6-13-2023

A Deep Hierarchical Variational Autoencoder for World Models in Complex Reinforcement Learning Environments

Sriharshitha Ayyalasomayajula
Portland State University

A Deep Hierarchical Variational Autoencoder for World Models in Complex Reinforcement Learning Environments

by

Sriharshitha Ayyalasomayajula

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

Thesis Committee:
Banfsheh Rekabdar, Chair
Wu-chi Feng
Atul Ingle

Portland State University
2023

Abstract

Model-based reinforcement learning (MBRL) approaches leverage learned models of the environment to plan and make optimal decisions, reducing the need for extensive real-world interactions and enabling more efficient learning in complex domains such as robotics, autonomous systems, and resource allocation problems. They also provide interpretability and insight into the underlying dynamics, facilitating better decision-making and system understanding.

The world model is a model-based RL approach that employs generative neural network models to learn a compressed spatial and temporal representation of the environment. This work explores world models and a simple single-layered RNN model to learn a simple policy based on the representations to solve tasks in complex RL environments. A traditional variational autoencoder (VAE) encodes environment features to latent representations in the world model approach. Recent research on generative models reveals that traditional VAE constraints cause information loss or distortion during compression and impede the world model based- agent's ability to learn accurate representations of complex environments. This thesis proposes a deep hierarchical variational autoencoder (NVAE) as the visual component of the world model to overcome the challenge of modeling complex data and long-range correlations and improve an agent's performance in complex RL environments such as car racing-v2 and panda-gym.

Dedication

Dedicated to Mum and Dad.

Acknowledgements

I am truly grateful to Dr. Rekabdar for her invaluable guidance, unwavering support, and continuous encouragement throughout my thesis journey. As my advisor, she has been an exceptional teacher and mentor and a source of inspiration and motivation. It has been an honor and privilege to be under her tutelage, and I will forever cherish the knowledge and skills gained through her guidance.

I sincerely thank Dr. Feng and Dr. Ingle for graciously agreeing to serve as my committee members. I appreciate your time and effort in reviewing my work and offering valuable feedback.

I am deeply grateful to my graduate advisor, Ella Barrett, for her support, encourgaement, and insightful feedback throughout my master's journey.

I sincerely thank Shayan and Bahareh, my exceptional lab mates, for their valuable feedback and support. It has been a pleasure to work alongside both of you.

I want to express my gratitude to the talented members of BlackPink for their exceptional music, which played a vital role in motivating me to surmount obstacles throughout my thesis journey. Additionally, I extend my deepest gratitude to Yu Jae Seok and Lee Kwang Soo of 'Running Man' for their unparalleled ability to elicit endless laughter and provide a much-needed respite during the demanding process of writing this thesis. I am thankful for their music, comedic talents, and positive impact on my life, granting me the necessary balance and peace of mind to face the challenges in my thesis journey.

I am incredibly fortunate to have the love and support of my dear friends, who have consistently placed their faith in my abilities. I want to thank Srija, Anvitha, Saiteja, Sunayana, and Saba for their incredible patience, understanding, and support throughout my master's journey. I am equally grateful to my friends in India—Manisha, Manasa, Ayesha, and Rashmitha—for their unwavering support and love. Furthermore, I

want to express my profound appreciation to my high school mentors - Ramakrishna sir, Basha sir, and Srikanth sir for recognizing my potential and encouraging me.

Lastly, I want to express my heartfelt gratitude to my family. I am deeply grateful for the invaluable guidance, constant support, boundless love, and immense patience shown by my sister and brother-in-law, Priya, and Srikanth. Their constant presence and support have been an inspiration throughout my master's journey. Moreover, I am indebted to my parents for their unwavering faith, trust, love, and support, without which these accomplishments would not have been possible. Their patience, consistent reminders of my potential, support, and unconditional love have been the pillars of my success. This thesis is a humble dedication to my parents.

I want to express my gratitude to all the individuals who have been a driving force, inspiring and motivating me on my thesis journey.

Table of Contents

List of Tables

List of Figures

List of Equations

Glossary of Abbreviations

Reinforcement learning (RL) – a type of machine learning approach where an agent learns to make optimal decisions by interacting with an environment and obtaining rewards or punishments as feedback.

Model-Based Reinforcement learning (MBRL) – a type of reinforcement learning approach where an agent learns a model of the environment's dynamics to plan and make optimal decisions by integrating experience-based learning and reasoning about the environment.

World Model – a model-based reinforcement learning approach that uses generative models to learn the environment's features, and to the learn internal model based on environment's dynamics. It also consists of a simple neural network for decision making. This model is applied to solve complex problems like video games, self-driving cards, robotics etc.

Visual Model (V), Memory model (M) and Controller model (C) – The three components of the world model.

Recurrent Neural Network (RNN) - an artificial neural network that processes sequential data by utilizing feedback connections to maintain and propagate information across previous time steps.

Variational Autoencoder (VAE) – a generative model that learns to encode and decode data by combining an encoder-decoder architecture with probabilistic inference, allowing it to generate new samples and perform efficient latent space interpolation.

Nouveau Variational Autoencoder (NVAE) - a variation of variational autoencoder that uses a deep hierarchical architecture with deep convolutions and residual cell to generate new samples and perform efficient latent space interpolation.

Mixture Density Network based Recurrent Neural Network (MDN-RNN) - a recurrent neural network architecture that combines the modeling power of RNNs with

mixture density networks to capture complex probability distributions and generate output sequences.

Chapter 1: Introduction

How do humans react instinctively to dangerous situations? How does the brain quickly react and suggest the following action in dangerous situations? The answer to these questions is a mental model. Humans construct a mental model based on the data provided by their five senses (Quiroga et al., 2005). This mental model influences their behavior and actions. Our brain learns abstract representation based on the spatial and temporal data of the world to manage the extensive data we encounter daily. We take in the visual cues and retain an abstract scene description. Study in the neural sciences suggests that - what we see at any time may also be determined by our brain's prediction of the future based on our internal model (Nortmann et al., 2015) .

Suppose we wish to understand the brain's predictive model. In that case, we should perceive it as a model for predicting future sensory data based on our current motor actions rather than a generalized model for predicting the future (Keller et al., 2012). We use this predictive model to react quickly and reflexively in dangerous situations without consciously planning actions (Mobbs et al., 2015).

For example, consider the sport of baseball. A batter's decision on how to swing the bat is made in a fraction of a second, far less than the time it takes for visual data to reach our brains. For professional players, the ability to anticipate where and when the ball will travel is innate and happens subconsciously (Hirshon et al., 2013). Their internal models predict the optimal time and location to swing the bat, and their muscles execute it automatically (Maus et al., 2013). As a result, their foresight is used for immediate action without systematically considering various scenarios.

Recent research in neural networks has provided the means to replicate the brain's predictive model for solving reinforcement learning-based problems (Kaelbling et al., 1996) using a neural network-based predictive model. In many reinforcement learning (RL) problems, a recurrent neural network (RNN) is used as the predictive model

(Werbos, 1989). RL research focuses on agent behavior in complex environments, and game environments are extensively utilized to test and assess agent performance.

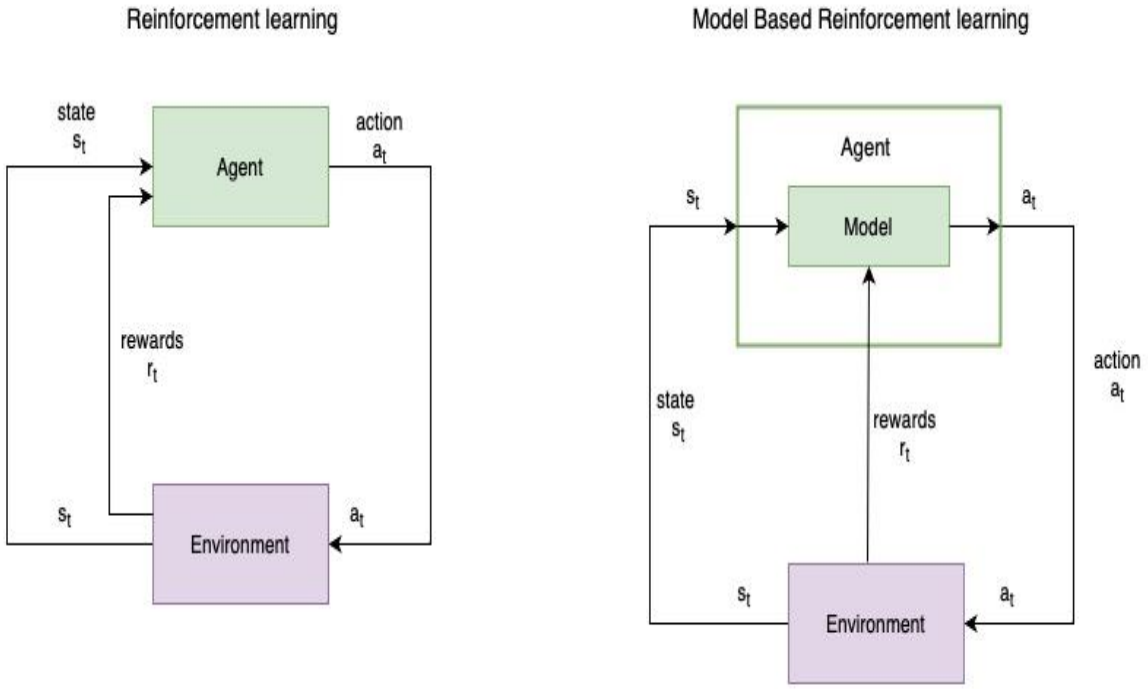

Figure 1:The figure depicts two flow charts; (left) illustrates the general reinforcement learning process, and (right) demonstrates the workings of model-based reinforcement learning.

In RL systems, the agent benefits from learning a model of the environment, which is a function that forecasts future state transitions and rewards. Before taking any action in the environment, a solution can is determined using traditional RL techniques if a model is available, i.e., if all the elements of the MDP (Markov Decision Process) are known, mainly the transition probabilities and the reward function. This is known as model-based reinforcement learning (as seen in Fig 1). Model-based reinforcement learning (MBRL) has become a promising approach for solving tasks in complex environments such as

games and robotics. MBRL has garnered attention by solving the Go game with the AlphaGo approach and car racing-v0 with world models.

This thesis explores the world model (Ha & Schmidhuber, 2018), a simple framework that uses generative neural network models to learn a compressed spatial and temporal representation of the environment, and a simple single-layered RNN model to learn a simple policy based on the representations to solve tasks in the complex RL environments. World models simulate the environment's behavior, allowing agents to plan and make decisions without directly interacting with the actual environment.

In previous RL algorithms, credit assignment problems often bottlenecked training large network models. Hence the world model framework employs smaller networks to learn a simple policy to solve a task. Here, the agent is divided into one large world model with visual (V) and memory (M) components and a small controller (C) model. A traditional Variational Autoencoder (VAE) serves as the visual component in the world model to learn a compressed abstract representation from a high-dimensional input data frame. A large RNN trained with a back-propagation algorithm is used as the memory (M) component to predict the future latent vectors. A small RNN is employed as the controller model responsible for determining the course of actions to take to maximize the expected cumulative reward of the agent in an environment. Here, V and M are trained together, keeping the computational complexity within the world model. C is trained independently from the world model so that it focuses on solving the credit assignment problem.

In contrast to the traditional RL algorithms, the world model approach takes in a stream of raw RGB pixel image data and directly learns the spatial-temporal representations. Therefore, the latent representations learned are supposed to be more accurate.

Earlier implementation of the world model approach was to solve the car racing-v0 task where the agent achieved benchmark results. The world model framework is an important model-based approach because the policy learned in the actual environment can be used in a hallucinated/dream environment, where the dream environment is generated based on the observed latent representations of the agent.

In the case of the car racing-v0 environment, we observed that the agent could navigate through the environment. However, its performance was much less than the agent's performance in the actual environment. This is because the features of the dream environment do not match those of the actual environment on which the policy used to train the agent in the dream environment is based.

Like the actual environment, to solve a task in the dream environment, the agent must collect maximum cumulative rewards by taking precise actions; in the case of a car racing-v0 environment, the agent should stay on track and navigate the turns for maximum time. Therefore, the controller's actions need to be precise, and the latent input representations learned by the visual component must be accurate.

A recent study in the generative models indicate the limitations of traditional VAE (Kingma & Welling, 2013) in modeling complex data and long-range correlations. Long-range correlations in the data refer to the interdependencies between distant pixels or regions within an image. When long-range correlations are present in the data (which is the case for most data), the encoding process by the visual component (VAE) results in information loss or distortion. It impedes the world model-based agent's ability to learn accurate representations of complex environments.

To address this challenge, this thesis, proposes using a deep hierarchical variational autoencoder (NVAE) as the visual component in the world model that helps enhance an agent's performance in complex RL environments such as robotics. Nouveau VAE (NVAE) is a deep hierarchical VAE with multiple levels of latent variables, allowing for more expressive and flexible modeling of complex data distributions (Vahdat & Kautz, 2020). NVAE addresses the limitations of traditional VAE, i.e., low modeling capacity and difficulty in modeling complex data distributions by utilizing depth-wise separable convolutions with residual cells. The hierarchical multi-scale model captures global long-range correlations at the top and local dependencies at lower levels addressing the issue of long-range correlations in data.

The main contributions of the thesis are –

1) This thesis introduces a novel framework called NVAE-based World Models, a modified version of the traditional world models to enhance agent's performance in complex RL environments.

2) This work highlights the significance of a hierarchical variational autoencoder (VAE) in enhancing the expressiveness and capability of traditional world models, allowing them to more accurately model complex data distributions and capture diverse and intricate patterns in the environment.

3) To the best of my knowledge, this work is one of the first successful attempts to apply a model-based reinforcement learning approach to the complex panda-gym (reach task) environment, and the results show promising performance.

The remainder of the thesis is structured as follows: Chapter 2 addresses recent work in VAE, the architectures of VAE, hierarchical VAE, and the world model approach. Chapter 3 describes the proposed method, the NVAE-based world model, and the proposed framework's components - vision, memory, controller, and architecture. Chapter 4 presents the training processes and evaluates the results of the RL agent in the car racing-v2, dream car racing-v2, and robotics environments. Chapter 5 is a summary of all the work done in this thesis.

Chapter 2: Background

This chapter provides a concise introduction to the technical methods employed in this work and a brief overview of the most recent and relevant literature related to the thesis. The chapter is structured as follows. Section 2.1 introduces the core framework called the world model. Section 2.2 introduces the generative model, variational autoencoder (VAE), and describes its architecture. Section 2.3 introduces the variation of variational autoencoder, hierarchical variational autoencoder (HVAE), and its architecture. Finally, Section 2.4 briefly reviews recent and relevant research in the hierarchical variational autoencoders (HVAE) employed in this thesis.

2.1 World Model

This thesis uses world models as the main framework for training RL agents to perform tasks in complex environments. The world model (Ha & Schmidhuber, 2018) is a model-based reinforcement learning approach (MBRL), where it creates an internal model (based on the human mental model) to represent the environment and its dynamics. This model is designed to capture the underlying structure and patterns of the environment. The model learns latent representations, which are compressed and abstract representations of the environment, from the available data. These learned latent representations are then used for policy training. Policy training refers to training an agent or system to make decisions and take actions based on available information. By using the latent representations learned by the model, the policy training process becomes more efficient in terms of data usage. This means that the model can achieve good performance with less training data compared to other approaches.

The world model approach employs generative models to learn and compress the spatial and temporal representations of the environment to a latent representation. For ease of computation, the model is split into one large world model [with VAE as the visual

sensory component (V) and an MDN-RNN as the memory component (M)] and a small controller model (as seen in Figure 2). The visual model (V) is a variational autoencoder (VAE) (Kingma & Welling, 2013) that learns compact latent representations of input data (such as images). The memory model (M), a recurrent neural network (RNN) with mixture density networks, captures temporal dependencies in the latent space. The controller (C) is a simple single-layered neural network that decides the actions based on the input from the visual and memory models.

The world model's visual model (V) learns to encode input data into a lower-dimensional latent space, generating new data samples. The memory model (M) captures the temporal dependencies in the data by modeling the dynamics of the latent representation over time using an RNN. The controller (C) is trained to use the latent representation and the memory to make decisions.

The latent vector z_t represents a compressed and abstract representation of the environment or world state at a specific timestep t. It is a fundamental aspect of the model's internal representation. The latent vector z_t often captures the environment's underlying dynamics, which holds information about the pertinent features and variables. It can include factors such as position, velocity, orientation, and other relevant attributes that are important for understanding and predicting the future states of the environment.

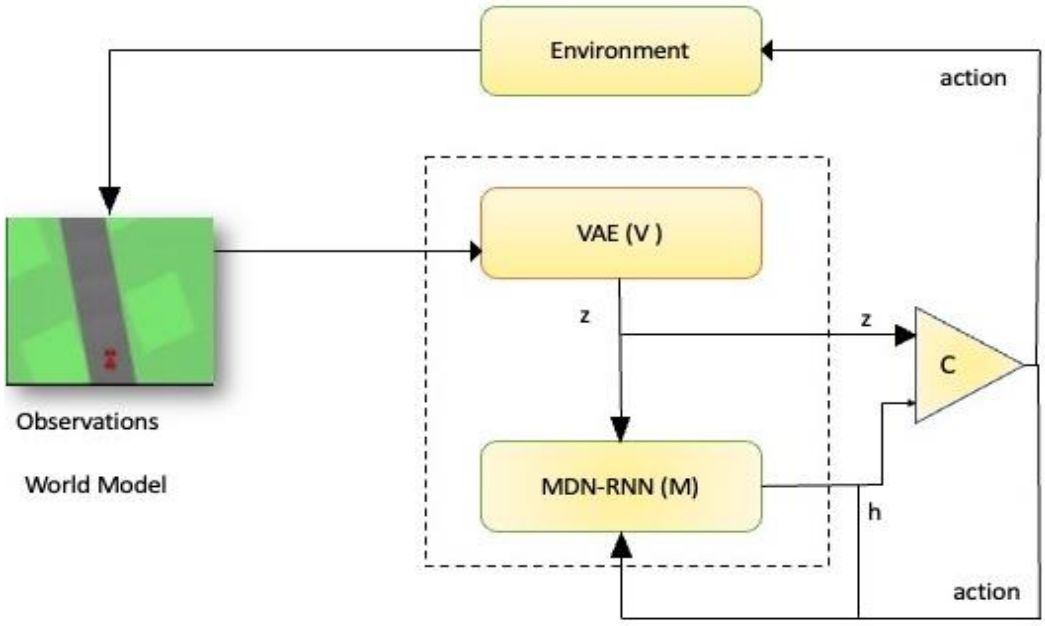

Figure 2: The figure illustrates the interaction of V, M, and C components with the car racing-v0 environment.

Here, the raw observations of the environment, a 2d-image, are first processed by V at each time step t to produce the latent representation z_t. M is trained to output a probability density function p(z) of the next latent vector z_{t+1} by approximating it as a mixture of Gaussian distribution. C takes the input of z_t from V and the hidden states h_t from M to output an action vector a_t for motion control that will affect the environment. M updates its hidden state by taking the current z_t and action a_t as input.

The world model approach provides a unified framework for generative modeling and control by leveraging the best features of variational autoencoders (VAEs) for learning representations, recurrent neural networks (RNNs) for modeling temporal dependencies, and controllers for decision-making.

2.2 Variational Autoencoder

The variational autoencoder (VAE) (Kingma & Welling, 2013) is a generative model parameterized by a neural network θ. It is defined by an observed variable x, which depends on the hierarchy of stochastic latent variables $z = z_1, \ldots, z_L$ such that:

$$p_\theta(x, z) = p_\theta(x|z_1)\, p_\theta(z_l)\, q_{l-1}\, p_\theta(z_i|z_{i+1})$$

Equation 1

The distributions $p_\theta(z_i|z_{i+1})$ over the latent variables in the VAE are usually modeled as Gaussians with diagonal covariance, where the parameters depend on the previous latent variable in the hierarchy. The top latent variable $p_\theta(z_l)$ is generally modeled as a Gaussian with mean 0 and identity covariance matrix (N (z_L; 0,1)). The likelihood $p_\theta(x|z_1)$ is typically modeled as a Gaussian distribution for continuous data or a Bernoulli distribution for binary data. The figure 3 illustrates the architecture of a traditional variational autoencoder.

The main objective of learning the parameters of VAE is to maximize the marginal log-likelihood over the training data.

$$p_i \log p_\theta(x_i) = p_i \log R\, p_\theta(x_i, z_i)\, dz_i$$

Equation 2

Variational Inference with posterior approximation $q_\varphi(z|x)$ parameterized by a neural network φ is used to improve the expressivity of the model while training with complex data distributions. Later, Jensen's inequality is applied to derive the evidence lower bound (ELBO), which is a lower bound to the integral in the marginal likelihood and is a function of the variational approximation $q_\varphi(z|x)$ and the generative model $p_\theta(x, z)$:

$$\log p_\theta(x) \geq \varepsilon\, q_\varphi(z|x) \log\, q_\varphi(z|x) \equiv L(\theta, \varphi)$$

Equation 3

Stochastic backpropagation and the reparameterization method are used to optimize the parameters θ and φ, enabling the employment of gradient ascent algorithms with low variance gradient estimators.

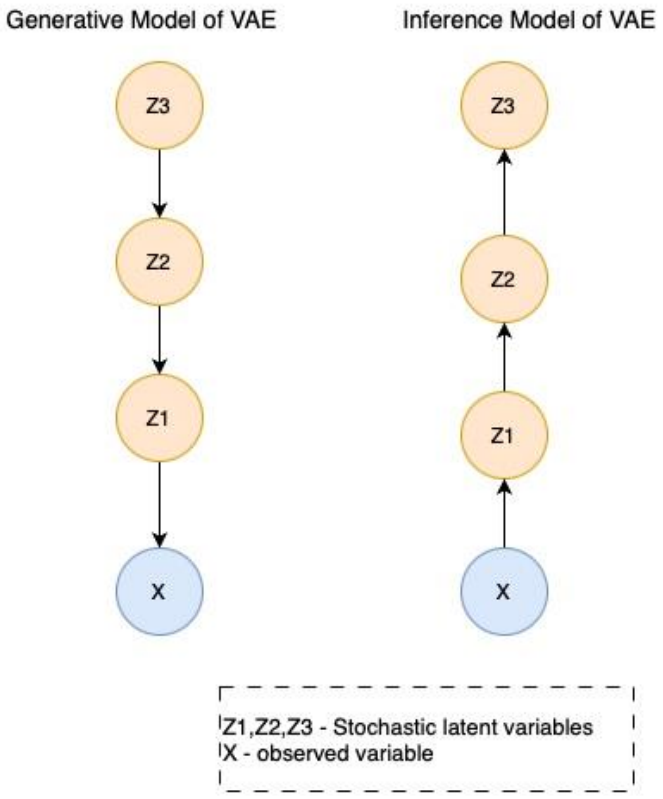

Figure 3: (Left) Generative model of Variational autoencoder (VAE). (Right) Inference model of Variational auto encoder.

Bottom-up factorization of the variational approximation ($q(z|x) = q(z1|x)\, q_{L1}\, q(zi+1|zi)$) is used in a VAE to condition each latent variable z_i on the variables below it in the hierarchy. For efficient computing, all the factors in the variational approximation are assumed to be gaussians whose mean and diagonal covariance are set by neural networks.

2.3 Hierarchical VAE

Previous work on variational autoencoders (VAEs) often used fully factorized gaussian distributions for both the approximate posterior $q_\varphi(z|x)$ and the prior $p_\theta(z)$, resulting in suboptimal outcomes for generating high-quality samples using latent variables of complex data distributions. A hierarchical VAE, which incorporates many stochastic layers of latent variables, is one technique to increase expressivity in both distributions (Klushyn et al., 2019). These latent variables are conditionally dependent on one another and released in groups z_0, z_1.... z_N.

During image processing, latent variables are rendered as feature maps of varying resolutions, where z_0 represents a low-resolution set of latent variables at the network's top, and z_N represents a high-resolution set of latent variables at the bottom of the network.

The top-down VAE, where both the prior and the approximation posterior generate latent variables in the same order, is one of the well-conditioned structures for hierarchical VAEs:

$$p_\theta(z) = p_\theta(z_0)\, p_\theta(z_1|z_0) \ldots\ldots\ldots p_\theta(z_N|\, z < N)$$

Equation 4

$$q_\varphi(z|x) = q_\varphi(z_0|\, x)\, q_\varphi(z_1|z_0, x) \ldots\ldots q_\varphi(z_N|\, z < N, x)$$

Equation 5

The encoder generally performs a deterministic "bottom-up" pass on the data to generate features, then performs a top-down run on the groups of latent variables to approximate the posterior. The hierarchical architecture can be seen in Figure 4.

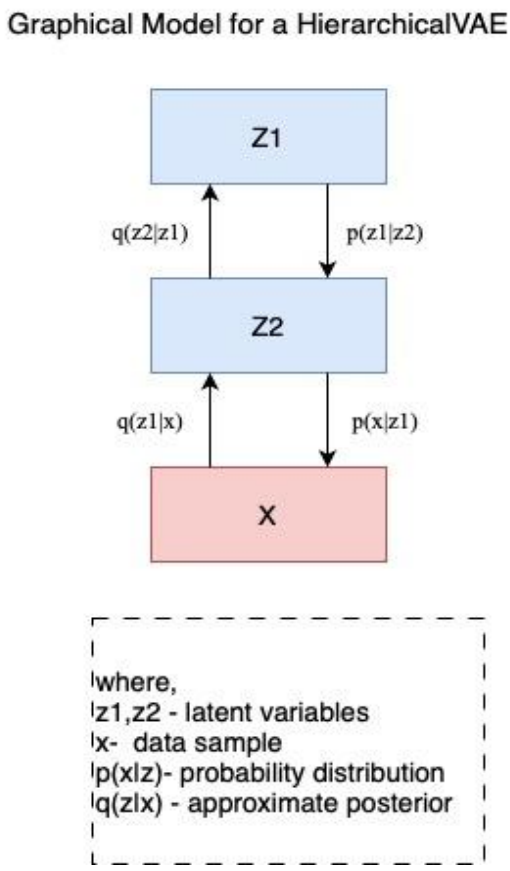

Figure 4: The figure illustrates a graphical model for hierarchical variational autoencoder (HVAE).

The residual blocks in this design are analogous to the ResNet bottleneck blocks. An application of the GELU nonlinearity precedes each convolution. Both the approximate posterior $q\varphi(.)$ and the prior $p\theta(.)$ are assumed to follow a diagonal Gaussian distribution. The latent variable z is sampled from $q\varphi(.)$ during training, but from $p\theta(.)$ during testing. The pooling layer employs average pooling, while the un-pool layer employs nearest neighbor up sampling.

In this approach, the features generated are shared among the approximation posterior, the prior, and the reconstruction network $p(x|z)$, which generates latent variables from the

top down via feedforward networks. This architecture of VAE is more suited for image data generation because it is simple, empirically effective, and hypothesized to resemble biological processes of perception.

2.4 Related Works

This section summarizes the recent work on variational autoencoders for image data generation. The following work inspires the proposed Nouveau VAE's architecture.

Over the past decade, research has been conducted on various generative model techniques, such as normalizing flows, autoregressive models, and variational autoencoders. Normalizing flow-based and autoregressive models outperformed the VAEs. Since VAE has the advantages of fast and traceable sampling and easy access to encoding networks, numerous researchers have begun to look for methods to enhance its performance. The following are instances of recent work performed on VAEs.

2.4.1 IAF-VAE

An IAF-VAE is a generative model that goes beyond the traditional variational autoencoder (VAE) by modeling the approximate posterior distribution with an extra flow-based transformation in the inference network (encoder). An inverse autoregressive flow, which normalizes data, executes the flow-based transformation (Kingma et al., 2016).

Based on the VAE architecture, IAF consists of an encoder (inference network) that approximates the posterior distribution $q(z|x)$ and a decoder (generative network) that models the conditional distribution $p(x|z)$, where x is the observed data and z is the latent variable. To approximate the posterior distribution $q(z|x)$, the VAE encoder uses IAF as a flow-based transformation. IAF is a normalizing flow that uses a series of invertible and

differentiable transformations on the latent variable z to convert it from a simple distribution (such as the Gaussian) to a more complex distribution.

In IAF-VAE, each invertible and differentiable transformation is applied in an autoregressive fashion, where it is dependent on the preceding transformations. Thus, IAF-VAE can capture intricate interdependencies among z's components. The visualized step of inverse autoregressive slow applied to VAE is given in Figure 5.

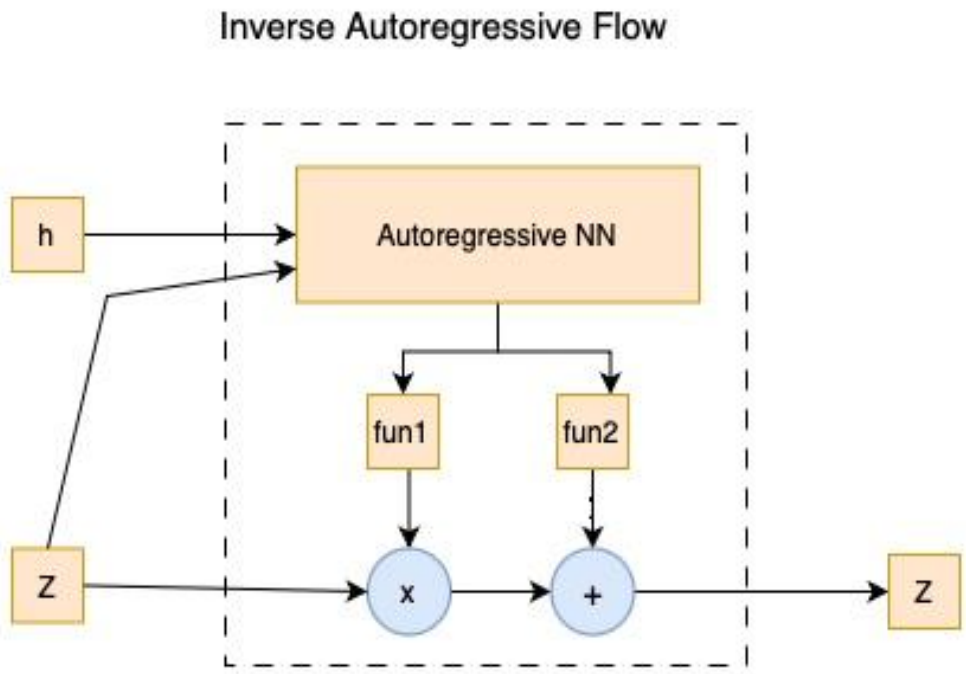

Figure 5: The figure illustrates how the inverse autoregressive flow step is applied in a Variational autoencoder (VAE) architecture to improve the model's performance.

IAF improves the variational inference in VAE by providing a more flexible and expressive approximation of the posterior distribution q(z|x). Modeling complex distributions with the help of autoregressive transformations improves IAF's performance

in capturing the true posterior distribution and producing high-quality samples from the generative model.

To achieve low-variance gradient estimators, IAF is also trained with stochastic gradient descent (SGD) by employing the reparameterization method, like VAE. The encoder parameters are optimized during training to reduce the Kullback-Leibler (KL) divergence between the estimated posterior q(z|x) and the actual posterior p(z|x). On the other hand, the decoder's parameters are adjusted to improve the observed data's likelihood, denoted by p(x|z). Compared to competing generative models, IAF-VAE performed more efficiently on several image generation tasks, including those from CIFAR-10 and ImageNet.

2.4.2 BIVA

Another approach is bidirectional-inference variational autoencoder (BIVA) which utilizes a skip-connected generative model and a bidirectional stochastic inference path in the inference networks (Maaløe et al., 2019). It has a deep hierarchical architecture of latent variables with multiple stochastic layers which is illustrated in Figure 6.

The latent variables in a BIVA model are organized into numerous layers, labeled as z_0, z_1...z_N, where z_0 represents a low-resolution set of latent variables at the network's "top" and z_N represents a higher-resolution set of latent variables at the network's "bottom."

The approximation and prior posterior produce latent variables in the same hierarchical order. For practicality, we will refer to the prior distribution as p(z) and the estimated posterior distribution as q(z|x). These distributions can be factored as follows:

$$p(z) = p(z_0)\, p(z_1|z_0) \ldots\ldots\ldots p(z_N|z_N)$$

Equation 6

$$q(z|x) = q(z_0|x)\, q(z_1|z_0, x) \ldots\ldots\ldots q(z_N|z_N, x)$$

Equation 7

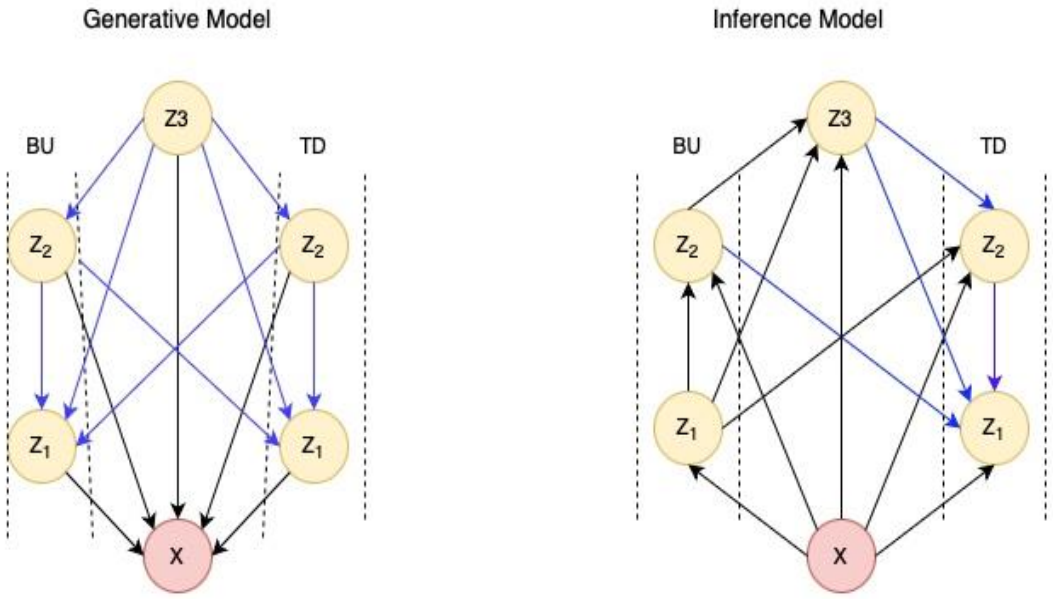

Figure 6: (Left) Generative model of BIVA. (Right) Inference model of BIVA. BU and TD in the diagram refer to bottom-up and top-down information flow. The nodes {z_1, z_2, z_3} represent the latent variables, while x represents the observed data sample.

Feedforward networks are used in BIVA to generate features from the data in a deterministic "bottom-up" pass. These characteristics are shared by the approximate posterior, prior, and reconstruction network p(x|z). Both the latent variables' prior and approximate posterior distributions are modeled as diagonal Gaussian distributions, p(z)

and q(z|x), respectively. Due to their tractability and differentiability, Gaussian distributions are commonly used in VAEs.

Compared to other cutting-edge techniques, the architecture of BIVA effectively disentangles image variation variables like, shape, color, and orientation. When applied to text generation tasks, BIVA-VAE generated paraphrases of sentences with improved diversity and fluency compared to other generative models.

2.4.3 VQ-VAE2

VQ-VAE2 is a deep learning model for image compression and generation that combines two famous architectures: variational autoencoder (VAE) and vector quantization (VQ) (Razavi et al., 2019).

There are three primary parts to VQ-VAE2's architecture: an encoder, a quantizer, and a decoder. Encoder receives an image as input and generates a latent representation (as seen in Figure 7). The quantizer then uses vector quantization to convert the continuous latent representation into a categorical representation by examining the closest vector in a codebook corresponding to each latent space point. The decoder then uses the categorical representation to produce a reconstructed image. The VQ-VAE2 model is trained end-to-end using a loss function that balances the reconstruction and codebook losses. This encourages the quantized representation to match the actual continuous representation. During inference, the codebook remains unchanged and new images are generated using the quantized representation.

The encoders and decoders in the model are deep neural networks. The input to the model is a 256 × 256 image that is compressed into quantized latent maps of sizes 64×64 and 32×32 for the bottom and top levels, respectively. The decoder then reconstructs the image from the two latent maps. The figure visually represents the image compression and reconstruction process using quantized latent maps.

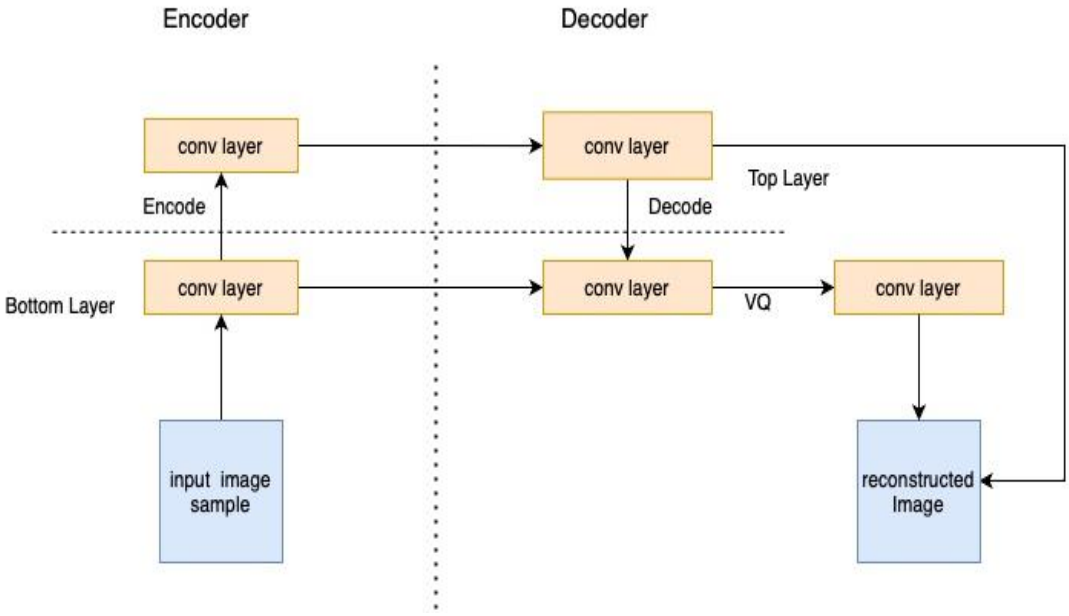

Figure 7: Architecture of a hierarchical Vector Quantized Variational Autoencoder (VQ-VAE2) model.

Compared to earlier models, the VQ-VAE2 model achieves state-of-the-art results in image compression and generation tasks. On the ImageNet dataset, for instance, the VQ-VAE2 model has compressed images at a rate of 3.0 bits per pixel (bpp), which is better than the previous state-of-the-art rate of 3.15 bpp while preserving equivalent image quality. Also, the VQ-VAE2 model has demonstrated promising results for generating high-quality images such as realistic faces and natural landscapes.

Chapter 3: NVAE-based World Model

This chapter describes integrating a deep hierarchical variational autoencoder as the visual model and interacting with other components and the environment. The agent's visual sensory component (V) uses a hierarchical VAE architecture. V's principal function is to encode visual information into a concise and expressive representation. This learned representation lets the memory part of the world model draw on past events and infer actions based on them. Finally, the decision-making controller of the RL agent utilizes the representations constructed from its visual input and memory to determine the most appropriate actions to take when performing a task in complex environments. The general structure of the NVAE-based world model consists of the following: NVAE as the visual model, a memory model, and the controller model.

3.1 NVAE as the visual model

In the field of RL, a complex task is one that is challenging to learn due to factors such as the number of alternative actions, the size of the state space, and the complexity of the environment. Complex tasks in RL include robotic manipulation, in which the agent must learn to control a robotic arm to grasp and manipulate objects in a cluttered environment; navigation in complex environments, such as indoor environments or outdoor terrain, in which the agent must learn to navigate using visual and sensory inputs; and multi-agent coordination, in which multiple agents must learn to work together to achieve a common goal while competing for resources.

The world model approach successfully carried out a car racing-v0 task involving navigation within a continuous control action space (Li, 2019). It successfully explored the world using only the raw pixel data as input. However, the performance of the RL agent in the dream car racing-v0 assignment was impeded because it could not accurately generate a dream environment based on the input. A traditional VAE was the visual model

in the earlier world model approach. However, new studies have shown that variational autoencoders (VAEs) have shortcomings in representing complex data with long-range correlations, leading to either data loss or distortion during the encoding of input frames. This issue reduces the efficacy of agents trained with world models on large datasets in complex environments like robotics. This thesis proposes a solution to this problem by using nouveau VAE (NVAE), a deeply hierarchical VAE, as a visual sensory component of the world model.

For NVAE, the encoder and decoder use the same top-down model, resulting in a hierarchical architecture (as shown in the Figure 8). The architecture consists of multiple layers that contain residual cells distinct from the encoder and decoder. The multi-scale hierarchical paradigm of NVAE enables the agent to capture global long-range correlations at the top of the hierarchy and local dependencies at the bottom (Vahdat & Kautz, 2020). NVAE additionally uses deep residual networks with larger kernel sizes and 1x1 regular convolutions to improve expressivity and enlarge receptive fields, providing an effective solution to the problem of long-range correlations for data with higher resolution (as shown in Figure 9).

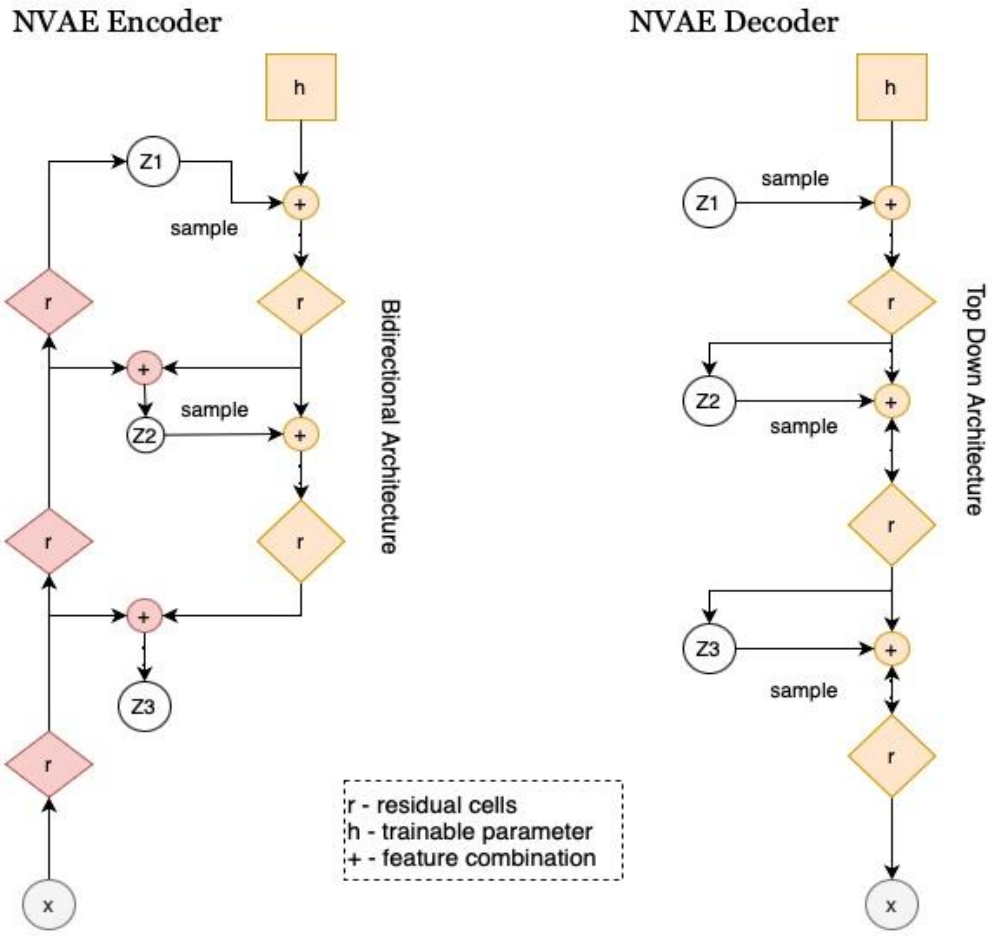

Figure 8: (Left) the encoder model of NVAE. (Right) the decoder model of NVAE.

Furthermore, by changing hyperparameters and regularizing scaling parameters during training, NVAE overcomes the difficulties of batch normalization (BN) during evaluation to achieve better stability, convergence, and performance. Through residual normal distributions and spectral regularization, it also addresses training stability with latent hierarchical groups and image sizes. When incorporated with the world model approach, NVAE's precise representation of input images enables efficient conversion of input image frame features to latent representations (z_1, z_2, ..., z_i).

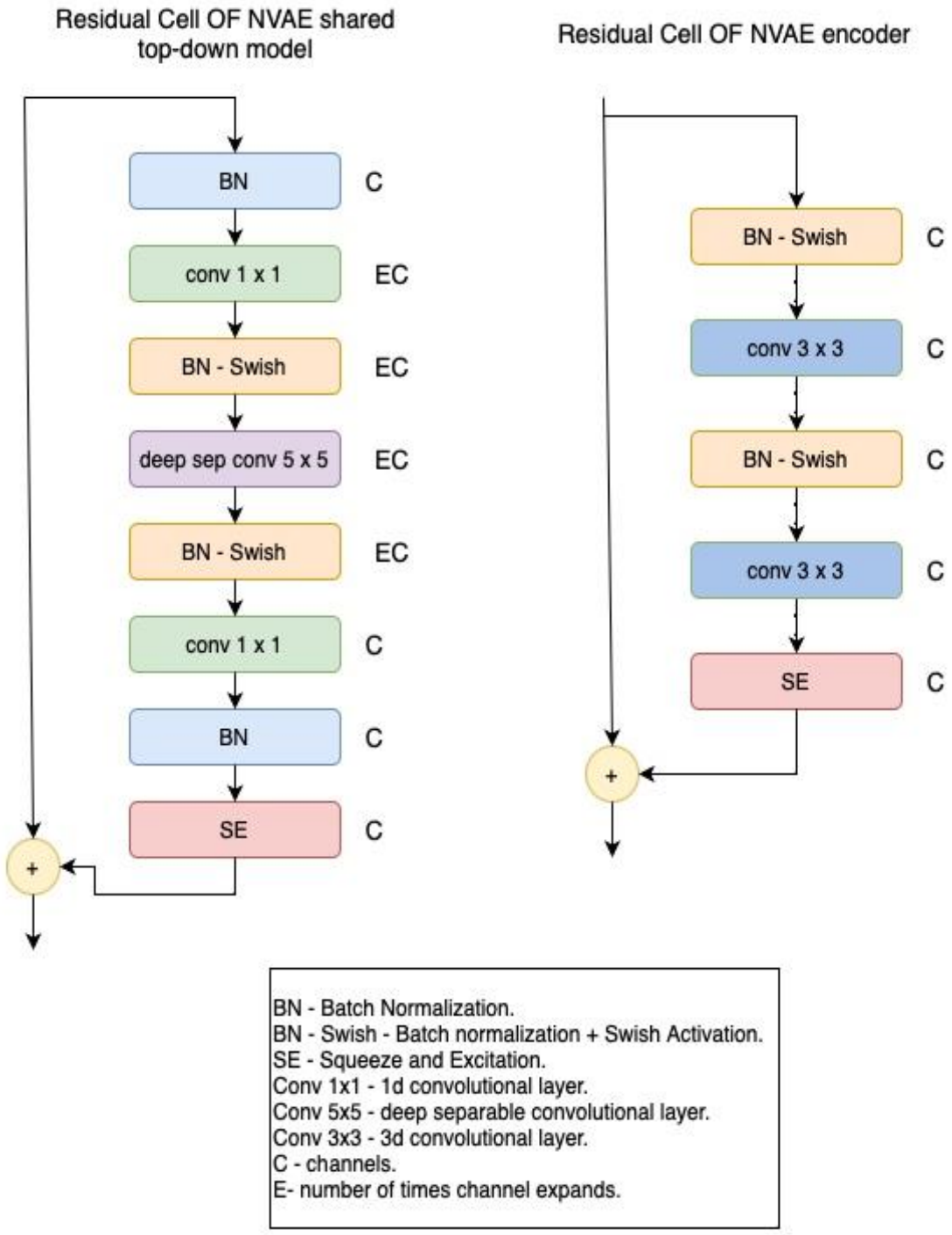

Figure 9: (Left) Residual cells in the encoder model. (Right) Residual cells in the decoder model. These cells help to improve the performance of the deep neural network.

Unlike conventional variational autoencoders, nouveau VAE employs a learnable gaussian prior distribution over the latent variables. The prior distribution is specifically characterized as a multivariate gaussian distribution with a learnable mean and covariance matrix. During training, this learnable prior is optimized with the remainder of the model. Utilizing a learnable prior in nouveau VAE enables the model to modify the prior distribution to the complex data characteristics, resulting in enhanced performance and more diverse samples. Moreover, the learnable prior can serve as a form of regularization, preventing overfitting and improving generalization.

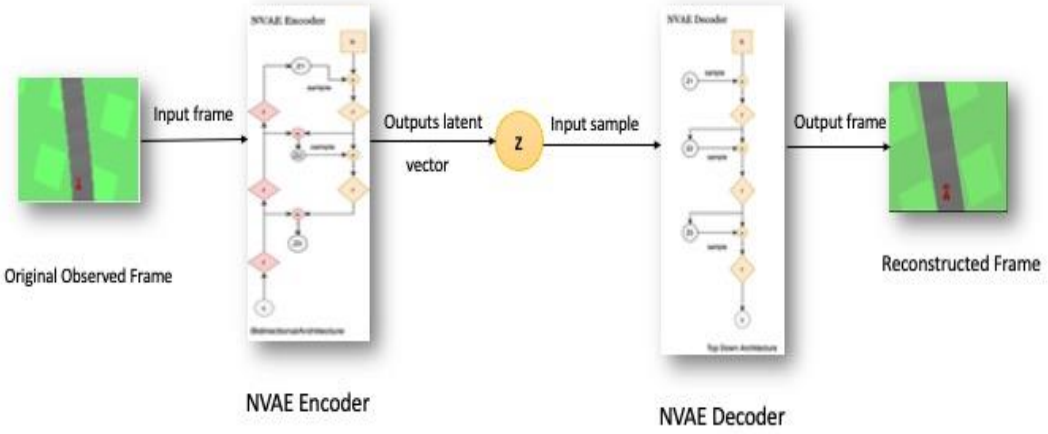

Figure 10: Flow diagram illustrating the process of image generation using NVAE as the encoder and decoder to generate realistic images.

The input during environmental interaction is provided as high-dimensional pixel images. First, we resize all the images to 128x128, which will serve as the "observation" for our visual models. The RGB components of each pixel are represented by three floating-point values between 0 and 1. To encode this 128 x 128 x 3 input into the low-dimensional vectors μ and σ of size N_z, the NVAE uses 4 convolutional layers. A gaussian prior N (z; g(x)) is learned and used to draw samples for the latent vector z. Four deconvolutional layers are used to decode and reconstruct the image based on the latent vector z. The stride size for both convolution and deconvolution layers are 2. All convolution and deconvolution layers utilize RELU activations except the output layer, which is in the range [0,1]. The model is trained for one epoch using a random policy across the data collected. The L2 distance between the input image and the reconstruction quantifies the reconstruction loss for optimization alongside the KL loss. Figure 10 illustrates the image generation process using NVAE.

3.2 Memory model

The M model employs a long short-term memory (LSTM) recurrent neural network with a mixture density network as the model's output layer (as shown in Figure 11). This network expects a mixture of gaussian distribution for the next z at the next time step. The correlation parameter between each element of z is not modeled by the memory model employed here. Instead, it has the MDN-RNN produce a factored gaussian distribution with a diagonal covariance matrix. As many complex settings are stochastic in nature, the MDN-RNN model is trained to produce a probability density function (p(z)) rather than a deterministic prediction of 'z.'

$P(z_{t+1} \mid a_t, z_t, h_t)$ where a_t is the action taken at time' t', h_t is the hidden state of RNN at time t, and z_t is the latent input vector.

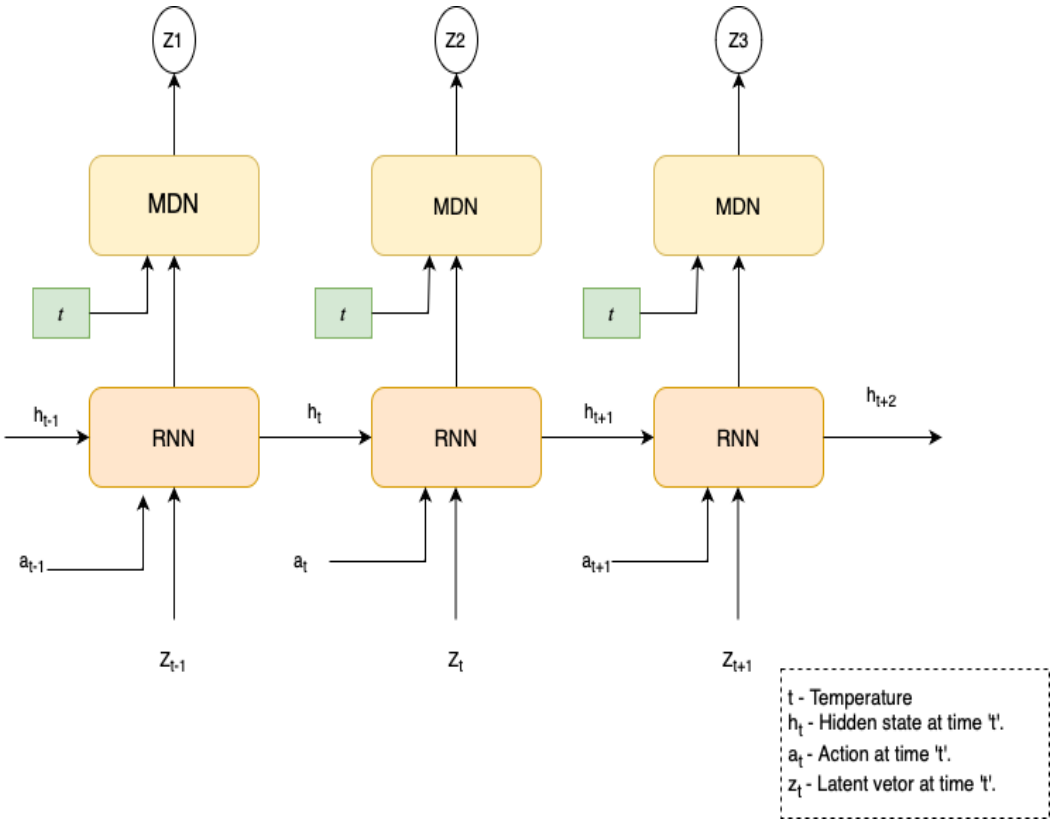

Figure 11: The architecture of mixture density network - recurrent neural network (MDN-RNN) model.

When compared to earlier research in handwriting and sketch generation, MDN-RNN models the probability density function (p(z)) of the following latent vector (z). The probability density function is sampled at each time step to generate the hallucinated environment. MDN-RNN can also forecast the agent's state probability in the hallucinated environment. If the likelihood is more than 50%, it is accepted as representative of the actual environment. In our experiments, the probability that the environment is accurately represented is greater than 65 percent. The memory component is trained for 20 epochs using data collected from the random policy.

M tries to predict the next value of z. This is fed to the MDN module whose goal is to introduce randomness that is, it changes the output of the LSTM which is a deterministic z value into a range of possibilities for z.

3.3 Controller model

The controller generates actions or policies based on the latent vectors provided by the encoder and the internal state of the memory component. According to the specific task or environment being modelled, the controller generates actions that are utilized to interact with the environment, such as moving, rotating, or executing other activities. The controller often uses learned representations in latent vectors and memory-component context to generate actions.

The controller can guide the agent in its exploration strategy using data from the latent vectors and the memory component. This allows the agent to effectively explore its environment and discover new states or actions while also maximizing rewards by utilizing previously learned information.

The proposed method uses a controller model that is a simple linear neural network model that directly maps each time step's action to the latent vector z_t and hidden state h_t, where:

$$a_t = W_c[z_t, h_t] + b_c$$

Equation 8

The linear model is described by the above equation, where W_c and b_c are the weighted matrix and bias vector (respectively), mapping the combined input vector $[z_t, h_t]$ to an output action vector (a_t).

The covariance matrix adaptation evolution strategy (CMA-ES) algorithm is used to determine the optimal values for the controller's parameters (W_c and b_c). In the

experiments, the action space is clipped and bound to the necessary ranges for all situations using tanh nonlinearities.

3.4 Architecture of NVAE based world model:

The following flow diagram (Figure 12) illustrates how V (NVAE), M (MDN-RNN), and C interact with the environment:

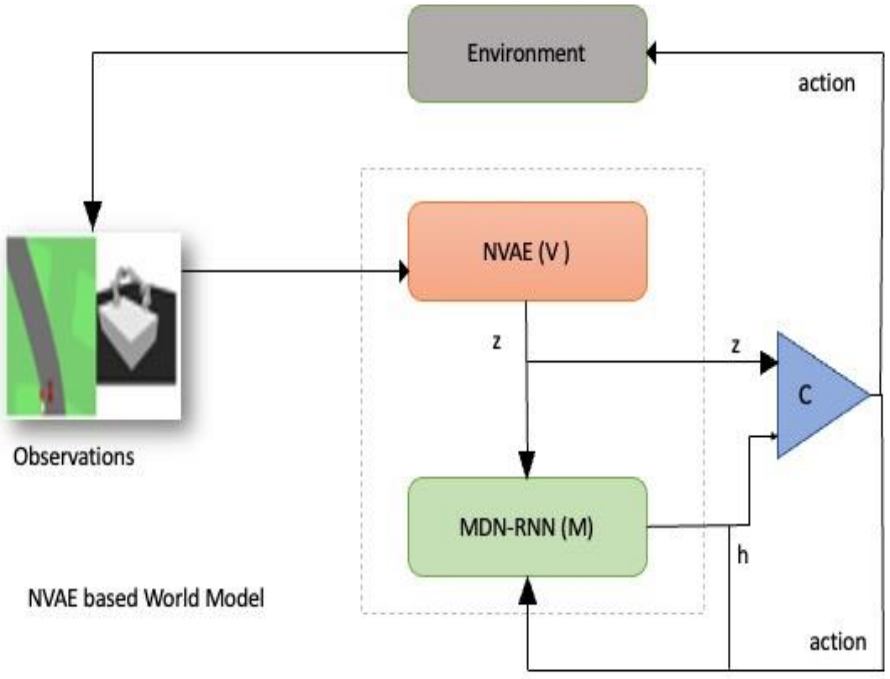

Figure 12: The flow chart of NVAE based world model.

The proposed NVAE-based world model has a different visual model design than the current one. In the proposed model, a deep hierarchical variational autoencoder (NVAE) is employed as the visual model to compress environmental features into compact latent

representations. At each time step, NVAE processes the raw environmental observations to generate latent representations.

Chapter 4: Experiments

This chapter will describe the training procedure for the proposed model-based agent to perform tasks in three distinct environments. Section 4.1 will explain the training process and performance of NVAE-based world model agent in car racing-v2 environment. Section 4.2 discusses the generation of the dream environment based on the car racing-v2 environment and the agent's performance in the dream environment and comparison of performance to the prior world model. Section 4.3 describes the training and performance of the proposed model-based agent in a robotic environment (the panda gym).

4.1 Car Racing-v2 Task

4.1.1 Environment

Car racing-v2 is an environment within the Open AI Gym, a toolkit for developing and comparing reinforcement learning algorithms. (Brockman et al., 2016) In this scenario, an agent drives a car that must race around a track to finish a predetermined number of laps as quickly as possible while avoiding obstacles and staying on the course. The agent receives the environment's 96x96 RGB image, velocity, and steering inputs. The agent aims to learn a strategy that maximizes its reward based on the speed and distance traveled around the track. Significant research studies have utilized car racing-v2 as a benchmarking environment for reinforcement learning algorithms.

The environment's intricate dynamics and non-linear rewards are significant obstacles to reinforcement learning algorithms. Therefore, the proposed agent is trained on the car racing-v2 task as shown in Figure 13.

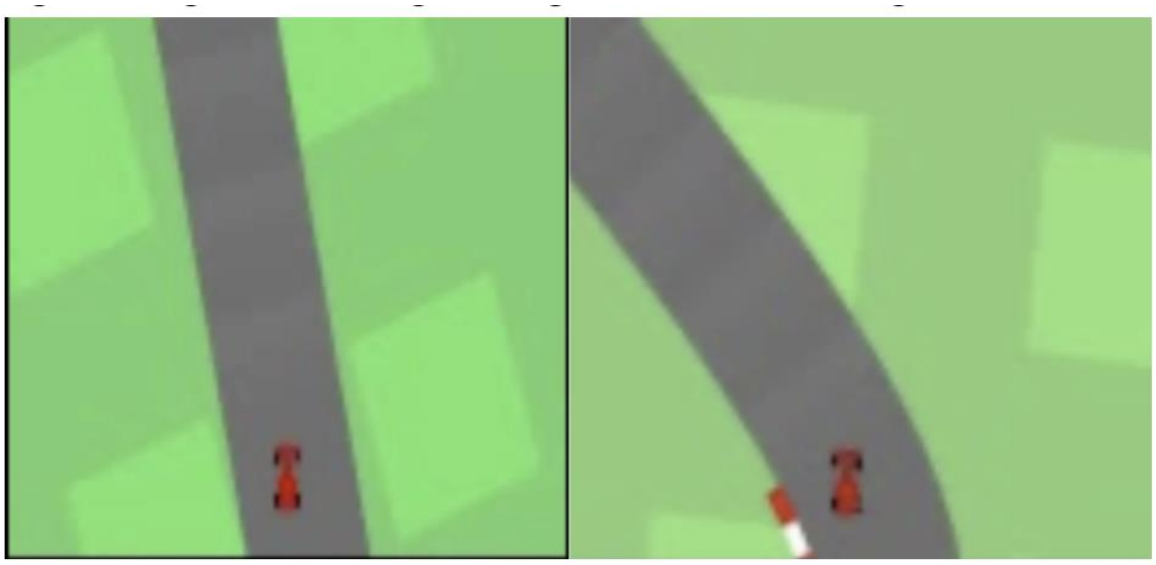

Figure 13: The figure depicts an agent exploring the car racing-v2 environment.

In this image, the agent is navigating through the environment by taking actions and making decisions based on the information it receives from the environment. This process helps the agent learn and improve performance in the given task.

4.1.2 Dataset

The agent's random interactions with the car racing-v2 environment generates the dataset. To comprehend the environment's dynamics, the agent keeps track of its observations during interactions. In this experiment, we collect 30 million images from the agent's interactions with the environment throughout 10,000 rollouts and 3000 timesteps. Each pair in the dataset consists of an observation (a 96x96 RGB image) and an action (a vector of two continuous values reflecting the car's velocity and steering angle). The dataset is divided into three subsets: a training set, a validation set, and a test set. The NVAE-based world model is trained with the training set, while hyperparameters and

progress are tracked with the validation set. The purpose of the test set is to evaluate the learned model's performance on unobserved data.

4.1.3 Training Process

The V (NVAE) model is first trained on the dataset to produce a latent vector z, a compact representation of each input frame. The latent vector z encodes each frame at time t into zt, allowing for a low-dimensional representation of the frame, which is then used to train the memory (M). M then learns to model a mixture of gaussians, using the pre-trained data and the observed random actions (at) to reflect the environment's dynamics. Training V and M together helps keep the world model's computational complexity under control. The size of the low-dimensional vectors for the car racing-v2 task is 64. There are 256 hidden units in the LSTM employed in MDN-RNN. V and M cannot access the environment's reward signal (reward scores). The reward signal is only available to the controller (C). C interacts with the environment by performing actions (at) after receiving latent vectors (z) from V and hidden states (ht) from M as inputs.

The algorithm used to train the agent in the car racing-v2 environment is as follows -

Algorithm

1. Sample 10,000 rollouts using a random policy.
2. Train V (NVAE) to encode frames into $z \in R^{32}$
3. Train M (MDN-RNN) to model $P(z_{t+1}|a_t, z_t, h_t)$
4. Use CMA-ES on controller (C) to maximize the expected reward of a rollout.

4.1.4 Results

The results of the car racing-v2 task are based on how many laps the agent has finished without going off the track for 1000 iterations. The highest score the agent can get is

1000. As shown in the table, the agent attained an average score of 800 ± 18 after 100 random trials, outperforming the traditional world model approach. Compared to the previous procedure, the rendered results have higher image quality.

Model	Average Scores
World Model	720 ± 13
NVAE based World Model	**800 ± 18**

Table 1: Average scores obtained by previous and proposed world model for car racing-v2 task.

Table 1 compares the average scores obtained by the previous world model and the NVAE based world model for the car racing-v2 task. The table displays the performance of each model and shows that the NVAE-based model outperforms the previous model in terms of average scores.

4.2 Dream Car Racing-v2 Task

4.2.1 Environment

Traditional model-based methods trained the agent in the actual environment where the agent learns a model of the environment. The world model enables the agent to explore a hallucinated environment based on the actual environment fully. Here, the agent's controller is trained in a simulation of its real-world environment created by the

agent's world model. The temperature parameter of M is used to regulate the randomness of the dream environment, hence minimizing the agent's exploitation of the environment's imperfections. C is then trained in the generated environment, which is noisier and more uncertain.

In this section, the dream car racing-v2 environment is built based on the internal world model M's understanding of the latent representations. The proposed world model-based agent is trained within this environment. Here, consider the latent representations of the car racing-v2 task (see section 4.1), where the proposed model learned a policy to complete the task successfully. The agent does not receive direct sensory input from the dream environment but instead relies on the data provided by the proposed world model to guide its observations and subsequent actions. This emphasizes the significance of capturing precise latent representations of the environment. To construct a dream car racing-v2 environment that replicates the actual environment's dynamics, wrap a gym.env interface around the memory model (M) and consider this to be the training environment for the agent (Gymnasium Documentation, n.d.). Figure 14 shows the agent navigating in dream car racing-v2 environment.

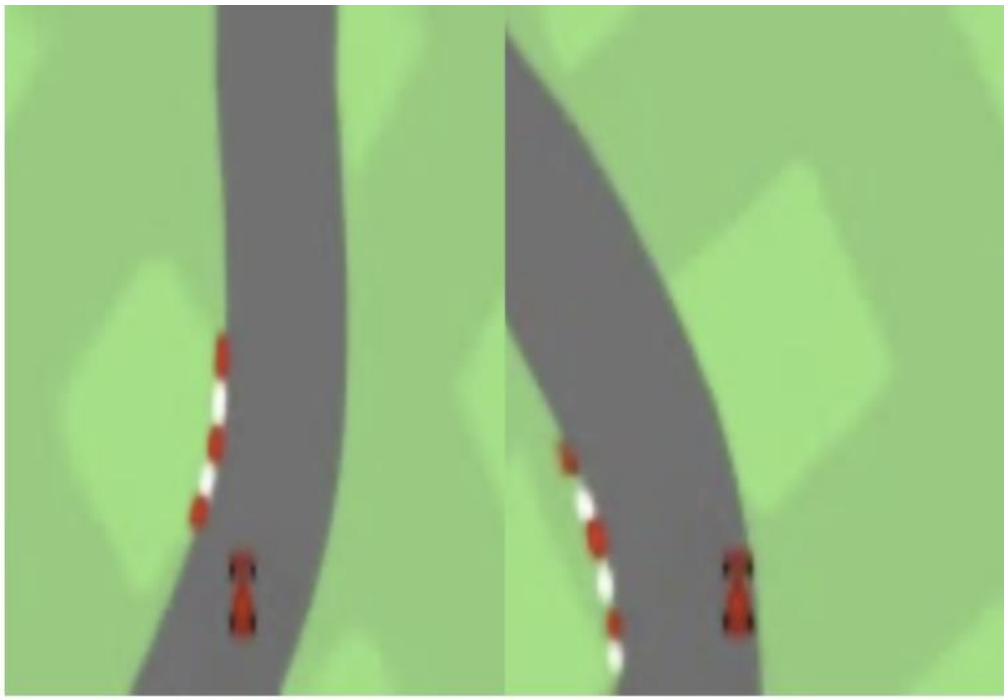

Figure 14: (left) the world-model agent navigating the car racing-v2 environment. (right) the NVAE-based world model agent navigating the car racing-v2 environment.

In this image, the agent navigates a simulated environment generated by the world model. This process allows the agent to practice and improve its performance without interacting with the actual environment, thus reducing the risk of damage or accidents.

4.2.2 Dataset

Creating a dataset would be similar in both the real world and the agent's dream world, as both involve training the agent to complete car racing-v2 tasks. The dataset is produced by the random interactions of the agent with the car racing-v2 environment. The agent maintains a record of its observations during interactions to better understand the environment's dynamics. This experiment encompasses 10,000 rollouts and 3000 timesteps, during which 30 million images of the agent's interactions with the environment are collected. The dataset contains 96 x 96 RGB images as observations and

a vector of two continuous values representing the car's speed and direction as actions. The dataset is segmented into a training set, a validation set, and a test set. The training set is used to train the NVAE-based world model, while the validation set is used for tracking hyperparameters and performance. The test set is used to assess the trained model's accuracy when applied to data that was not used in the training process.

4.2.3 Training Process

The agent's main objective in the dream car racing-v2 environment is to maximize its total rewards by navigating around the generated racetrack as quickly as possible. The agent is trained in a way consistent with Section 4.1.3 since the primary task to be completed in the dream environment is identical to that of the real world. To construct a compact representation of each input frame, the V (NVAE) model is trained on the dataset to generate a latent vector z. Each frame at time t is represented by a low-dimensional representation of the frame (zt) encoded by the latent vector z, which is used to train the memory (M). M then uses the pre-trained data and the observed random actions (at) to create a mixture of Gaussians that accurately represents the dynamics of the environment. Co-training V and M reduce the computational complexity of the world model. For the dream car racing-v2 task, the low-dimensional vector size is 64. The LSTM used by MDN-RNN has 256 hidden units. The environment's reward signal (scores) is unavailable to V and M. Only the controller (C) has access to the reward signal. After obtaining latent vectors (z) from V and hidden states (ht) from M, C interacts with the environment by conducting actions (at). The controller (C) is optimized with the help of the CMA-ES evolutionary approach. The following algorithm is used to train the agent in dream environment.

Algorithm

1. Sample 10,000 rollouts using a random policy.
2. Train V (NVAE) to encode frames into $z \in R^{32}$

3. Train M (MDN-RNN) to model $P(z_{t+1}|a_t, z_t, h_t)$
4. Use CMA-ES on controller (C) to maximize the expected reward of a rollout.

4.2.4 Results

The trained NVAE-world model-based agent must navigate in the dream car racing-v2 environment to maximize rewards while navigating obstacles on the racetrack. The results of the dream car racing-v2 task is also based on how many laps the agent has finished without going off the track for 1000 iterations. The highest score the agent can get is 1000. The results are given in Table 2 where the agent achieved an impressive average score of 760±18 on the dream car racing v2 task, outperforming the previous world-model agents, which achieved an average score of 696 ± 16 in the dream environment.

To comprehend the impact of the deep hierarchical VAE (NVAE) employed in the visual component of the world model, the FID score was used to assess the accuracy of the reconstructed image in both car racing-v2 and its generated dream environment. The Fréchet Inception Distance (FID) is a measure used to assess the robustness and diversity of image synthesis models like generative adversarial networks (GANs) and variational autoencoders (VAEs). The lower the FID score, the closer the output image is to the input image. For a traditional VAE, the FID score is 271.58, and for an NVAE, it is 174.67.

There is a 45 percent improvement in image quality when NVAE is used. Therefore, utilizing NVAE for the visual representation of the input environment improves the agent's performance.

Model	Average Scores
World Model	694 ± 16
NVAE based World Model	**760 ± 18**

Table 2: Average scores obtained by previous and proposed world model for dream car racing-v2 task.

4.3 Panda-Gym Reach Task

In this section, the agent is taught to perform a robotic manipulation task in the panda-gym environment called the reach task as the proposed NVAE based world model agent outperformed the traditional world model in continuous control tasks.

4.3.1 Environment

Robotic manipulation is one of the most challenging tasks for reinforcement learning algorithms. Regarding training RL algorithms, the panda-gym environment is one of the best-simulated robotics environments available. It is designed to simulate the control of a real-world robotic arm, which can perform various tasks, including pick-and-place, reach, and grasping. The realistic physics simulation in the panda-gym environment is a key feature that facilitates the training of RL algorithms to operate a robotic arm in a physics-based environment. Various environmental sensors and actuators give a reward signal that can be used to train an RL agent to perform tasks in the environment (Gallouédec et al., 2021).

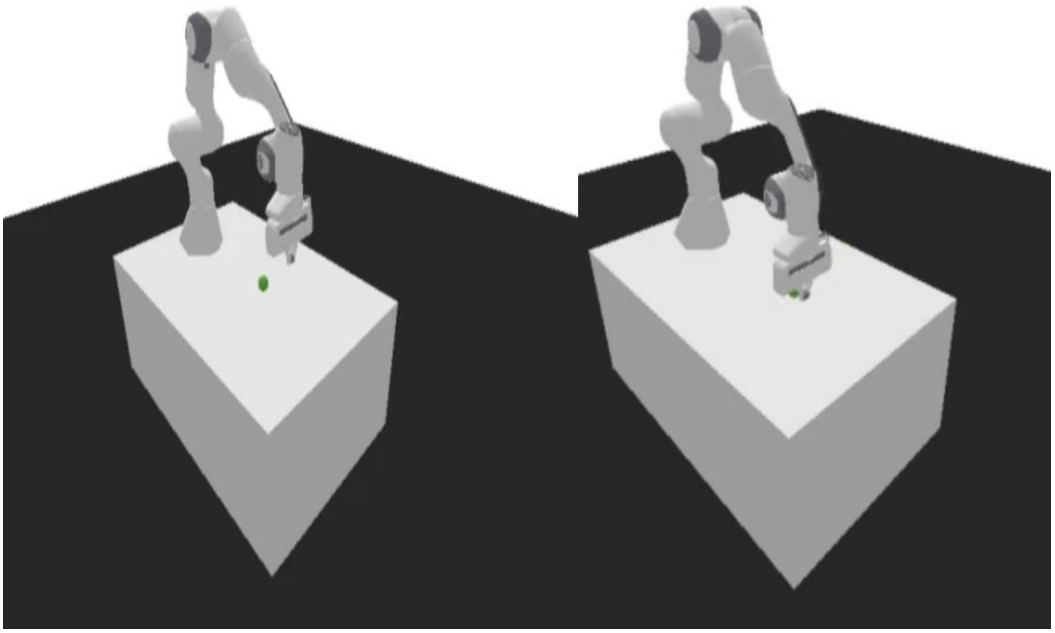

Figure 15: The above figure depicts the panda-gym environment, a simulation where a robotic arm acts as the agent and must complete the task of reaching an object and moving it to a target location.

Figure 15 shows the agent in action, with the robotic arm moving toward the object and grasping it before moving toward the target location to complete the task. This environment is designed to test the agent's ability to perform complex manipulation tasks that require precise motion control and decision-making abilities.

The Franka Emika robotic arm was the basis for the panda gym environment. Using the multi-objective RL architecture, the panda-gym environment will produce a new goal for each episode, with the goal type changing depending on the activity being performed. The agent's action space is a single robot arm performing either end-effector displacement control or joint control across six discrete tasks. The agent is taught in panda-gym's joint control mode for the Reach task, which trains the robotic arm on how

to move an object to a specific location. To complete the Reach task, the model must use the seven-jointed robotic arm to pick up an object and move it to a predetermined location. Given that this task requires continuous control, the robotic arm must be trained to move all its joints simultaneously (Lillicrap et al., 2019). Figure 15 is the visual rendition of agent performing the reach task in panda-gym environment.

4.3.2 NVAE's Architecture

The agent must complete the reach task in a panda-gym environment based on a physical simulation of robot arm movements. As a result, the environment is more complex than the car racing-v2 environment. Therefore, the agent must encode various resolutions of the input image frames. Since the proposed method uses NVAE as the visual component to encode the input image frames, the convolutional layers must be modified to process the image to a much smaller latent dimension. The adaptable design of NVAE allows for processing varying picture data without impacting the agent's performance. In this work, both the encoder and decoder models of NVAE are constructed using solely convolutional 1x1 channels. The remainder of the architecture remains the same as shown in Figure 16.

4.3.3 Dataset

As the proposed world model is used to train the agent, a random policy is used to interact with the panda-gym environment to create the dataset. The dataset is produced using 10,000 rollouts spread over 3000 timesteps. The dataset includes 84x84 RGB images and joint angles as observations, as well as an action vector and reward vector. For this experiment, the latent dimension is 8.

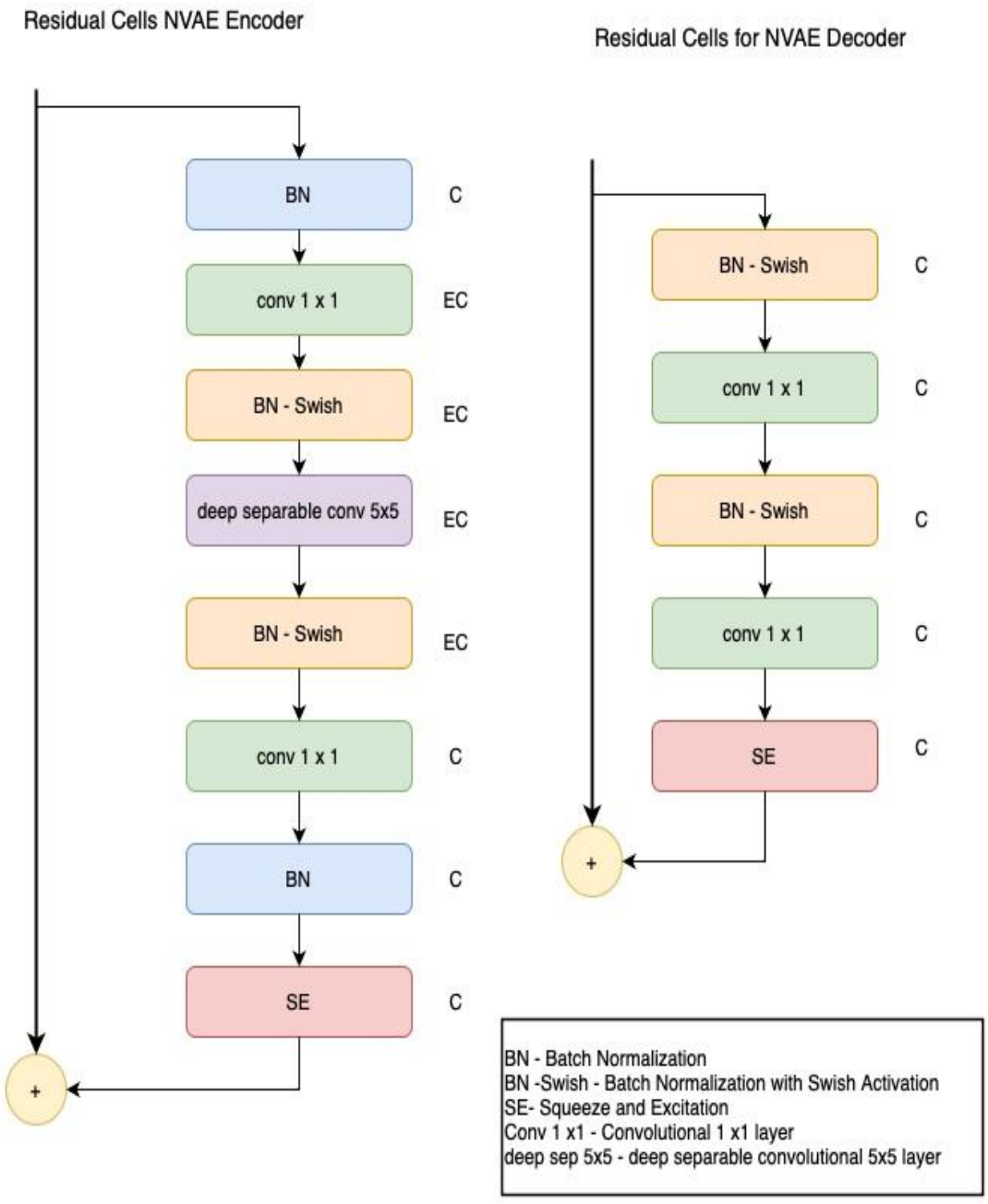

Figure 16: (Left) Modified residual cells in encoder model. (Right) Modified residual cells in decoder model.

4.3.4 Training Process

With modifications to the NVAE architecture, the V component is trained to encode each frame into a latent vector (z) of size 8. The trained V component was then used to train the M component to map each frame at time step t to a new vector, zt. The M component is designed to simulate a Gaussian distribution by combining this data with observations of random behavior (at). The characteristics of V and M components are then passed onto controller C for decision-making. MLP (multi-layer perceptron neural network) and CMA-ES (covariance matrix adaptation - evolution strategy) are used for optimization because the C component is a simple linear model with just 432 parameters.

Algorithm

1. Sample 10,00 rollouts using a random policy in the panda-gym environment.
2. Train V (NVAE) to encode frames into $z \in R^8$
3. Train M(MDN-RNN) to model $P(z_{t+1}|a_t, z_t, h_t)$
4. Apply MLP and CMA-ES optimization methods on controller (C) to maximize the expected reward of a rollout.

4.3.5 Results

The NVAE-based world model agent completed the reach task with a 95% success rate. While prior model-free RL methods like SAC, DDPG, and TD3 were all applied to the reach task, only DDPG was successful. Compared to model-free methods like DDPG, the proposed model, based on a model-based RL approach employing NVAE, has produced competitive results. The NVAE-based world model method is the first model-based RL approach in this environment and has shown promising outcomes (as shown in Table 3).

Model	Success Rate
DDPG (model-free)	100%
SAC (model-free)	85%
TD3 (model-free)	65%
NVAE based World Model (model-based)	**95%**

Table 3: Success Rate of Model free RL methods and the proposed MBRL for panda-gym reach task.

The table presents the success rates of model-free reinforcement learning (RL) methods and the proposed model-based RL method on the panda-gym reach task. The success rate measures the percentage of successful episodes where the robotic arm successfully reaches the target object and moves it to the target location. The results show that the proposed model-based RL method outperforms various model-free RL methods in terms of success rate, demonstrating the effectiveness of the proposed approach on this task.

Chapter 5: Conclusion

This thesis proposes a deep hierarchical variational autoencoder (NVAE) to substitute the traditional VAE in the existing world model, resulting in a new and improved version of the world model. Recent research has shown that the V model plays a vital role in the agent's performance across all contexts because of its ability to compress the spatial/temporal representations of the input frame into a small latent vector. The controller uses the latent vector and the hidden state to predict the agent's actions to perform a task. Using NVAE as a visual model significantly improved the agent's performance.

On the CarRacing-v2 task, the proposed model scored an average of 800 ± 18, while on the CarRacing-v2 task in the dream environment, it scored 760 ± 18. Compared to the previous model, the proposed method performed remarkably well on the dream task. The FID score is used to evaluate the effect of V(NVAE) on agent performance. When NVAE is used, the quality of regenerated images improves by 45 percent. This validates the proposal that the agent’s performance improves when NVAE is used as the visual component of the world model. The proposed model-based agent is trained to complete the Reach task in the panda gym. Since the input frame is a vector, the proposed model required only minor modifications to the NVAE's convolutional layers to train the agent to perform the Reach task successfully. The model achieved a 95% success rate. The proposed methodology is believed to be the first model-based approach to be applied to the panda gym environment.

References

1. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym (arXiv:1606.01540). arXiv. https://doi.org/10.48550/arXiv.1606.01540
2. Gallouédec, Q., Cazin, N., Dellandréa, E., & Chen, L. (2021). panda-gym: Open-source goal-conditioned environments for robotic learning (arXiv:2106.13687). arXiv. https://doi.org/10.48550/arXiv.2106.13687
3. Gymnasium Documentation. (n.d.). Retrieved May 3, 2023, from https://gymnasium.farama.org/index.html
4. Ha, D., & Schmidhuber, J. (2018). World Models. https://doi.org/10.5281/zenodo.1207631
5. Hirshon, J. M., Risko, N., Calvello, E. J. B., Stewart de Ramirez, S., Narayan, M., Theodosis, C., O'Neill, J., & Acute Care Research Collaborative at the University of Maryland Global Health Initiative. (2013). Health systems and services: The role of acute care. Bulletin of the World Health Organization, 91(5), 386–388. https://doi.org/10.2471/BLT.12.112664
6. Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research. https://arxiv.org/abs/cs/9605103v1
7. Keller, G. B., Bonhoeffer, T., & Hübener, M. (2012). Sensorimotor mismatch signals in primary visual cortex of the behaving mouse. Neuron, 74(5), 809–815. https://doi.org/10.1016/j.neuron.2012.03.040
8. Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improved Variational Inference with Inverse Autoregressive Flow. Advances in Neural Information Processing Systems, 29. https://papers.nips.cc/paper_files/paper/2016/hash/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Abstract.html
9. Kingma, D. P., & Welling, M. (2013, December 20). Auto-Encoding Variational Bayes. ArXiv.Org. https://arxiv.org/abs/1312.6114v11

10. Klushyn, A., Chen, N., Kurle, R., Cseke, B., & van der Smagt, P. (2019). Learning Hierarchical Priors in VAEs. Advances in Neural Information Processing Systems, 32. https://papers.nips.cc/paper_files/paper/2019/hash/7d12b66d3df6af8d429c1a357d8b9e1a-Abstract.html
11. Li, C. (2019, November 2). Challenging On Car Racing Problem from OpenAI gym. ArXiv.Org. https://arxiv.org/abs/1911.04868v1
12. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2019). Continuous control with deep reinforcement learning (arXiv:1509.02971). arXiv. https://doi.org/10.48550/arXiv.1509.02971
13. Maaløe, L., Fraccaro, M., Liévin, V., & Winther, O. (2019). BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling (arXiv:1902.02102). arXiv. https://doi.org/10.48550/arXiv.1902.02102
14. Maus, G. W., Fischer, J., & Whitney, D. (2013). Motion-dependent representation of space in area MT+. Neuron, 78(3), 554–562. https://doi.org/10.1016/j.neuron.2013.03.010
15. Mobbs, D., Hagan, C. C., Dalgleish, T., Silston, B., & Prévost, C. (2015). The ecology of human fear: Survival optimization and the nervous system. Frontiers in Neuroscience, 9, 55. https://doi.org/10.3389/fnins.2015.00055
16. Nortmann, N., Rekauzke, S., Onat, S., König, P., & Jancke, D. (2015). Primary Visual Cortex Represents the Difference Between Past and Present. Cerebral Cortex (New York, NY), 25(6), 1427–1440. https://doi.org/10.1093/cercor/bht318
17. Quiroga, R. Q., Reddy, L., Kreiman, G., Koch, C., & Fried, I. (2005). Invariant visual representation by single neurons in the human brain. Nature, 435(7045), Article 7045. https://doi.org/10.1038/nature03687
18. Razavi, A., van den Oord, A., & Vinyals, O. (2019). Generating Diverse High-Fidelity Images with VQ-VAE-2. Advances in Neural Information Processing Systems, 32. https://papers.nips.cc/paper_files/paper/2019/hash/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Abstract.html

19. Vahdat, A., & Kautz, J. (2020, July 8). NVAE: A Deep Hierarchical Variational Autoencoder. ArXiv.Org. https://arxiv.org/abs/2007.03898v3
20. Werbos, P. J. (1989). Neural networks for control and system identification. Proceedings of the 28th IEEE Conference on Decision and Control, 260–265. https://doi.org/10.1109/CDC.1989.70114

Appendix: Hyperparameters

Considering the large number of datasets and the associated high computational demands, we do not need to perform a complete optimization of the hyperparameters. In general, larger networks, a greater number of hierarchical groups, and more residual cells per group result in improved performance. However, they require more training time and have smaller training batch sizes. The hyperparameters are adjusted to ensure that the model can be trained in under a week. Our experimental hyperparameters are summarized in Table 4.

Channel Sizes: Here, only the initial number of channels is provided for the bottom-up encoder. When the features are down sampled spatially, the encoder's channels are doubled. In the top-down model, the number of channels is set in the reverse order.

Datasets: The proposed model was examined on the data generated by random interactions on car racing-v2, dream car racing-v2, and panda-gym environments. The data is split into tests and training for all three tasks.

Hyperparameters	Car racing-v2	Dream car racing-v2	Panda Gym Reach
epochs	1000	1000	1000
Spatial dimensions of z ($\mathbf{D^2}$)	$8^2, 16^2, 32^2 64^2$	$8^2, 16^2, 32^2 64^2$	$8^2, 16^2$
Channel size	20	20	20
$\boldsymbol{\lambda}$	0.1	0.1	0.1
GPUs	3	3	2
GPU type	32GB	32GB	16GB
Batch size per GPU	32	32	16
normalizing flows	2	2	2
stride	2	2	2

Table 4: The table summarizes hyperparameters used in training NVAE based world model in three environments. D2 indicates a latent variable with the spatial dimensions of D × D.