2003. pages 8

[24] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2625, 2008. pages 8, 9

[25] Jacob M Graving and Iain D Couzin. VAE-SNE: a deep generative model for simultaneous dimensionality reduction and clustering. *bioRxiv*, page 2020.07.17.207993, jul 2020. pages 10, 163

[26] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. feb 2018. pages 10, 11

[27] Xingquan Zhu and Xindong Wu. Class Noise vs. Attribute Noise: A Quantitative Study. *Artificial Intelligence Review*, 22(3):177–210, nov 2004. pages 11

[28] Christopher M Bishop. Neural Networks for Pattern Recognition. 1995. pages 11

[29] Jocelyn Sietsma and Robert J.F. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67–79, jan 1991. pages 11

[30] Lasse Holmstrom and Petri Koistinen. Using Additive Noise in Back Propagation Training. *IEEE Transactions on Neural Networks*, 3(1):24–38, 1992. pages 11

[31] Yves Grandvalet, Stéphane Canu, and Stéphane Boucheron. Noise Injection: Theoretical Prospects. *Neural Computation*, 9(5):1093–1108, jul 1997. pages 11

[32] Chris M. Bishop. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7(1):108–116, jan 1995. pages 11

[33] Guozhong An. The Effects of Adding Noise during Backpropagation Training on a Generalization Performance. *Neural Computation*, 8(3):643–674, apr 1996. pages 11

[34] José A. Sáez, Mikel Galar, Julián Luengo, and Francisco Herrera. Analyzing the presence of noise in multi-class problems: Alleviating its influence with the One-vs-One decomposition. *Knowledge and Information Systems*, 38(1):179–206, jan 2014. pages 11

[35] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014. pages 11

[36] Alan Joseph Bekker and Jacob Goldberger. Training deep neural-networks based on unreliable labels. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 2016-May, pages 2682–2686, 2016. pages 11

[37] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019. pages 11

[38] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 8527–8537, 2018. pages 11

[39] Devansh Arplt, Stanislaw Jastrzbskl, Nicolas Bailas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Benglo, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 1, pages 350–359, jun 2017. pages 11

[40] Sunil Thulasidasan, Tanmoy Bhattacharya, Jeffrey Bilmes, Gopinath Chennupati, and Jamaludin Mohd-Yusof. Combating Label Noise in Deep Learning Using Abstention. 2019. pages 12

[41] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 2233–2241, 2017. pages 12

[42] Aritra Ghosh, Himanshu Kumar, and P. S. Sastry. Robust loss functions under label noise for deep neural networks. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 1919–1925, dec 2017. pages 12

[43] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep Learning is Robust to Massive Label Noise. 2017. pages 12, 89

[44] Michael I. Jordan and David E. Rumelhart. Forward Models: Supervised Learning with a Distal Teacher. *Cognitive Science*, 16(3):307–354, jul 1992. pages 13

[45] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated Binary Crossover for Continuous Search Space. *Complex Systems*, 9(2):1–34, 1994. pages 17

[46] Kalyanmoy Deb and Samir Agrawal. A Niched-Penalty Approach for Constraint Handling in Genetic Algorithms. In *Artificial Neural Nets and Genetic Algorithms*, pages 235–243. Springer Vienna, 1999. pages 17

[47] Jean-Baptiste Mouret and Stéphane Doncieux. Sferes v2 : Evolvin' in the Multi-Core World. Technical report. pages 22

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. 2019. pages 22

[49] John D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*, 9(3):99–104, may 2007. pages 22

[50] Stephane Doncieux, Alban Laflaquière, and Alexandre Coninx. Novelty search: A theoretical perspective. In *GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference*, pages 99–106, New York, NY, USA, jul 2019. Association for Computing Machinery, Inc. pages 92

[51] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings*, pages 10–21. Association for Computational Linguistics (ACL), nov 2016. pages 124

[52] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, nov 2017. pages 124

[53] Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 6307–6316. Neural information processing systems foundation, nov 2017. pages 124

[54] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, nov 1997. pages 157

[55] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, dec 2015. pages 162