Using Unsupervised Dynamic Feature Selection to Enhance Latent Representations

Anonymous Authors[1]

Abstract

Latent representations are critical for the performance and robustness of machine learning models, as they encode the essential features of data in a compact and informative manner. However, these representations are often affected by noisy or irrelevant features, which can degrade the model's performance and generalization capabilities. This paper presents a novel approach for enhancing latent representations using unsupervised Dynamic Feature Selection (DFS). For each instance, the proposed method identifies and removes misleading or redundant information, ensuring that only the most relevant features contribute to the latent space. By leveraging an unsupervised framework, our approach avoids reliance on labeled data, making it broadly applicable across various domains and datasets. Experiments conducted on image datasets demonstrate that models equipped with unsupervised DFS achieve significant improvements in generalization performance across various tasks, including clustering and image generation, while incurring a minimal increase in the computational cost.

1. Introduction

Feature selection and feature extraction are two essential techniques in machine learning and data analysis, both aimed at improving the efficiency and effectiveness of predictive modeling (Zebari et al., 2020). While both approaches share the goal of reducing the dimensionality of datasets, they diverge significantly in their fundamental strategies and objectives.

Feature selection aims to identify a subset of relevant features from a larger set of available features that are most informative for a given task (Dhal & Azad, 2022). Traditional feature selection methods typically rely on static criteria, selecting a fixed set of features during model training (Balın et al., 2019; Roffo et al., 2020; Cancela et al., 2023). However, in dynamic and evolving data environments, where the relevance of features may change over different samples, static feature selection may not be optimal. Dynamic Feature Selection (DFS) (also referred as *Instance-based or Instancewise Feature Selection* (Yoon et al., 2018; Panda et al., 2021; Liyanage et al., 2021)) represents a paradigm shift in feature selection by recognizing the variability of feature importance (Chen et al., 2018; Arik & Pfister, 2021) between each sample. Unlike traditional static methods, DFS algorithms adaptively adjust the feature subset during model training or deployment, accommodating changes in data characteristics and task requirements.

Moreover, feature extraction techniques transform the original feature space into a new space by creating a set of derived features that capture essential information from the original data. These methods, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), or any deep features derived from a deep learning model, seek to maximize the discriminative power or variance of the transformed features (Perera & Patel, 2019; Tang et al., 2022; Izmailov et al., 2022). In doing so, they often create a smaller, more compact representation of the data, potentially enhancing model performance but hindering its interpretability.

While both feature extraction and feature selection enhance the quality of input data for machine learning models, they differ fundamentally in their approach. Feature extraction generates entirely new features, potentially altering the interpretability of the data, while feature selection retains the original features, preserving the original meaning and context. In this context, DFS arises as a technique that tries to merge the versatility of feature extraction techniques with the interpretability of feature selection approaches.

This work presents a novel DFS method to provide an alternative to the classic data representation using deep features. The main contributions of this paper are the following:

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

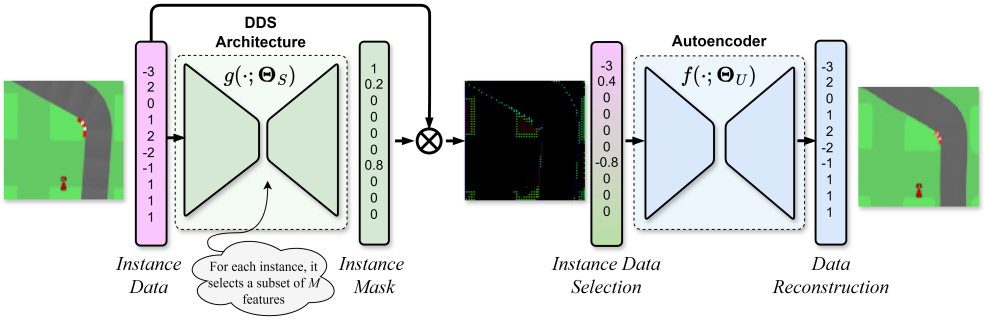

Figure 1. Proposed method. The DDS module is prepended to an existing architecture (an autoencoder in this case), substituting its input with an equally shaped masked version that retains only the most relevant features. DDS is in charge of selecting, for each sample, the most relevant features for the downstream architecture to solve the unsupervised task (in this particular example, data reconstruction).

- We present Dynamic Data Selection (DDS), a novel unsupervised dynamic feature selection algorithm. To our knowledge, this is the first attempt to provide a DFS solution for unsupervised scenarios.
- Contrary to previous supervised approaches like (Chen et al., 2018), DDS's memory consumption is minimal, and invariant to the maximum number of selected features.
- DDS can be easily adapted to a variety of problems and architectures, and it enables the architect designer to use complex networks since it preserves the position of the selected features. Fig. 1 shows, as an example, how to attach the DDS architecture to solve a data reconstruction task.
- We report extensive tests of DDS in two different unsupervised scenarios: clustering, and representation learning for world models. These experiments show DDS's adaptability to enhance the quality of latent representations, as well as its ease of use.

2. Related Work

DFS is a recent field of study, with almost no contributions before the rise of deep learning architectures. Three works stand out among the rest: *Learning to Explain* (L2X) (Chen et al., 2018), *INVASE* (Yoon et al., 2018), TabNet (Arik & Pfister, 2021) and LSPIN (Yang et al., 2022). However, it is worth noting that these algorithms were entirely developed for supervised learning.

L2X (Chen et al., 2018) (and variants like Greedy (Covert et al., 2023)) consists of an autoencoder-like architecture that is attached before the classification model. Its output is of the form $\mathcal{R}^{N\times F\times M}$, with M being the maximum number of features to be selected. Then, each input sample is transformed by performing a matrix multiplication. Although this solution provides remarkable results in supervised scenarios, it suffers from two major drawbacks: first, it has high memory requirements, as the output size of the model is dependent on the number of maximum features to be selected, forcing M to have smaller values; and second, the matrix multiplication procedure forces the input data to be one dimensional, preventing the use of complex layers like 2D convolutions in the classification model.

INVASE (Yoon et al., 2018) consists of 3 different networks: a selector, a predictor, and a baseline. Inspired by the actor-critic method (Peters & Schaal, 2008), the predictor and the baseline output classification scores but using different input data: the baseline receives all input features, but the predictor only uses a small subset of the input features. The selector algorithm determines the most suitable subset of input features. Compared to L2X, the main advantage of this algorithm is that it can preserve the spatial information of the input data. A major drawback is that it cannot be adapted to unsupervised scenarios, since it requires supervised data to train.

Moreover, TabNet (Arik & Pfister, 2021; Shah et al., 2022) and LSPIN (Yang et al., 2022) use sequential attention to select the most important features per sample. They were specially designed to be used in microarray scenarios, where the number of features is far greater than the number of samples. The main advantage of these approaches is their inherent explainability: they facilitate investigating the relevant features for each classification. However, their computational cost is too high for big data environments, and their accuracy is lower than L2X and INVASE.

3. Problem Formulation

Let $\mathbf{X} \in \mathcal{R}^{N\times F}$ be our input data, where N is the number of instances and F is the total number of different features. We aim to select, for each instance, a maximum number of features, denoted by M. Formally speaking, our algorithm

aims to solve the following minimization problem:

$$\begin{aligned} \underset{\boldsymbol{\Theta}_S,\boldsymbol{\Theta}_U}{\text{minimize}} \quad & \mathcal{L}(f(g(\mathbf{X};\boldsymbol{\Theta}_S)\circ\mathbf{X};\boldsymbol{\Theta}_U)) \\ \text{subject to} \quad & g(\mathbf{X};\boldsymbol{\Theta}_S)\in[0,1]^{N\times F}, \\ & \|g(\mathbf{X};\boldsymbol{\Theta}_S)^{(i)}\|_0 \leq M, \forall i\in\{1..N\}. \end{aligned} \tag{1}$$

where $\mathcal{L}$ is the unsupervised loss function, M is the maximum number of features to be selected, $g(\cdot;\boldsymbol{\Theta}_S)$ is the DDS network, and $f(\cdot;\boldsymbol{\Theta}_U)$ is the unsupervised task to solve. For the sake of simplicity, initially $f(\cdot;\boldsymbol{\Theta}_U)$ will be considered as an autoencoder that aims to reconstruct the initial features, including those that are masked by $g(\cdot;\boldsymbol{\Theta}_S)$. Section 4 shows how this approach can be adapted to solve other unsupervised tasks. The key idea is simple: an autoencoder architecture (f) is trained while introducing an extra network (g), tasked with selecting, at most, M relevant features per sample and masking the rest. After that, those features will be passed to the autoencoder model to reconstruct the whole unmasked input data.

In this paper, a novel DDS algorithm is proposed, with a $\mathcal{R}^{N\times F}$ output. The size of this output is not dependent on the maximum number of selected features. Note that this method is easily applicable to existing architectures solving unsupervised problems, as long as they can be trained by using gradient descent. DDS is defined as a module to be inserted before the existing architecture, replacing its input with masked input data. After training the whole architecture, the output of the DDS layer will contain only the most relevant features of each input.

3.1. DDS Implementation

As depicted in Eq. 1, two constraints have to be addressed to solve the minimization problem. To get a differentiable approach, we propose to reformulate the problem as

$$\begin{aligned} \underset{\boldsymbol{\Theta}_S,\boldsymbol{\Theta}_U}{\text{minimize}} \quad & \mathcal{L}(f(\tau(\tilde{g}(\mathbf{X};\boldsymbol{\Theta}_S)+\delta)\odot\boldsymbol{\Gamma}_M\odot\mathbf{X};\boldsymbol{\Theta}_U)) \\ \text{subject to} \quad & \tau(\tilde{g}(\mathbf{X};\boldsymbol{\Theta}_S)+\delta)\in[0,1]^{N\times F}, \\ & \boldsymbol{\Gamma}_M\in\{0,1\}^{N\times F}, \\ & \|\boldsymbol{\Gamma}_M\|_0\leq M, \forall i\in\{1..N\}. \end{aligned} \tag{2}$$

where

$$\tau(x)=\min\left(1,\max\left(0,\sigma\left(\frac{x}{\beta}\right)(\zeta-\gamma)+\gamma\right)\right), \tag{3}$$

where σ is the sigmoid function. $\tau(x)$ is the hard concrete gate presented in (Louizos et al., 2018), and δ is a hyperparameter to ensure non-zero values at the beginning of the training.

The idea is to split the DDS model g into two different terms. In the first place, $\tilde{g}(\cdot;\boldsymbol{\Theta}_S)$ is implemented in the same way as g was previously defined, without the 0-norm constraint. This constraint is now placed to a different matrix, called $\boldsymbol{\Gamma}_M$. $\boldsymbol{\Gamma}_M$ is a binary matrix with all zeroes but the top-M scores of each sample of $\tilde{g}(\cdot;\boldsymbol{\Theta}_S)$. During all the experiments reported in Section 4, the hyper-parameters were set to $\beta=\frac{2}{3}$, $\delta=1$, $\zeta=1$ and $\gamma=0$.

3.2. Hacking the Training Procedure

The training procedure is similar to the one used in (Louizos et al., 2018), with three slight variations.

First, the l_0-norm regularization term is not included. As the Γ matrix already removes $F-M$ features, it is unnecessary to use a regularization term to force some features to drop their importance to 0. For the same reason, $\gamma<0$ values are not needed. Our ablation study (Section 4.3) shows that using the default $\zeta=1.1$ and $\gamma=-0.1$ values results in a performance drop in the performance.

Second, an additional hyperparameter is used to control the binary concrete distribution. The hard binary concrete distribution presented in (Louizos et al., 2018) aims to increase the probability mass near 0 and 1, to either force some features to be discarded, or to increase the feature probability to near 1. Using the original distribution is also counterproductive, as it could unadvisedly force the model to remove more features than the target M, causing a degradation in performance. A variation of this distribution is presented to avoid this problem, and a hyperparameter is included in the formulation. This distribution is defined as

$$\tau_u(x)=\min\left(1,\max\left(0,\tilde{\sigma}_u(x)(\zeta-\gamma)+\gamma\right)\right), \tag{4}$$

where

$$\tilde{\sigma}_u(x)=\sigma\left(\frac{\kappa(\log(u)-\log(1-u))+x}{\beta}\right),\ u\in\mathcal{U}(0,1), \tag{5}$$

and $\kappa\in(0,1]$. By default, all experiments were performed using $\kappa=0.1$ during the training procedure, and $\kappa=0$ for testing.

Finally, early experiments suggest that the algorithm struggles with initialization when using low M values ($M<<F$). To solve this problem, M is dynamically changed, during training, by using the following variation:

$$M_t^{(i)}=\begin{cases} M & p_t^{(i)}>\epsilon, \quad p_t^{(i)}\in\mathcal{U}(0,1) \\ F & \text{otherwise} \end{cases} \tag{6}$$

For each training instance i, at any given epoch t, the DDS mask Γ_M selects, instead of M features, all F features, with a probability lower than ϵ. By default, ϵ is set to 0.1.

4. Experiments

The experimental section of this paper explores two main tasks. First, the DDS model is attached to a state-of-the-art clustering technique, to show how its inclusion can dramatically increase its performance; Second, DDS is used to learn a representation to be used by an agent in a World Model. Results show that DDS increases agent performance and the visual quality of the reconstructions. All algorithms, scripts, and results are accessible via GitHub[1]

4.1. Clustering

First, we explored the use of DDS in a clustering scenario. Our model was integrated into a state-of-the-art architecture without any calibration or hyperparameter tuning.

Over recent years, contrastive learning has boosted the quality of unsupervised image clustering. Techniques like Contrastive Clustering (CC) (Li et al., 2021) or its upgrade, the Twin Contrastive Clustering (TCL) (Li et al., 2022) achieved remarkable results nearing those obtained by supervised techniques. However, some of these works perform image resizing, changing the initial `CIFAR-10` image size 32×32 to a much bigger one (224×224 for TCL, for instance). Therefore, it makes no sense to perform a dynamic feature selection over an artificially enlarged image.

To make a fair experiment, DDS should be attached to a clustering model that does not require image resizing to achieve state-of-the-art results. Over these solutions, ProPos (Huang et al., 2022) stands out from the rest. Therefore, the DDS module $g(\cdot; \mathbf{\Theta}_S)$ was attached to the ProPos model $f(\cdot; \mathbf{\Theta}_U)$, without any hyperparameter tuning. The same training procedure presented in (Huang et al., 2022) was applied, although the number of epochs was increased to 2000 for `Tiny-Imagenet` and 4000 in the other datasets. The U-Net model (with $C = 16$) was used as the DDS architecture, and two values of M were tested: 10% and 25% of the total image pixels. As in (Huang et al., 2022), a ResNet-18 architecture will be used as the backbone for the Tiny-Imagenet dataset, whereas a ResNet-34 will be used for the others.

As baselines, we used SCAN (Van Gansbeke et al., 2020), NMM (Dang et al., 2021), GCC (Zhong et al., 2021), TCC (Shen et al., 2021), SimSiam (Chen et al., 2020), BYOL (Grill et al., 2020), IDFD (Tao et al., 2020) and PCL (Li et al., 2020) in addition to the aforementioned CC, TCL, and ProPos. Table 1 shows the clustering results obtained over four different datasets: `CIFAR-10`, `CIFAR-20` (Krizhevsky et al., 2009), `ImageNet-10` (Russakovsky et al., 2015) and `ImageNet-Dogs` (Khosla et al., 2011). The results show that DDS significantly reduced the number of input features without hindering the performance on small images. Furthermore, for larger images(`ImageNet-10` and `ImageNet-Dogs` models use 224×224 images), DDS achieved state-of-the-art results when selecting, per sample, only one-quarter of their features.

Table 2 presents the results obtained on the `Tiny-ImageNet` dataset. In this case, 64×64 images served as input for the model, reaching new state-of-the-art results and surpassing ProPos in the three metrics by a margin greater than 15%. We also show the result obtained when doubling the number of epochs, which increases this improvement up to an impressive 80% in the ARI score. These results suggest that removing unnecessary information, such as background details, can enormously help the clustering algorithm.

4.2. World models

The reinforcement learning problem of constructing agents that learn to interact with a dynamic environment has recently been tackled using world models. These models serve as generative frameworks for simulating environments internally and enable agents to predict and act based on imagined scenarios rather than relying on direct interactions with their surroundings. The usual architecture of these agents (Ha & Schmidhuber, 2018) relies on a separately trained vision model to construct a compact and structured latent representation of the environment which the agent then uses to determine the actions to be taken. A complete description is provided in Appendix A.

The vision model was originally solved by using a Variational Autoencoder (VAE) (Kingma, 2013), which is known to produce reconstructions that are blurry (Tomczak & Welling, 2018). We propose to maintain the architecture in (Ha & Schmidhuber, 2018) but enhance the Vision model (V), which obtains the latent representations of the environment, with our DDS model. The goal is to improve the efficacy of the obtained representations both in terms of agent performance and the visual quality of the generated images. The DDS architecture introduced in Fig. 1 is adapted by introducing a VAE to learn latent interpretations for the agent to use in the process of learning to reconstruct the mask selected by DDS. Changing the goal of the VAE to reconstructing the masked input ($\mathbf{mask} = g(\mathbf{X}) \circ \mathbf{X}$) instead of the original input ($\mathbf{X}$) simplifies the task of the VAE by removing noise, which allows it to obtain better representations. Fig. 2 describes the new architecture.

The training procedure has two steps: First, everything but the VAE is trained. To train as much of the network as possible in this first stage, the VAE is split into three functions $\mathbf{mask'} = f_{up}(f_{VAE}(f_{down}(\mathbf{mask})))$ and f_{VAE} is removed in this training step. The function to be learned is, therefore,

[1] URL will be included upon acceptance.

Table 1. Clustering results on four different datasets. The best and second-best results are shown in bold and underlined, respectively. The DDS+ProPos model maintains similar results to those obtained by the original ProPos, even when using much fewer selected features.

Method	CIFAR-10			CIFAR-20			ImageNet-10			ImageNet-Dogs		
	NMI	**ACC**	**ARI**	**NMI**	**ACC**	**ARI**	**NMI**	**ACC**	**ARI**	**NMI**	**ACC**	**ARI**
SCAN (Van Gansbeke et al., 2020)	79.7	88.3	77.2	48.6	50.7	33.3	-	-	-	-	-	-
NMM (Dang et al., 2021)	74.8	84.3	70.9	48.4	47.7	31.6	-	-	-	-	-	-
CC (Li et al., 2021)	70.5	79.0	63.7	43.1	42.9	26.6	85.9	89.3	82.2	44.5	42.9	27.4
GCC (Zhong et al., 2021)	76.4	85.6	72.8	47.2	47.2	30.5	84.2	90.1	82.2	49.0	52.6	36.2
TCL (Li et al., 2022)	81.9	88.7	78.0	57.9	53.1	35.7	87.5	89.5	83.7	62.3	64.4	51.6
TCC (Shen et al., 2021)	79.0	90.6	73.3	47.9	49.1	31.2	84.8	89.7	82.5	55.4	59.5	41.7
SimSiam (Chen et al., 2020)	78.6	85.6	73.6	52.2	48.5	32.7	83.1	92.1	83.3	58.3	67.4	50.1
BYOL (Grill et al., 2020)	81.7	89.4	79.0	55.9	56.9	39.3	86.6	93.9	87.2	63.5	69.4	54.8
IDFD (Tao et al., 2020)	71.1	81.5	66.3	42.6	42.5	26.4	89.8	95.4	90.1	54.6	59.1	41.3
PCL (Li et al., 2020)	80.2	87.4	76.6	52.8	52.6	36.3	84.1	90.7	82.2	44.0	41.2	29.9
ProPos (Huang et al., 2022)	**88.6**	**94.3**	**88.4**	60.6	**61.4**	45.1	89.6	95.6	90.6	69.2	74.5	62.7
DDS(10%)+ProPos	80.2	86.3	75.7	54.4	50.5	37.2	**91.8**	**96.7**	**92.8**	74.4	76.0	66.5
DDS(25%)+ProPos	87.6	93.9	87.2	**62.2**	58.4	**46.6**	90.8	96.2	91.7	**75.9**	**78.6**	**69.5**

Table 2. Clustering Results (%) on Tiny-ImageNet. The best and second-best results are shown in bold and underlined, respectively. The Long version was trained by doubling the number of epochs.

Method	NMI	ACC	ARI
CC (Li et al., 2021)	34.0	14.0	7.1
GCC (Zhong et al., 2021)	34.7	13.8	7.5
PCL (Li et al., 2020)	35.0	15.9	8.7
SimSiam (Chen et al., 2020)	35.1	20.3	9.4
BYOL (Grill et al., 2020)	36.5	19.9	10.0
ProPos (Huang et al., 2022)	40.5	25.6	14.3
DDS(10%) + ProPos	43.8	27.1	16.0
DDS(25%) + ProPos	47.0	30.5	18.9
DDS(25%) + ProPos (Long)	**54.6**	**40.3**	**26.3**

$f_{unet}(f_{up}(f_{down}(g(\mathbf{X}) \circ \mathbf{X})))$. The main idea is to learn to select the relevant features, as well as a lower resolution representation $\mathbf{h} = f_{down}(g(\mathbf{X}) \circ \mathbf{X})$ representing key points of the image structure and texture. The primary optimization objective is to minimize a pixel-level reconstruction loss, typically implemented as the mean squared error (MSE) between the input image $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$. The second step begins once both U-Nets (g and f_{unet}) and the downscaling/upscaling modules (f_{down} and f_{up}) have been fully trained and frozen. We train the *VAE* to reconstruct **mask** $= (g(\mathbf{X}) \circ \mathbf{X})$. In this process, it learns a more compact latent vector $\mathbf{z} \in \mathbb{R}^d$. Unlike standard VAEs (Kingma & Welling, 2022) that directly compress high-resolution inputs (often requiring large, deep networks), our VAE leverages the frozen f_{down} and f_{up}, and trains only f_{VAE}. With this approach, the reconstruction loss approaches a *perceptual loss* (Hou et al., 2017) without requiring supervised pre-trained models: it computes the MSE between the output of each layer of f_{up} when applied to either the latent representation $\mathbf{h}$ and its reconstruction $\hat{\mathbf{h}}$.

Finally, a known drawback of VAEs is that strict adherence to the $\mathcal{N}(0, 1)$ prior can lead to dimensional collapse: a few latent dimensions carry most of the information, while others remain dormant. To address this, we adopt the *free bits* method (Kingma et al., 2016), which modifies the VAE loss to allow each dimension a small "budget" of KL divergence.

Formally speaking, the VAE loss function is defined as:

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{perceptual}} + \alpha \max\left(\lambda, D_{\text{KL}}\big(q(\mathbf{z}|\mathbf{h}) \parallel p(\mathbf{z})\big)\right), \tag{7}$$

where

$$\mathcal{L}_{\text{perceptual}} = \frac{1}{|f_{\text{up}}|} \sum_{k=1}^{|f_{\text{up}}|} \|f_{\text{up}}(\hat{\mathbf{h}})_k - f_{\text{up}}(\mathbf{h})_k\|^2, \tag{8}$$

$f_{\text{up}}(\mathbf{X})_k$ is the output of the k-th layer of the upscaling module, and $\alpha = 1$ and $\lambda = 0.02$ are some regularization terms for the *free bits penalization.*

In our study, we followed the same procedure for dataset collection, model training, and evaluation, ensuring that our experiments remained consistent with the original (Ha & Schmidhuber, 2018) methodology, ensuring that any observed differences in performance and image fidelity could be attributed solely to the changes in the latent space construction method. We use the same latent dimension size ($\mathbf{z} \in \mathbb{R}^{32}$) for both the original VAE in (Ha & Schmidhuber, 2018) and for our method. We tune hyperparameters so that our approach does not exceed the baseline's parameter budget. While the Controller and Memory models are unchanged, the number of parameters for the Vision Model is reduced from 4,348,547 in (Ha & Schmidhuber, 2018) to

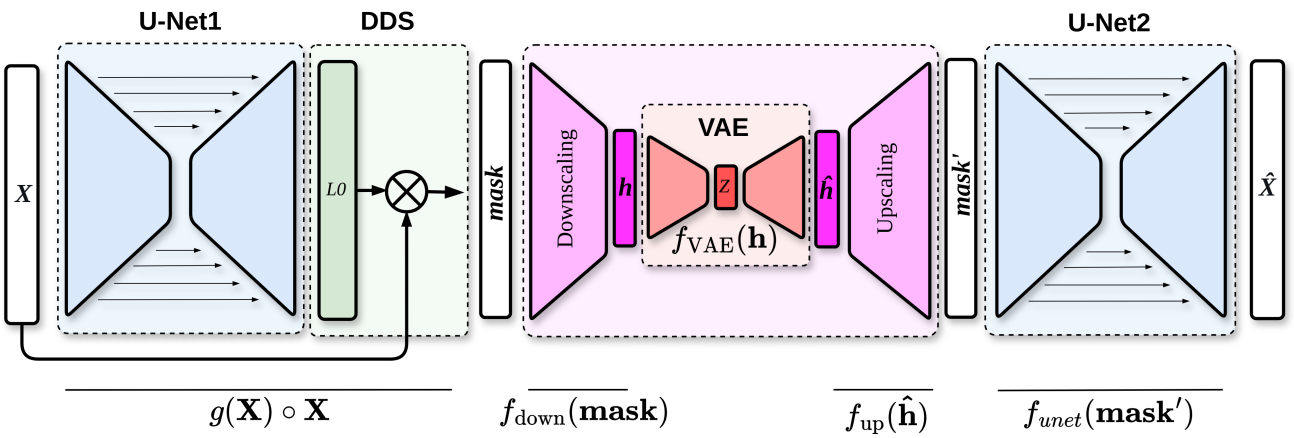

Figure 2. Adaptation of the DDS architecture to the World Model problem. The previously presented DDS architecture (green and blue) is augmented to yield a structured latent space with the addition of a Variational Autoencoder (red) that aims to reconstruct the masked inputs (i.e. the relevant features of the input image). The training procedure is divided in two steps: (1) the DDS is trained to learn to select the relevant features of each image without the *VAE* section (i.e. $\mathbf{h} = \hat{\mathbf{h}}$), and then (2) the *VAE* is trained to compute the $\hat{\mathbf{h}}$ reconstruction of $\mathbf{h}$.

4,039,089 in our approach. This shows that substantial gains in reconstruction fidelity and downstream RL performance can be achieved with a smaller model.

Image quality experiment: To assess the quality of the reconstructed images, we measured the reconstruction error of the different models. The results shown in Table 3 highlight that our DDS+VAE approach yields a noticeably lower reconstruction error than the baseline VAE, even when DDS selects as few as 4–8% of the input pixels. This indicates that DDS successfully pinpoints crucial image regions—capturing objects, textures, and boundaries—thus enabling the model to more accurately reconstruct the environment. For a qualitative illustration of the intermediate representations of our model, refer to Appendix B.

After training the Memory model using our new latent representation $\mathbf{z}$, we can generate *dream sequences* that simulate future states of the environment without direct interaction. Figure 3 shows the contrast between the baseline's results and our method's. The images *dreamed* by the baseline model are often blurry and show artifacts that make them visually dissimilar from the real environment. Furthermore, although the predicted dynamics remain loosely consistent with real motions, these generated states sometimes diverge significantly from the appearance and dynamics of the actual environment (particularly when the agent executes sharp turns or accelerates rapidly). In contrast, our approach yields dream sequences that retain more detail and exhibit smoother transitions between frames. This enhanced internal simulation capability can facilitate more effective training of downstream policies as shown in the next experiment.

We evaluate the proposed DDS+VAE model's ability to generate dream sequences compared to the baseline architecture proposed by Ha et al. (Ha & Schmidhuber, 2018). We generate and analyze three sets of sequences, each containing 100 sequences with 1,000 frames. The first set consists of real environment sequences. The second set comprises dream sequences generated using the baseline. The third set includes dream sequences produced with the proposed DDS+VAE model.

To assess the quality of the generated sequences, we use two standard metrics: Fréchet Inception Distance (FID)(Heusel et al., 2017), which measures image distribution similarity using features from a pre-trained Inception V3 model, and Fréchet Video Distance (FVD)(Unterthiner et al., 2018), which extends FID to evaluate temporal coherence using a pre-trained I3D model. Lower FID and FVD scores indicate higher visual fidelity and better sequence consistency, respectively. As shown in Table 4, the DDS+VAE model achieves significantly lower FID and FVD scores compared to the baseline world model, demonstrating improved image reconstruction quality and enhanced temporal coherence in the generated sequences.

Agent performance: Finally, we examine the agent's performance when deciding its policy using the latent representations obtained by the original and proposed Vision models. Figure 4 shows that the latent representations lead to superior performance, although the agent learns slightly slower. The evaluation, conducted over 100 episodes for each controller, highlights the advancements of the new

Table 3. Reconstruction MSE over 10.000 random rollouts (10 million images) from the `CarRacing-v3` environment using different M values. DDS configuration is the same as provided for Fig. 2, except for the values included in each row.

Vision model architecture	**2%**	**4%**	**8%**	**16%**	**32%**	**100%**
Baseline (VAE)	-	-	-	-	-	0.00165
Proposed (DDS + VAE)	0.00134	0.00039	0.00042	0.00047	0.00051	-

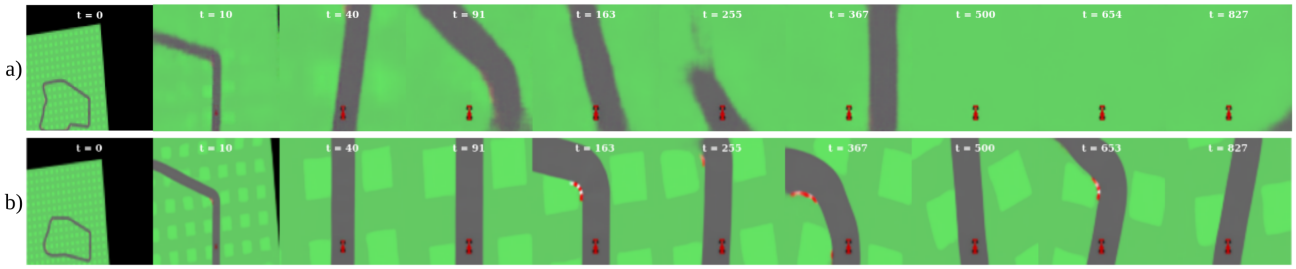

Figure 3. Comparison of dream sequence generation by the World Model. a) Original Vision model with a VAE architecture. b) Proposed (VAE+DDS) as Vision model.

Table 4. Comparison of FID and FVD between baseline Vision model (VAE) and the proposed DDS+VAE over 100 sequences. Lower is better.

Metric	**Baseline (VAE)**	**DDS+VAE**
FID	59.46	**25.35**
FVD	239	**176**

model. The original 2018 architecture achieved an average reward of 734.96 ± 162.75, whereas the new model reached 818.58 ± 147.05, confirming its enhanced performance.

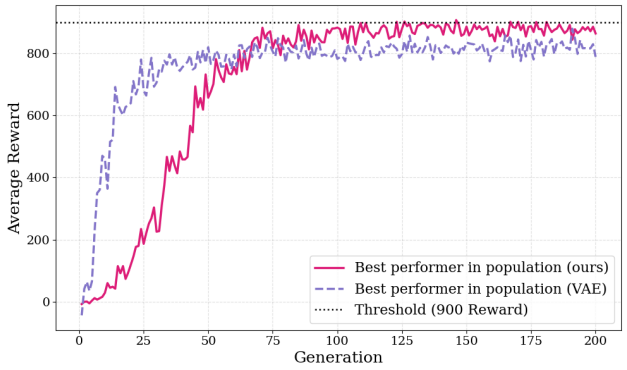

Figure 4. Average reward of the best-performing individual in the population. Two curves are shown: one representing the evolution of the original 2018 world model architecture and the other showing the improvement brought by the new vision.

4.3. Ablation Study

Multiple configurations were tested to determine both the correct model hyperparameters and the limitations of the architecture. We measured changes in performance for different configurations using the `CIFAR-10` dataset.

Table 5 shows the results of different MSE reconstruction configurations when using an autoencoder-like configuration (see Fig 1). The first and most important insight from this experiment is that the most advanced architecture of the model plays a pivotal role in the accuracy of the solution. DFS outperforms the naive AutoEncoder by preserving spatial information while maintaining a compact representation with fewer variables. This enables the use of advanced networks with residual links, whereas traditional feature selection methods require a latent representation comparable in size to the original input to leverage such architectures. Another insight is that using the hard sigmoid configuration provided in (Balın et al., 2019) drops the performance by a slight margin, specially when selecting low M values. This phenomenon is caused by the zero gradient obtained when the feature importance is cropped (values higher than 1 or lower than 0) in early training stages, causing a limitation in the ability of the model to adapt. This validates the change proposed in Section 3.1.

Lastly, the use of the Binary Concrete distribution was tested. Using the classic configuration ($\kappa = 1$) resulted in a drop in performance. Since the reconstruction task is a regression problem, it was found that high variations in the output of the DDS model result in the inability of the reconstruction model (Θ_U) to achieve good generalization. However, when setting $\kappa = 0$ the results are comparable with the ones obtained by our default configuration, suggesting that, for

Table 5. Reconstruction MSE over `CIFAR-10` retaining different M features.

M	64	128	256	512	1024
Naive AE	0.01820	0.01264	0.00806	0.00526	0.00449
DDS w/o Residual Links	0.03129	0.01804	0.01302	0.00844	0.00817
DDS Hard Sigmoid ($\zeta = 1.1, \gamma = -0.1$)	0.01838	0.00861	0.00397	0.00126	0.00030
DDS classic Binary Concrete ($\kappa = 1.$)	0.03759	0.01771	0.01415	0.00320	0.00064
DDS w/o Binary Concrete ($\kappa = 0.$)	0.01641	0.01145	0.00484	0.00148	0.00024
DDS w/o Dynamic M ($\epsilon = 0$)	0.01609	0.00778	0.00385	0.00088	0.00017
DDS	0.01636	0.00945	0.00469	0.00119	0.00025

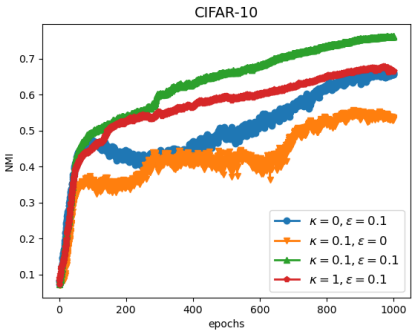

Figure 5. DDS(10%) + ProPos clustering NMI over CIFAR-10, using a ResNet-18 as backbone.

this reconstruction task, the Binary Concrete distribution is useless. A similar effect occurs when removing the dynamic M variation provided in Eq. 6 ($\epsilon = 0$). In this case, the results suggest that using it may be counterproductive.

In contrast, Fig. 5 shows the NMI clustering results when training DDS(10%) + ProPos over 1000 epochs, using a ResNet-18 as the backbone. In this case, the default configuration achieves the best results. The result when removing the dynamic M variation is of special interest, since not only obtain the worst results, but also provide a more unstable output. The same problem arises when no Binary Concrete distribution is included in the training procedure ($\kappa = 0$).

Since the aim of this work to provide a single set of useful hyperparameters, no matter which type of problem needs to be solved, the default configuration $\kappa = 0.1$, $\epsilon = 0.1$, $\zeta = 1$, $\gamma = 0$ obtains good solutions in both experiments. However, the results can be improved if it is carefully tuned for a specific task.

5. Discussion

This paper presents a general recipe for Dynamic Feature Selection in unsupervised scenarios. The presented DDS module can be attached to the input of any architecture tackling an unsupervised task. The module consists of an autoencoder-like architecture that outputs the selection of, at most, M relevant features with their respective score, with M being a fixed parameter tuned by the operator. Our experiments show that DDS can perform data compression with better results than the alternatives, even when accounting that extra memory is needed for saving the feature selection indexes. This improvement is caused by two factors: first, the information provided by the selected features is extremely discriminative; and second, DDS allows the use of more complex downstream architectures since the input data structure is always preserved. Finally, we show that the DDS architecture can be attached to two different architectures tackling very different problems improving the performance in both cases.

5.1. Limitations.

When using the DDS module on an existing architecture, the training procedure must be adapted. The number of epochs often needs to be doubled (compared to training the same architecture without the DDS module).

It is worth noting that the output of the DDS architecture is not forced to be binary. In fact, preliminary studies show that the feature importance score rarely reaches the perfect score of 1. This reduces explainability, as the stored compressed data is modified from the original. However, this can be solved by storing the input data and their importance scores separately, although this almost doubles the memory requirements. If a small memory footprint is a requirement, the DDS output can be forced to be binary by introducing more restrictions into the model, although initial tests show significant degradation in performance.

5.2. Future Work.

As feature work we plan to take advantage of the ability of the DDS architecture to preserve the input data structure. Novel contrastive learning loss functions can be derived from this idea, as the selected pixels of an image should be similar no matter how many geometrical operations are applied to perform data augmentation over them. Finally, it would also be interesting to extend this architecture to the supervised scenario, as in previous DFS algorithms.

Impact Statement

Our Dynamic Feature Selection framework enhances the interpretability of machine learning models by identifying the most relevant components of each input sample. This capability can deepen trust in data-driven solutions, as decision pathways become more transparent and directly traceable to their most essential inputs. In addition, the method's flexible design facilitates integration with diverse unsupervised tasks, from clustering to generative modeling, broadening its potential impact in both academic research and real-world deployments.

References

Arik, S. Ö. and Pfister, T. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6679–6687, 2021.

Balın, M. F., Abid, A., and Zou, J. Concrete autoencoders: Differentiable feature selection and reconstruction. In *International conference on machine learning*, pp. 444–453. PMLR, 2019.

Cancela, B., Bolon-Canedo, V., and Alonso-Betanzos, A. E2e-fs: An end-to-end feature selection method for neural networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 45(07):8311–8323, 2023.

Chen, J., Song, L., Wainwright, M., and Jordan, M. Learning to explain: An information-theoretic perspective on model interpretation. In *International conference on machine learning*, pp. 883–892. PMLR, 2018.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Covert, I. C., Qiu, W., Lu, M., Kim, N. Y., White, N. J., and Lee, S.-I. Learning to maximize mutual information for dynamic feature selection. In *International Conference on Machine Learning*, pp. 6424–6447. PMLR, 2023.

Dang, Z., Deng, C., Yang, X., Wei, K., and Huang, H. Nearest neighbor matching for deep clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13693–13702, 2021.

Dhal, P. and Azad, C. A comprehensive survey on feature selection in the various fields of machine learning. *Applied Intelligence*, pp. 1–39, 2022.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Ha, D. and Schmidhuber, J. World models. *CoRR*, abs/1803.10122, 2018. URL http://arxiv.org/abs/1803.10122.

Hansen, N. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016. URL http://arxiv.org/abs/1604.00772.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. URL http://arxiv.org/abs/1706.08500.

Hou, X., Shen, L., Sun, K., and Qiu, G. Deep feature consistent variational autoencoder. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 1133–1141. IEEE, 2017.

Huang, Z., Chen, J., Zhang, J., and Shan, H. Learning representation for clustering via prototype scattering and positive sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Izmailov, P., Kirichenko, P., Gruver, N., and Wilson, A. G. On feature learning in the presence of spurious correlations. *Advances in Neural Information Processing Systems*, 35:38516–38532, 2022.

Khosla, A., Jayadevaprakash, N., Yao, B., and Fei-Fei, L. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

Kingma, D. P. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

Kingma, D. P., Salimans, T., and Welling, M. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934, 2016. URL http://arxiv.org/abs/1606.04934.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Li, J., Zhou, P., Xiong, C., and Hoi, S. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, 2020.

Li, Y., Hu, P., Liu, Z., Peng, D., Zhou, J. T., and Peng, X. Contrastive clustering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8547–8555, 2021.

Li, Y., Yang, M., Peng, D., Li, T., Huang, J., and Peng, X. Twin contrastive learning for online clustering. *International Journal of Computer Vision*, 130(9):2205–2221, 2022.

Liyanage, Y. W., Zois, D.-S., and Chelmis, C. Dynamic instance-wise joint feature selection and classification. *IEEE Transactions on Artificial Intelligence*, 2(2):169–184, 2021.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018.

Panda, P., Kancheti, S. S., and Balasubramanian, V. N. Instance-wise causal feature selection for model interpretation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1756–1759, 2021.

Perera, P. and Patel, V. M. Learning deep features for one-class classification. *IEEE Transactions on Image Processing*, 28(11):5450–5463, 2019.

Peters, J. and Schaal, S. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

Roffo, G., Melzi, S., Castellani, U., Vinciarelli, A., and Cristani, M. Infinite feature selection: a graph-based feature filtering approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4396–4410, 2020.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Shah, C., Du, Q., and Xu, Y. Enhanced tabnet: Attentive interpretable tabular learning for hyperspectral image classification. *Remote Sensing*, 14(3):716, 2022.

Shen, Y., Shen, Z., Wang, M., Qin, J., Torr, P., and Shao, L. You never cluster alone. *Advances in Neural Information Processing Systems*, 34:27734–27746, 2021.

Tang, Y., Zhang, L., Min, F., and He, J. Multiscale deep feature learning for human activity recognition using wearable sensors. *IEEE Transactions on Industrial Electronics*, 70(2):2106–2116, 2022.

Tao, Y., Takagi, K., and Nakata, K. Clustering-friendly representation learning via instance discrimination and feature decorrelation. In *International Conference on Learning Representations*, 2020.

Tomczak, J. and Welling, M. Vae with a vampprior. In *International conference on artificial intelligence and statistics*, pp. 1214–1223. PMLR, 2018.

Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., and Gelly, S. Towards accurate generative models of video: A new metric & challenges. *CoRR*, abs/1812.01717, 2018. URL http://arxiv.org/abs/1812.01717.

Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., and Van Gool, L. Scan: Learning to classify images without labels. In *European conference on computer vision*, pp. 268–285. Springer, 2020.

Yang, J., Lindenbaum, O., and Kluger, Y. Locally sparse neural networks for tabular biomedical data. In *International Conference on Machine Learning*, pp. 25123–25153. PMLR, 2022.

Yoon, J., Jordon, J., and van der Schaar, M. Invase: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2018.

Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D., and Saeed, J. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, 1(2):56–70, 2020.

Zhong, H., Wu, J., Chen, C., Huang, J., Deng, M., Nie, L., Lin, Z., and Hua, X.-S. Graph contrastive clustering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9224–9233, 2021.

A. World Models: A Detailed Overview

In model-based reinforcement learning (RL), the term *world model* typically refers to the combination of two main components that together learn a generative representation of the environment:

- **Vision Model (V):** Compresses each high-dimensional observation (e.g., an image) into a lower-dimensional latent vector $\mathbf{z}_t$. A common choice for this part is a Variational Autoencoder (VAE), which learns both an encoder and a decoder. The encoder maps images to latent representations, and the decoder is capable of reconstructing or "imagining" frames purely from latent codes.
- **Memory Model (M):** Captures the *temporal* dynamics of the environment in latent space. Typically, this involves training a recurrent model (e.g., an RNN) that outputs a Mixture Density Network (MDN) predicting the next latent vector $\mathbf{z}_{t+1}$ given the current latent $\mathbf{z}_t$, the agent's action $\mathbf{a}_t$, and the hidden state $\mathbf{h}_t$. Formally:
$$\mathbf{z}_{t+1} \sim \mathcal{P}(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t),$$
where $\mathcal{P}$ is modeled by the parameters of the mixture components output by the MDN-RNN.

Once trained, these two modules—the Vision Model and the Memory Model—constitute the *world model*. They allow an agent to generate sequences of predicted future latent states, which can be decoded back into image space if desired. In essence, the Vision Model provides spatial compression and reconstruction, while the Memory Model predicts how these compressed representations evolve over time, effectively simulating environment dynamics in a more manageable latent space.

A.1. Controller

Although crucial for decision-making, the *Controller* (or policy) lies *outside* the world model itself. The Controller uses information from the latent state $\mathbf{z}_t$ and the RNN hidden state $\mathbf{h}_t$ to decide which action $\mathbf{a}_t$ to take. In the specific setting we consider here, the Controller is a simple neural network (without a hidden layer) of the form:

$$\mathbf{a}_t = W_c \begin{bmatrix} \mathbf{z}_t \\ \mathbf{h}_t \end{bmatrix} + \mathbf{b}_c,$$

where $\mathbf{z}_t \in \mathbb{R}^d$ is the current latent state, $\mathbf{h}_t$ is the hidden state of the RNN, and W_c, $\mathbf{b}_c$ are learnable parameters. In practice, the Controller can be optimized to maximize returns using a variety of standard algorithms (e.g., evolutionary strategies, policy gradients). In our experiments, it is trained on *real environment* rollouts (i.e., no synthetic data is used to train the Controller).

A.2. Why World Models?

Sample Efficiency A key motivation behind world models is *sample efficiency*. By learning a generative model of the environment, the goal is to create an internal simulation that is so realistic that the controller can be trained solely within these internal "dreams." This approach would be highly efficient because it eliminates the need to process real images, relying instead on compact latent representations. Developing robust internal models can still yield significant benefits. These include enhanced representations for downstream decision-making and potential improvements in how quickly the agent can learn from real-world samples.

Partial Observability and Model Imperfection Because the world model only "sees" what is encoded in the Vision Model's latent vectors, it may miss unobserved or unmodeled factors that influence the true state. If the Memory Model or the Vision Model are poorly learned (e.g., due to insufficient data or training instability), the latent transitions and reconstructions will deviate from reality. Despite these challenges, well-trained world models often provide a powerful abstraction that can simplify policy learning.

A.3. Training Procedure

A conventional workflow for building and using a world model can be summarized as follows:

1. **Data Collection:** Gather trajectories of observations and actions using a random or exploratory policy in the real environment.
2. **Train the Vision Model:** Fit an autoencoder (e.g., a VAE) on the collected frames so that each image $\mathbf{x}_t$ is mapped into a latent vector $\mathbf{z}_t$, and the model can reconstruct $\mathbf{x}_t$ from $\mathbf{z}_t$.
3. **Train the Memory Model:** Use latent sequences $\{(\mathbf{z}_t, \mathbf{a}_t)\}$ to train a recurrent network that outputs mixture density parameters. At each time step t, it predicts a distribution over possible next latents $\mathbf{z}_{t+1}$.
4. **Train the Controller:** Employ the (fixed) Vision and Memory Models to encode real environment observations into latents, and update the Controller's parameters to maximize an RL objective (e.g., via CMA-ES (Hansen, 2016)).

Because the environment dynamics are approximated by the Memory Model directly in latent space, the agent can generate short- or medium-horizon predictions. In some setups, these predictions can be used for planning or to reduce real-environment interactions. In our setting, the Controller is trained solely with real-environment rollouts,

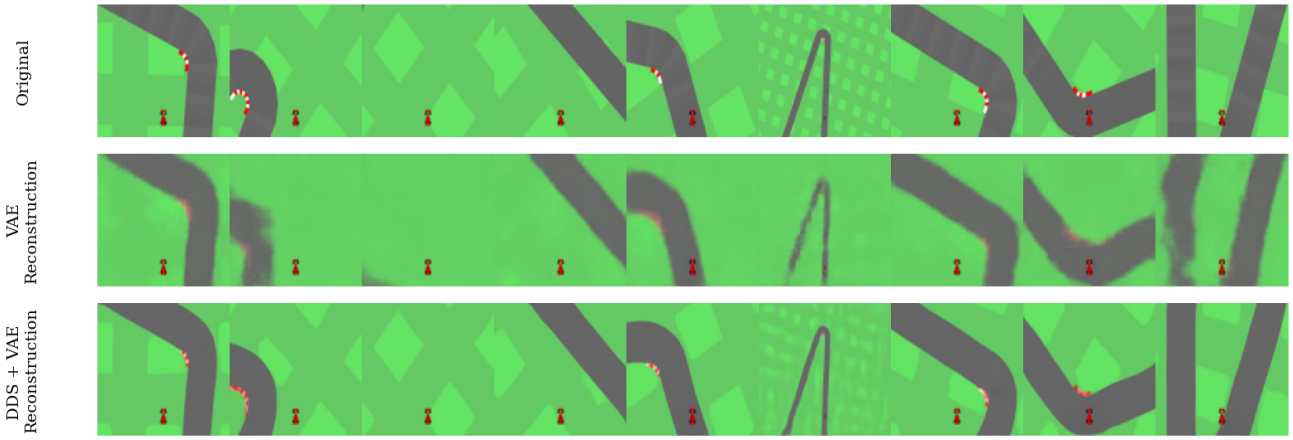

Figure 6. comparison of images reconstructed through a VAE such as (Ha & Schmidhuber, 2018) and our model with the same number of trainable params and latent space size $z \in \mathcal{R}^{32}$.

even though *in principle* the world model could be used for additional hypothetical scenarios.

A.4. Beyond Reinforcement Learning: Real-Time Game Generation

While originally introduced as a model-based RL strategy, world models have recently gained traction in domains *beyond* direct policy learning—particularly in the game-generation community. Here, the world model's generative capacity is harnessed as a form of *real-time game engine*, where game levels or scenarios can be dynamically created through latent-space rollouts. By sampling how states evolve over time using the Memory Model and then decoding them back to an observable format (e.g., 2D or 3D graphics), developers and researchers can produce procedurally generated worlds that respond to player actions in a highly adaptive manner. This "world model as a game engine" paradigm enables unique forms of content creation and interactive storytelling, blending the boundaries between model-based RL and creative generative applications.

B. Visualization

In this section, we present a series of image reconstruction results generated by our proposed DDS+VAE model and compare them to those produced by the original VAE architecture (see Figure 6). The comparison highlights the enhanced reconstruction quality, including sharper image details over the standard VAE.

Furthermore, Figure 7 showcases some of the internal representations learned by the DDS+VAE model. These shows how DDS+VAE captures more nuanced features and structures within the data.

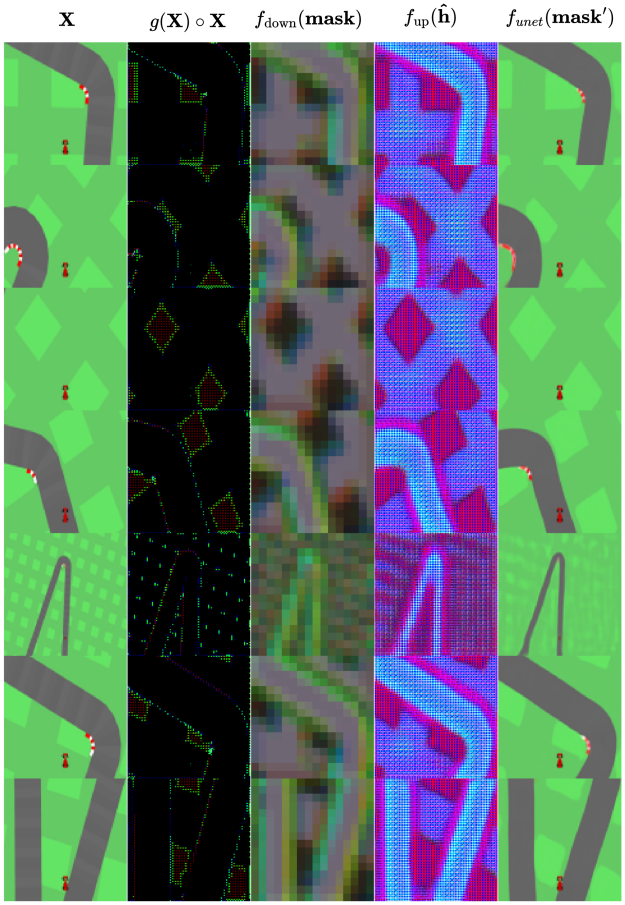

Figure 7. Visualization of intermediate image of our DDS+VAE model trained with $M = 3\%$