

# **Variational Autoencoder-based Reinforcement Learning for 7-DoF Arm and 2-DoF Gripper Model**

By Sangbeen Lee

DENM100 Extended Research Project September 2023

Supervisor: Dr Lorenzo Jamone

SCHOOL OF ENGINEERING AND MATERIALS SCIENCE - ENGINEERING

SCIENCE AND ENGINEERING

EXTENDED RESEARCH PROJECT

DENM100

SEPTEMBER 2023

DECLARATION


This report entitled

VARIATIONAL AUTOENCODER-BASED REINFORCEMENT LEARNING FOR 7-DOF ARM 2-DOF  
GRIPPER MODEL

Was composed by me and is based on my own work. Where the work of the others has been used, it is fully  
acknowledged in the text and in captions to table illustrations.

This report has not been submitted for any other qualification.

Name Sangbeen Lee

Signed 

Date .....08/09/23.....

## List of Contents

List of Symbols.....	4
Abstract.....	5
I. INTRODUCTION.....	5
II. LITERATURE REVIEW .....	6
A. Reinforcement Learning.....	6
B. Autoencoder .....	8
C. Variational Autoencoder .....	9
III. METHODOLOGY .....	10
A. Algorithm of VAE .....	10
B. Layer of VAE.....	11
1) Encoder.....	11
2) Decoder.....	11
3) VAE Model .....	12
IV. RESULTS.....	12
A. Parameters.....	12
B. Loss .....	12
C. Representation .....	13
1) Abstract Start Representation.....	13
2) 2D Projection: .....	13
3) Abstract Goal (Precondition of Action):.....	14
V. DISCUSSION .....	14
A. Trainable and Non-trainable Parameters .....	14
B. Impact of the Number of Objects on Training Loss.....	15
C. Manifold Learning and VAE Components .....	16
D. Manifold Assumption and Dimensionality Reduction.....	17
VI. CONCLUSION .....	18
ACKNOWLEDGMENTS .....	19
REFERENCES .....	19
APPENDIX .....	22
Appendix A.....	22
Appendix B.....	25
Student's and other's contributions to project.....	33

### List of Symbols

Symbols	Remarks
$p_{\theta}(x)$	Likelihood or probability distribution of data $x$ given the generative model with parameters $\theta$ .
$p_{\theta}(x z)$	Likelihood or probability distribution of data $x$ given latent variable $z$ and the generative model with parameters $\theta$ .
$p(z)$	Prior probability distribution of the latent variable $z$ .
$q_{\phi}(z x)$	Approximate posterior probability distribution of latent variable $z$ given data $x$ , parameterized by $\phi$ .
$\log$	Natural logarithm.
$\int$	Integral symbol, representing the integral over a continuous variable.
$dz$	Differential element in the integral with respect to the latent variable $z$ .
$\int q_{\phi}(z x) \log p_{\theta}(x z) dz$	Reconstruction term, measuring how well the model reconstructs the input data $x$ from latent variable $z$ .
$KL(q_{\phi}(z x)  p(z))$	Kullback-Leibler ( $KL$ ) divergence between the approximate posterior $q_{\phi}(z x)$ and the prior $p(z)$ .
$KL(q_{\phi}(z x)  p_{\theta}(z x))$	$KL$ divergence between the approximate posterior and the posterior under the generative model.

**Abstract - This report explores the transformative potential of Variational Autoencoders (VAE) in robotic vision especially, reinforcement learning, focusing on efficient environment representation. VAE reduces dimensionality while preserving data structure, enabling robust navigation of complex state spaces. By encoding data into a lower-dimensional latent space, VAE enhances robotic decision-making and adaptability. The latent space's probabilistic nature ensures continuity, smoothness, and adherence to a known distribution, facilitating meaningful transitions between states and data generation. A correlation between environmental complexity and VAE training dynamics is uncovered, revealing that simpler scenarios yield rapid convergence to low loss values, while more intricate environments involve slower convergence and higher final loss values, necessitating adaptive learning strategies. The interplay between the encoder and decoder components proves pivotal. The encoder maps data to a lower-dimensional latent space, and the decoder reconstructs data while preserving relationships. This synergy empowers manifold learning, bolstering reinforcement learning, control, and decision-making in robotics. This work underscores VAE's transformative potential in addressing the challenges posed by complex robotic environments, opening avenues for future research and innovative solutions in autonomous robotics.**

## I. INTRODUCTION

Reinforcement learning has made remarkable strides in recent years, demonstrating its prowess in training agents to make decisions in complex and dynamic environments [1]. Its applications span diverse domains, from conquering games like AlphaGo to powering robotic systems and autonomous agents [2][3][4]. Nevertheless, despite these notable achievements, reinforcement learning encounters significant hurdles, especially when it encounters high-dimensional state spaces. This issue is magnified in visual reinforcement learning, where agents should

decipher policies and behaviours directly from image-based observations [5].

The central challenge in visual reinforcement learning arises from the tremendous dimensionality of raw image inputs. These images are often filled with intricate details and complex elements, such as noisy backgrounds, which confound agents trying to directly gain meaningful insights from them [6]. As a result, agents are faced with the arduous task of learning and generalizing more effectively across various environments, leading to extended training periods, vulnerability to issues like overfitting, and difficulties in achieving optimal performance.

To tackle these formidable challenges, this research investigates the transformative potential of Variational Autoencoders (VAE) in the realm of visual reinforcement learning [7]. VAE constitutes a subset of autoencoders celebrated for its prowess in dimensionality reduction and representation learning. Unlike traditional autoencoders, however, VAE introduces several pivotal advantages [7][8][9][10]:

1) Generative Prowess: VAE is inherently endowed with generative capabilities. It not only encodes data into lower-dimensional latent spaces but also decodes these representations back into data points that faithfully resemble the training data. This generative facet proves invaluable in scenarios such as data augmentation and data completion.

2) Structured Latent Space: VAE forges a structured latent space where each point corresponds to a meaningful representation of the data. This structured

representation proves indispensable for tasks necessitating abstraction, reasoning, and analysis of input conditions, extending its utility beyond mere image generation.

3) Uncertainty Estimation: VAE provides a natural means to estimate uncertainty in its reconstructions. This proves particularly advantageous when handling noisy or uncertain data sources. Gaining insights into the level of uncertainty associated with each abstraction proves pivotal for downstream decision-making and tasks.

4) Flexibility and Representation Learning: VAE exhibits remarkable adaptability to various data types and complex data distributions. Its aptitude for capturing intricate data patterns and distributions renders it a versatile choice for representation learning—a cornerstone of reinforcement learning.

5) Regularization: VAE comes fortified with a built-in regularization term that encourages a Gaussian-like distribution in the latent space. This regularization augments the robustness of learned representations and mitigates overfitting, mitigating common challenges in reinforcement learning.

This report revolves around variational autoencoder-based reinforcement learning, delving into the application of deep learning techniques to enhance reinforcement learning. The research probes how reinforcement learning agents can harness VAE to acquire low-dimensional, semantically meaningful representations of their environments, while simultaneously proposing efficient training

algorithms to optimize both encoder and decoder components.

In conclusion, the transformative potential of VAE in the realm of visual reinforcement learning becomes apparent. By capitalizing on its generative prowess, structured representations, uncertainty estimation, flexibility, and regularization attributes, VAE emerges as a pivotal tool in elevating the efficiency, resilience, and adaptability of reinforcement learning agents in intricate visual landscapes.

## II. LITERATURE REVIEW

### A. Reinforcement Learning

Human learning and adaptation are intrinsically tied to interaction with the environment, encompassing the ability to perceive surroundings, make decisions, and learn from consequences without explicit instructions. Building upon this human learning, the subfield of machine learning known as reinforcement learning centres around the concept of goal-directed learning [11][12].

Reinforcement learning presents a unique paradigm within machine learning, where a learning agent, devoid of prior knowledge about optimal actions, aims to identify actions that maximize rewards [13]. Unlike conventional machine learning paradigms such as supervised and unsupervised learning, reinforcement learning exhibits distinct characteristics [14]:

1) Trial and Error: Reinforcement learning hinges on a trial-and-error process. The agent systematically explores and takes actions within its environment,

learning from the outcomes of these actions.

2) Reinforcement Signal: Learning in reinforcement learning is propelled by a reinforcement signal, offering feedback on the quality of the agent's actions. This signal guides the agent toward actions that lead to favourable outcomes.

3) Delayed Reward: A defining feature of reinforcement learning is grappling with delayed rewards. Actions may have consequences not immediately evident, necessitating the agent to consider long-term implications [15].

4) Exploration and Exploitation: Reinforcement learning agents must strike a balance between exploration (trying new actions to discover their effects) and exploitation (choosing actions known to yield high rewards). This trade-off is pivotal for learning optimal policies [16].

Overall, in reinforcement learning, an agent endeavours to uncover optimal actions through iterative trials and errors. It interacts with the environment, experiencing rewards and penalties, and leverages these experiences to refine its actions over time. This iterative process equips the agent to acquire an optimal strategy that maximizes cumulative rewards [17]. Moreover, reinforcement learning grapples with the notion of delayed rewards, compelling the agent to anticipate and weigh the future consequences of its actions while striving for long-term objectives. Consistently selecting actions that yield higher rewards constitutes the central objective in maximizing cumulative returns [18].

Machine learning is primarily composed of three main categories: supervised learning, unsupervised learning, and reinforcement learning [19]. Each category has unique characteristics, with supervised learning relying on labelled data for precise predictions [20], while unsupervised learning discovers hidden patterns in data, often with reduced accuracy. In contrast, reinforcement learning, although more challenging to implement, excels at solving complex problems involving decision-making that can overwhelm other learning approaches.

Understanding several key components is important to fully grasp the fundamental principles of reinforcement learning, and the structure is as shown in Fig. 1:

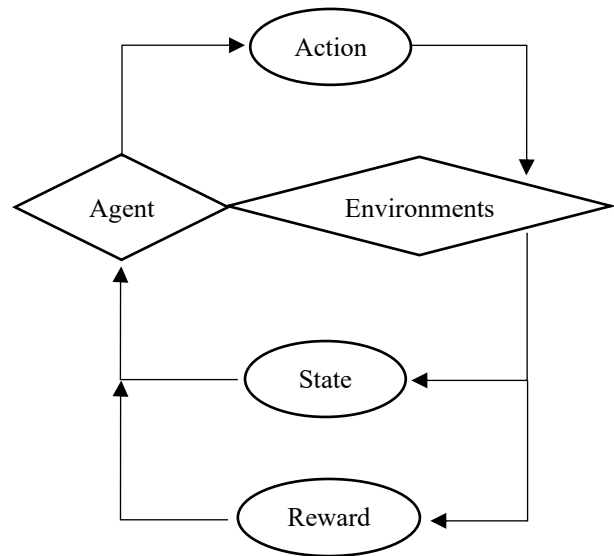


Fig. 1. The Overall Structure of Reinforcement Learning

1) Agent: The decision-maker within the environment, responsible for interacting with it. Agents make decisions based on policies and receive rewards corresponding to their actions.

2) Environment: The stage in which the agent operates, defining states, actions, and rewards. Interactions between the agent and the environment propel the learning process.

3) State: Represents the current context or situation of the agent within the environment. States encapsulate pertinent information, influencing the agent's decision-making process. A state's value takes into account its long-term desirability, considering expected future states and rewards.

4) Action: Denotes the choices available to the agent in response to its current state. Agents select actions with the aim of maximizing cumulative rewards.

5) Reward: A signal received by the agent after taking an action in a specific state. Rewards serve as the agent's motivation for learning and enhancing decision-making. Positive rewards reinforce favourable actions, while negative rewards discourage unfavourable ones [21].

In reinforcement learning, the ultimate objective for the agent is to optimize cumulative rewards. Rewards function as the compass directing the agent toward actions that lead to positive outcomes while steering it away from actions bearing negative consequences.

### B. Autoencoder

An autoencoder represents a neural network model designed to produce output values closely mirroring the input values [22]. To achieve this, it employs the

concepts of an encoder and a decoder (shown in Fig. 2). Through unsupervised learning, an autoencoder autonomously extracts features from input data, effectively reducing high-dimensional, intricate data to a lower dimension while retaining essential information [23][24]. As data dimensionality escalates, computational complexity follows suit, rendering the handling and processing of such data a formidable challenge.

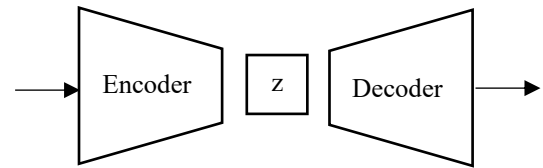


Fig. 2. Overall Structure of Autoencoder

Autoencoders surmount this challenge by efficiently extracting the core facets of data using a limited number of neurons or neuron layers. The primary objective is not to achieve perfect one-to-one accuracy but rather to create an encoding function that closely approximates input values within a constrained latent space.

The core features extracted from the input data are referred to as the "Latent Vector," denoted as "z" after undergoing encoding [25]. Assessing the quality of this latent vector holds paramount importance. Autoencoders employ the decoder to reverse the created latent vector via an inverse process, aspiring to reconstruct the original data as faithfully as possible for comparison with the input prior to entering the encoder. This process permits accuracy assessment and model refinement by evaluating the disparity between the original data and the reconstruction.

The latent vector serves as a technique for representing data in a space more conducive to data



reduction compared to its original space [26]. When dealing with data, preserving all essential facets without omission assumes critical importance. Lower dimensions are preferred due to their reduced data processing requirements and model simplification. In this context, the encoder imposes constraints to forge small and efficient latent vectors, leading to significant dimensionality reduction [28]. The decoder functions as a means to validate and measure whether the encoder has accurately and appropriately reduced dimensions.

### C. Variational Autoencoder

A Variational Autoencoder (VAE) stands as an advanced neural network structure that extends upon the foundational concept of the traditional autoencoder [29]. While the primary objective remains encoding input values into output values closely resembling the input, VAE introduces a probabilistic approach. It represents the values of the extracted latent code not as single numbers but as probability values predicated on a Gaussian probability distribution, engendering more versatile latent codes.

VAE, akin to other generative models such as Generative Adversarial Networks (GAN) and diffusion models, operates on the premise of creating a meaningful latent space and generating data within it. In VAE, the latent space is conceived through an encoder and decoder pair, mirroring an autoencoder (shown in Fig. 3) [30]. The fundamental aspiration of VAE centres on generating data that closely approximates real-world data.

The VAE process commences by extracting features from input data, such as images, and storing them in a latent vector "z" [31]. This latent vector "z" serves as the bedrock for generating entirely new data. VAE presumes that each feature conforms to a Gaussian distribution, with the latent vector "z" denoting the mean and variance of these features. The encoder takes input data "x" and transmutes it into the latent vector "z", while the decoder reverses this process to generate data closely resembling the original input "x". VAE establishes probability distributions for individual features within the input data. By pinpointing areas of high probability, VAE can generate entirely new data points likely to exist in reality. The model's parameters, represented as " $\theta$ ", are learned to maximize the probability of the input data " $p_\theta(x)$ ", endeavouring to approximate the authentic distribution of the data " $p(x)$ ".

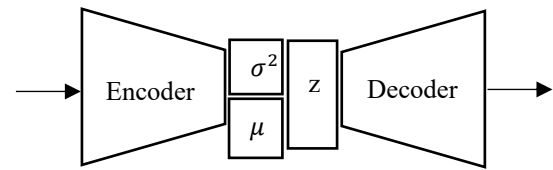


Fig. 3. Overall structure of Variational Autoencoder

The encoder's role encompasses mapping input data into the latent space and approximating the posterior distribution of the latent vector "z" given the input data "x", denoted as " $q(z|x)$ " [32]. Conversely, the decoder transforms the latent vector back into the input space, aspiring to approximate the true distribution of the input data given the latent vector "z", denoted as " $p(x|z)$ " [33]. The latent space embodies concealed vectors, and to preclude the perpetual generation of data possessing identical characteristics to the input, noise is introduced during the formation of the latent vector. This noise is drawn from a standard normal distribution (mean = 0, standard deviation = 1), scaled by the variance

obtained from the encoder, and added to the mean, yielding the latent vector "z". This technique is known as the "Reparameterization Trick."

Training a VAE hinges on maximizing the likelihood by minimizing the loss [34]. The principal objective entails discovering the parameter " $\theta$ " that maximizes " $p_\theta(x)$ ". This can be expressed as the maximization of the log-likelihood, as depicted by the ensuing equation:

$$\log p_\theta(x) = \int q_\phi(z|x) \log p_\theta(x) dz \quad (1)$$

$$= \int q_\phi(z|x) \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} dz \quad (2)$$

$$= \int q_\phi(z|x) \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} \frac{q_\phi(z|x)}{q_\phi(z|x)} dz \quad (3)$$

$$= \int q_\phi(z|x) \log p_\theta(x|z) dz - KL(q_\phi(z|x)||p(z)) + KL(q_\phi(z|x)||p_\theta(z|x)) \quad (4)$$

Navigating through Bayes' rule, the equation underscores the significance of the encoder, decoder, and the latent space in approximating the genuine distribution of the input data.

The negative cross-entropy between " $p(x|z)$ " and " $q(z|x)$ ", constituting the reconstruction error [35]. It encourages the encoder and decoder, resembling an autoencoder, to excel in reconstruction. Moreover, the Kullback-Leibler ( $KL$ ) divergence between the posterior " $q(z|x)$ " and the prior " $p(z)$ " referred to as the regularization error (Eq. (4)) [36]. It encourages the posterior and prior to closely resemble each other,

preventing VAE from excelling solely in reconstruction tasks.

$$\geq \int q_\phi(z|x) \log p_\theta(x|z) dz - KL(q_\phi(z|x)||p(z)) \quad (5)$$

By maximizing the Evidence Lower Bound ( $ELBO$ ) (Eq. (5)), a lower bound of the log-likelihood, and training both the encoder and decoder, VAE learns to generate new data points [37]. This process entails sampling "z" from a standard normal distribution and employing this "z" as input for the decoder to generate new data.

This comprehensive exploration has elucidated the intricate landscape of reinforcement learning, autoencoders, and VAE. These components constitute the foundational knowledge necessary to delve into the transformative role of VAE in reinforcement learning.

### III. METHODOLOGY

#### A. Algorithm of VAE

The encoder takes an input tensor representing an image and produces two output tensors, " $z\_mean$ " and " $z\_log\_var$ ", representing the mean and logarithm of the variance of the latent distribution. The " $z$ " tensor is generated using the reparameterization trick, which samples from a normal distribution with the specified mean and variance. The decoder takes the latent variables "z" as input and produces an output tensor that represents the reconstructed image. The decoder consists of a single dense layer with a sigmoid activation function, which is used to reconstruct the original image.

---

**Algorithm 1: Abstractor with VAE**

---

```
class VAEAbstractor():  
    def __init__(self, images, latent_dim):  
        # Define encoder  
        inputs = Input(shape=(image_dim,))  
        z_mean = Dense(latent_dim)(inputs)  
        z_log_var = Dense(latent_dim)(inputs)  
        z = z_mean + exp(0.5 * z_log_var) *  
        random_normal(shape=(batch_size,  
        latent_dim))  
        encoder = Model(inputs, [z_mean,  
        z_log_var, z])  
        # Define decoder  
        latent_inputs =  
        Input(shape=(latent_dim,))  
        outputs = Dense(original_dim,  
        activation='sigmoid')(latent_inputs)  
        decoder = Model(latent_inputs,  
        outputs)  
        # Define VAE  
        outputs = decoder(encoder(inputs)[2])  
        vae = Model(inputs, outputs)  
        # Compile and train VAE  
        vae.compile(optimizer='adam')  
        vae.fit(x_train, epochs=epochs,  
        batch_size=batch_size,  
        validation_data=(x_test, None))
```

---

The "output" tensor of the VAE is the reconstructed image obtained by passing the latent variables through the decoder. The input to the encoder is the original image tensor. The Adam optimizer is used and the model is trained using the training data "x\_train" for a specified number of epochs. The batch size is also specified for training. The validation data "x\_test" is used for validation during training. The full algorithm is presented in Algorithm 1.

**B. Layer of VAE****1) Encoder**

"encoder\_input" is an input layer that takes data with a shape of "(None, 76800)", where "None" represents the batch size. "dense" is a fully connected layer with an output shape of "(None, 512)", containing 39,322,112 parameters. It transforms the input data into a smaller representation. "z\_mean" and "z\_log\_var" are layers produce the mean and logarithm of the variance of the latent space distribution "(None, 7)" respectively, each with 3,591 parameters. "z(Lambda)" is a custom Lambda layer that takes "z\_mean" and "z\_log\_var" to create a latent vector (shown in Table. 1).

Layer(type)	Output Shape	Parameters
encoder_input	(None, 76800)	0
dense	(None, 512)	39,322,112
z_mean	(None, 7)	3,591
z_log_var	(None, 7)	3,591
z (Lambda)	(None, 7)	-

Table. 1. Layer of Encoder

**2) Decoder**

"z\_sampling" is an input layer taking data with a shape of "(None, 7)" which represents a sampled vector from the latent space. "dense\_1" is a fully connected layer with an output shape of "(None, 512)" and 4,096 parameters. It transforms the latent vector into a higher-dimensional representation and "dense\_2" is another fully connected layer producing an output shape of "(None, 76800)" with 39,398,400 parameters. This layer generates the final output data (shown in Table. 2).

Layer(type)	Output Shape	Parameters
z_sampling	(None, 7)	0
dense_1	(None, 512)	4,096
dense_2	(None, 76800)	39,398,400

Table. 2. Layer of Decoder

### 3) VAE Model

" *encoder\_input* " is the input layer with " (None,76800) " to receive the input data. "*encoder (Fuctional)*" is the encoder component, which takes the "*encoder\_input*" and outputs both the mean and log variance of the latent space, as well as the custom "z" vector. "*decoder (Fuctional)*" is the decoder component, taking the sampled "z" vector from the encoder and generating output data. "*Other Layers*" are various intermediate layers for calculations related to loss and reconstruction (shown in Table. 3).

Layer(type)	Output Shape	Parameters
encoder_input	(None, 76800)	0
encoder (Functional)	(None, 7)	39,329,294
decoder	(None, 7)	39,402,496
dense	(None, 512)	39,322,112
z_log_var	(None, 7)	3,591
z_mean	(None, 7)	3,591
Other Layers	...	...

Table. 3. Layer of the VAE model

The VAE architecture comprises an encoder, transforming input data into a smaller representation, a decoder that generates the final output, and intermediate layers for loss calculations, enabling efficient data transformation.

## IV. RESULTS

### A. Parameters

Trainable parameters are learned from the data during training and contribute to the model's ability to fit the training data, and the result values is 78,731,790. Non-trainable parameters are fixed or constrained values that are not updated during training and may include pretrained weights, regularization terms, or fixed components, and in the experiment, it is zero. Therefore, the number of total parameters is 78,731,790.

### B. Loss

The training loss in this case is a crucial metric for assessing how effectively the VAE is learning to represent visual data and how well it aligns with the objectives of the task, which may include image generation, object recognition, or other robotic vision applications. Lower training loss values typically indicate better performance, although other evaluation metrics, such as validation loss or downstream task performance, should also be considered for a comprehensive assessment of the model.

In the case of one object ( $n_{obj} = 1$ ), the training and validation loss values gradually decreased over the epochs. The model reached a relatively low validation loss of around 9765.5 after 30 epochs. The model learned to approximate the optimal solution for the single object it was trained on. The decreasing loss values indicate that the model's performance improved over time (shown in Fig. 4).

For  $n_{obj} = 2$ , similar to the case with a single object,

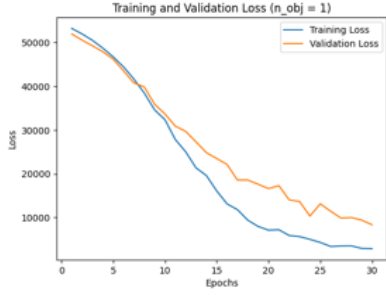


Fig. 4. Loss of  $n_{obj} = 1$

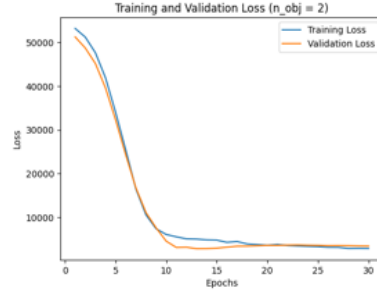


Fig. 5. Loss of  $n_{obj} = 2$

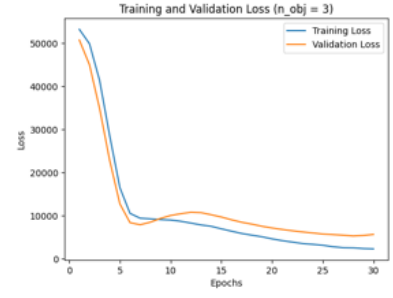


Fig. 6. Loss of  $n_{obj} = 3$

the training and validation loss values decreased as the epochs progressed. The validation loss reached around 3366.6 after 30 epochs. While the validation loss is comparable to the  $n_{obj} = 1$  case, the additional object adds complexity to the problem.

In the case of  $n_{obj} = 3$ , as before, the training and validation loss values decreased with increasing epochs. The validation loss reached around 5625.6 after 30 epochs. The validation loss is higher compared to the previous cases ( $n_{obj} = 1$  and  $n_{obj} = 2$ ).

### C. Representation

The representation part consists of a total of three parts: Abstract Representation, 2D projection, and Abstract Goal (Precondition of Action). Firstly, the abstract representation part represents the initial state or configuration of the environment when there is object. It encodes information about the object's position and orientation. Secondly, the 2D projection captures important information about the scene but in a simplified form, which can be useful for planning and decision-making. Finally, the third part represents the abstract goal or precondition for an action or task in the environment with object. It encodes information about where the system should aim to achieve a

specific task. The current and goal images generated through these representation components are presented (show in Fig. 7, Fig. 8, and Fig. 9).

#### 1) Abstract Start Representation

$n_{obj} = 1$ :

[-14.72412, 11.302932, -15.957655,  
-8.749756, -27.958857, 12.964725, 13.961952]

$n_{obj} = 2$ :

[-14.378856, -11.2165365, 22.159285,  
-21.26708, 11.030341, -7.0615177, 14.390376,  
-20.547997, -7.8535643, 14.510471, 6.5396295,  
12.8372755, -8.66571, -28.457037]

$n_{obj} = 3$ :

[ 2.1450174, -2.9223979, -10.061906,  
-1.9011259, -17.701748, -1.500329, -7.0388083,  
-22.1439, 9.582621, -15.650179, -1.9188226,  
15.0972185, -2.494203, -8.466422, -2.7144487, -  
8.775903, -26.2219, 5.525549, 5.499649,  
15.942222, 3.716352]

#### 2) 2D Projection

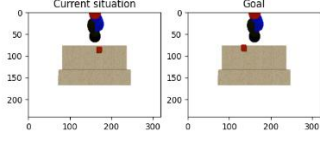


Fig. 7. Image of situation ( $n_{obj} = 1$ )

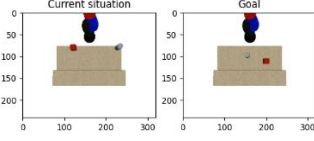


Fig. 8. Image of situation ( $n_{obj} = 2$ )

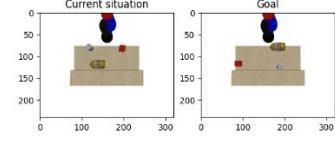


Fig. 9. Image of situation ( $n_{obj} = 3$ )

$n_{obj} = 1$ :

0.17791939, -0.17952916, -0.2187883]

[[ 0.0317128, 0.12910708],

[-0.10726225, -0.03727605]]

$n_{obj} = 2$ :

[[-0.19915462, 0.25517428],

[ 0.00360761, -0.02420754]]

$n_{obj} = 3$ :

[[-0.24262343, -0.17147487],

[ 0.03366467, -0.16723716]]

3) Abstract Goal (Precondition of Action)

$n_{obj} = 1$ :

[ -9.943, 13.600912, -22.900776,

-12.723337 -23.143553 ,8.5652895, 9.460598]

$n_{obj} = 2$ :

[-0.19584027, -0.16707355, 0.21172337,

-0.19525324, 0.11948041, -0.11767057, 0.14446048,

-0.19194908, -0.18396583, 0.17769301, 0.07229026,

$n_{obj} = 3$ :

[ 0.32495028, -0.22297516, -0.3105173,

-0.09223183, -0.25559965, -0.3245753,

-0.05757309, -0.23426968, 0.11859599,

-0.15786745, -0.12317993, 0.41920805,

-0.10139299, -0.24926186, 0.28432047,

-0.23734269, -0.34312466, 0.19089322, 0.2987334,

0.26860636, -0.23382592]

## V. DISCUSSIONS

### A. Trainable and Non-trainable Parameters

In the realm of neural networks, the distinction between trainable and non-trainable parameters forms a fundamental concept. Trainable parameters encompass the weights and biases embedded within the neural network architecture, subject to iterative adjustments during the model's training process. These adjustments are aimed at minimizing the model's loss function, which, in turn, empowers the network to make predictions and glean insights from data.

Conversely, non-trainable parameters represent

elements within the neural network that remain fixed and unaltered throughout the training process. These parameters find specific utility in particular layers or components of neural networks, such as batch normalization or layers integrating pre-trained weights. Non-trainable parameters serve distinct purposes, often associated with regularization or the incorporation of prior knowledge into the model. In some instances, non-trainable parameters are deployed to enforce constraints on the network's behaviour or to maintain stability during training [38].

It is worth noting that within the context of this research, there are scenarios in which a neural network may not incorporate non-trainable parameters. In such cases, the entire architectural framework relies on the adaptability of trainable parameters to effectively capture and represent the underlying data patterns. This research underscores the absence of non-trainable parameters, emphasizing the network's dependence on learning and adaptation achieved through the optimization of trainable parameters.

### *B. Impact of the Number of Objects on Training Loss*

An intriguing aspect of this research revolves around the analysis of how the number of objects within a scene influences the training loss of the VAE model. This analysis sheds light on the intricate relationship between scene complexity and the convergence behaviour of the VAE during its training regimen [39].

In scenarios where a solitary object inhabits the scene, the training loss typically exhibits swift convergence toward a low value. This phenomenon is attributed to the inherent simplicity of the VAE's task when dealing

with a solitary object. The encoder-decoder architecture of the VAE is designed to encode and decode a single object, a task inherently less complex than handling multiple objects. Consequently, the loss curve for this scenario mirrors a decreasing exponential trend, characterized by rapid initial convergence followed by progressively slower improvements as training advances. Ultimately, the loss value upon completion of training tends to be notably low, signalling the VAE's proficiency in encoding and decoding solitary objects.

However, as the scene's complexity escalates with the introduction of two objects, the training loss may converge to a slightly higher value than that observed in the one-object scenario, although it remains lower than scenarios featuring three objects. The loss curve continues to manifest a decreasing trend, but the convergence may demand more time compared to the one-object scenario. While the VAE grapples with the augmented complexity of encoding and decoding two objects concurrently, it still manages to attain a lower loss compared to scenarios involving more objects.

Training a VAE with three objects represents the most complex scenario, mandating the model to simultaneously encode and decode multiple objects. In such instances, the training loss may converge to a higher value relative to both the one-object and two-object scenarios. The loss curve maintains its decreasing trajectory, but the convergence may be sluggish and may not reach as low a value as observed in the less intricate scenarios. This phenomenon reflects the heightened complexity associated with managing multiple objects within the scene.

This research unveils that the number of objects exerts a direct influence on VAE training complexity. Simpler scenarios featuring fewer objects yield faster convergence and lower training losses, whereas more intricate scenarios with multiple objects result in slower convergence and higher training losses, courtesy of the augmented intricacies entailed in encoding and decoding multiple objects.

### *C. Manifold Learning and VAE Components*

One of the core principles anchoring this research is the concept of manifold learning and its pivotal role in amplifying the efficiency and efficacy of the VAE model. Grasping this concept necessitates a comprehensive exploration of the roles executed by the diverse components of the VAE architecture: the encoder, VAE layer, and decoder.

Manifold learning functions as a cornerstone dimensionality reduction technique. Its primary mission is to map high-dimensional data points, such as images or state representations, to a lower-dimensional latent space, all the while preserving the intrinsic structure and interrelationships between data points originating from the original space [40][41].

The encoder within the VAE architecture emerges as a central figure in the manifold learning process [42]. Its core duty is to effectuate the mapping of high-dimensional data points from the original space to a lower-dimensional latent space. This mapping ideally conserves the intrinsic structure and relationships between data points, ensuring that analogous inputs correspond to proximate points in the latent space. The encoder accomplishes this by mastering a set of

parameters designed to capture the underlying patterns and interdependencies nestled within the data. These parameters undergo fine-tuning during the training process, empowering the encoder to adeptly execute the mapping of high-dimensional data to lower-dimensional representations that faithfully encapsulate the data's critical characteristics.

The VAE layer, a distinctive feature, unifies the encoder and decoder within a probabilistic framework. It imposes specific constraints on the latent space, prompting it to exhibit fundamental attributes such as continuity, smoothness, and adherence to a predefined Gaussian. These constraints stand as essential prerequisites for effective manifold learning within the VAE. The probabilistic nature of the VAE layer introduces an element of uncertainty into the encoding and decoding process, empowering the model to manipulate and generate data that adheres to the underlying structure of the data manifold.

Crucially, the VAE layer embodies two principal components—the encoder and the decoder—working in concert to realize manifold learning objectives [43][44]. While the encoder centres on mapping high-dimensional data to the latent space, the decoder's role pivots around reversing this process, translating latent space points back into meaningful data points within the original space. This bidirectional flow of information assumes paramount importance in safeguarding the intrinsic structure of the data manifold. Essentially, the decoder ensures that proximate points within the latent space correspond to akin data points in the original space, upholding the manifold's integrity.



In summation, manifold learning, shepherded by the VAE's encoder, VAE layer, and decoder, functions as a linchpin of the research. The encoder adeptly captures crucial features and dependencies nestled within the data, enabling an accurate mapping to the latent space. The VAE layer introduces critical constraints governing the latent space, shaping it to align with manifold learning principles. Meanwhile, the decoder, while primarily concerned with data reconstruction, indirectly contributes to manifold learning by safeguarding that the latent space accurately mirrors the intrinsic structure of the data.

#### *D. Manifold Assumption and Dimensionality Reduction*

The manifold assumption, a foundational concept underlying this research, provides a theoretical scaffold for comprehending why dimensionality reduction techniques can wield remarkable efficacy and utility across various data analysis scenarios. It postulates that real-world, high-dimensional data often harbours underlying lower-dimensional structures or manifolds [45]. These manifolds encapsulate vital features and interdependencies within the data, furnishing a means to simplify and elucidate the data without incurring substantial information loss [46].

The application of dimensionality reduction techniques, guided by the manifold assumption, bestows several compelling advantages. Foremost among them is the enhancement of data visualization, achieved through the projection of high-dimensional data onto lower-dimensional spaces, rendering it more accessible for human interpretation. In the context of complex datasets replete with numerous dimensions,

this visualization emerges as a potent tool for gaining insights and discerning data patterns.

Furthermore, dimensionality reduction contributes to heightened computational efficiency. High-dimensional data can pose considerable computational challenges, rendering specific machine learning tasks impracticable. By curbing data dimensionality, these techniques mitigate these computational burdens, facilitating the efficient processing of voluminous datasets and expediting machine learning algorithms.

Dimensionality reduction can also confer improvements in the generalization performance of machine learning models. By culling noisy or irrelevant dimensions and honing in on the most informative features, dimensionality reduction techniques steer models toward superior generalization to unseen data. This reduction in the risk of overfitting enhances model robustness.

However, it is crucial to approach dimensionality reduction with careful consideration of the dataset's specific attributes and the analysis objectives. While the manifold assumption intimates the presence of lower-dimensional structures, not all datasets subscribe to this assumption. Therefore, dimensionality reduction techniques must be tailored to align with the dataset's unique characteristics, being judiciously applied to bolster data comprehension and utility.

Dimensionality reduction techniques, fortified by the manifold assumption, represent a potent instrument in data analysis. They provide the means to distil

intricate data, enrich visualization, bolster computational efficiency, and enhance the generalization capabilities of machine learning models. Nonetheless, the application of these techniques must be carried out judiciously, always mindful of the dataset's attributes and the specific analysis goals.

## VI. CONCLUSIONS

This report emphasizes the potential of VAE as powerful tools for representing and navigating complex and multi-dimensional environments. Through this research, manifold learning orchestrated by VAE provides a way to simplify high-dimensional state representations while retaining the intrinsic structure of the data. This work has yielded important findings and opened up possibilities for further exploration, making significant contributions to the field.

One of the primary achievements of this project lies in the successful implementation of VAE-based abstractors. These abstractors have demonstrated their adeptness in reducing the dimensionality of intricate state representations within robotic environments. By encoding these representations into a lower-dimensional latent space, the power of dimensionality reduction is harnessed without relinquishing the critical details and relationships inherent in the data. This accomplishment carries immediate implications for self-efficient robotic systems and streamlined decision-making processes, paving the way for leaner and more agile robots capable of operating in complex environments.

Moreover, this approach's ability to preserve the structure of the manifold is a crucial feature. The VAE abstractor, equipped with a probabilistic framework and manifold-enforcing constraints, has proven to be a reliable guardian of the core structure of the data. This preservation is evident in the smooth interpolation between different environmental states and the generation of data samples consistent with the underlying structure learned during training. This capability plays a significant role in enhancing the adaptability and performance of robotic systems, especially in dynamic and complex scenarios where understanding the underlying structure is paramount.

This research has revealed a subtle relationship between the complexity of the robotic environment and the dynamics of the training process. Specifically, the number of objects present in the environment directly influences the training dynamics. In simpler scenarios with a single object, the VAE quickly converges to low loss values, effectively capturing the essence of the solitary object. However, as the complexity increases with two or three objects, the training process becomes more intricate, leading to slower convergence and resulting in higher final loss values. These observations highlight the complexities associated with dealing with complex robotic environments and emphasize the need for adaptive learning strategies and nuanced approaches to effectively address such intricacies.

The synergy between the encoder and decoder components of the VAE architecture has been pivotal in these accomplishments. The encoder excels at mapping high-dimensional data to a lower-dimensional latent space, capturing hierarchical

features and preserving relationships within the data. The decoder, in its turn, expertly reconstructs data from latent space representations, ensuring that nearby points in the latent space correspond to meaningful data points in the original space. This intricate interplay between encoder and decoder has been instrumental in achieving effective manifold learning, facilitating seamless transitions between different states, and empowering the VAE to navigate the complex data manifold effectively.

Leveraging abstract representations for reinforcement learning tasks within robotic environments stands as a natural progression, and the potential benefits of such an approach warrant comprehensive exploration. Furthermore, the practical application of these representations in real-world robotic scenarios holds significant promise and necessitates further investigation. By bridging the gap between abstract representations and practical robotic applications, a new generation of adaptable and intelligent robotic systems can be envisioned.

In conclusion, this report represents a significant advancement in understanding of manifold learning in the context of robotic vision, underscoring the profound relationship between VAE and reinforcement learning. It highlights the prowess of VAE-based abstractors in simplifying and enhancing complex environmental representations. The lessons garnered from the exploration of training dynamics and the practical utility of abstract representations lay the groundwork for future advancements in reinforcement learning, control, and decision-making in robotic systems. This work reaffirms the significance of VAE as indispensable tools in the quest

for adaptable and intelligent robotic systems in a world teeming with complexity.

## ACKNOWLEDGMENTS

Thank you to my supervisor, Dr Lorenzo Jamone, for providing feedback and guidance throughout this project. Thanks also to my family and friends for comforting and supporting me in this whole journey.

## REFERENCES

- [1] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–503, 2016, doi: 10.1038/nature16961.
- [2] D. Ha and J. Schmidhuber, "World Models," *Forecast. Bus. Econ.*, pp. 201–209, 1980, doi: 10.1016/b978-0-12-295180-0.50030-6.
- [3] V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," pp. 1–9, 2013.
- [4] J. Schrittwieser et al., "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020, doi: 10.1038/s41586-020-03051-4.
- [5] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of Real-World Reinforcement Learning," *Springer*, vol. 110, pp. 2419–2468, 2019.
- [6] T. Yu, G. Shevchuk, D. Sadigh, and C. Finn, "Unsupervised Visuomotor Control through Distributional Planning Networks," *Robot. Sci. Syst.*, 2019, doi: 10.15607/RSS.2019.XV.020.
- [7] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," 2nd Int. Conf. Learn. Represent. *ICLR 2014 - Conf. Track Proc.*, no. ML, pp. 1–14, 2014.
- [8] I. Higgins et al., " $\beta$ -VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK," *ICLR*, pp. 1–22, Jul. 2022, [Online].

Available: <https://openreview.net/forum?id=Sy2fzU9gl>.

[9] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," *ICML*, pp. 1–10, 2016.

[10] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *CoNLL 2016 - 20th SIGNLL Conf. Comput. Nat. Lang. Learn. Proc.*, pp. 1–12, 2016, doi: 10.18653/v1/k16-1002.

[11] P. Dayan and Y. Niv, "Reinforcement learning: The Good, The Bad and The Ugly," *Curr. Opin. Neurobiol.*, vol. 18, no. 2, pp. 1–12, 2008, doi: 10.1016/j.conb.2008.08.003.

[12] D. Held, X. Geng, C. Florensa, and P. Abbel, "Automatic Goal Generation for Reinforcement Learning Agents," *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 4, pp. 2458–2471, 2018.

[13] Y. Niv et al., "Reinforcement Learning in Multidimensional Environments Relies on Attention Mechanisms," *J. Neurosci.*, vol. 35, no. 21, pp. 8145–8157, 2015, doi: 10.1523/JNEUROSCI.2978-14.2015.

[14] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1–2, pp. 99–134, 1998.

[15] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction second edition. *The MIT Press*, 2018. [Online]. Available: <https://lccn.loc.gov/2018023826>.

[16] A. L. Strehl and M. L. Littman, "An analysis of model-based Interval Estimation for Markov Decision Processes," *J. Comput. Syst. Sci.*, vol. 74, no. 8, pp. 1309–1331, 2008, doi: 10.1016/j.jcss.2007.08.009.

[17] B. Eysenbach, J. Ibarz, A. Gupta, and S. Levine, "DIVERSITY IS ALL YOU NEED: LEARNING SKILLS WITHOUT A REWARD FUNCTION," *7th Int. Conf. Learn. Represent. ICLR 2019*, pp. 1–22, 2019.

[18] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013, doi: 10.1177/0278364913495721.

[19] E. Alpaydm, "Introduction to Machine Learning Second Edition." 2014.

[20] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven Exploration by Self-supervised Prediction," *34th Int. Conf. Mach. Learn. ICML 2017*, vol. 6, pp. 4261–4270, 2017.

[21] P. Kormushev, S. Calinon, and D. G. Caldwell, "Reinforcement Learning in Robotics: Applications and Real-World Challenges," *Robotics*, vol. 2, no. 3, pp. 122–148, 2013, doi: 10.3390/robotics2030122.

[22] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[23] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013, doi: 10.1109/TPAMI.2013.50.

[24] H. Zhu et al., "The Ingredients of Real-World Robotic Reinforcement Learning," *8th Int. Conf. Learn. Represent. ICLR 2020*, pp. 1–20, 2020.

[25] T. Malloy, T. Klinger, and C. R. Sims, "Modeling Human Reinforcement Learning with Disentangled Visual Representations," pp. 1–4, 2022.

[26] S. Ayesha, M. K. Hanif, and R. Talib, "Overview and comparative study of dimensionality reduction techniques for high dimensional data," *Inf. Fusion*, vol. 59, pp. 44–58, 2020, doi: 10.1016/j.inffus.2020.01.005.

[27] T. Lemke and C. Peter, "EncoderMap: Dimensionality Reduction and Generation of Molecule Conformations," *J. Chem. Theory Comput.*, vol. 15, no. 2, pp. 1209–1215, 2019, doi: 10.1021/acs.jctc.8b00975.

[28] P. Baldi and K. Hornik, "Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima," *Neural Networks*, vol. 2, no. 1, pp. 53–58, 1989, doi: 10.1016/0893-6080(89)90014-2.

[29] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S.

- Levine, "Visual Reinforcement Learning with Imagined Goals," *Adv. Neural Inf. Process. Syst.*, pp. 1–10, 2018.
- [30] D. J. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows," *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 2, pp. 1–10, 2016.
- [31] E. Cartoni, D. Montella, J. Triesch, and G. Baldassarre, "REAL-X — Robot open-Ended Autonomous Learning Architectures: Achieving Truly End-to-End Sensorimotor Autonomous Learning Systems," pp. 1–14, 2022.
- [32] C. Doersch, "Tutorial on Variational Autoencoders," pp. 1–23, 2016.
- [33] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "DRAW: A Recurrent Neural Network For Image Generation," *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 2, pp. 1–10, 2015.
- [34] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.
- [35] M. Ranzato and M. Szummer, "Semi-supervised Learning of Compact Document Representations with Deep Networks," *Proc. 25th Int. Conf. Mach. Learn.*, pp. 1–8, 2008, doi: 10.1145/1390156.1390256.
- [36] D. Barber, *Bayesian Reasoning and Machine Learning*. 2012. doi: 10.1017/cbo9780511804779.026.
- [37] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational Inference: A Review for Statisticians," *J. Am. Stat. Assoc.*, vol. 112, no. 518, pp. 859–877, 2018, doi: 10.1080/01621459.2017.1285773.
- [38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016. [Online]. Available: <https://www.deeplearningbook.org/>.
- [39] C. M. Bishop, *Pattern Recognition and Machine Learning*. 2021. doi: 10.1007/978-3-030-57077-4\_11.
- [40] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 209, no. 5500, pp. 2323–2326, 2000.
- [41] A. J. Izenman, "Introduction to manifold learning," *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 4, no. 5, pp. 439–446, 2012, doi: 10.1002/wics.1222.
- [41] Z. Zhang, J. Wang, and H. Zha, "Adaptive Manifold Learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 253–265, 2012, doi: 10.1109/TPAMI.2011.115.
- [43] T. Lin and H. Zha, "Riemannian manifold learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 796–809, 2008, doi: 10.1109/TPAMI.2007.70735.
- [44] A. Talwalkar, S. Kumar, and H. Rowley, "Large-Scale Manifold Learning," *26th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR*, 2008, doi: 10.1109/CVPR.2008.4587670.
- [45] M. Edraki, "Implication of Manifold Assumption in Deep Learning Models for Computer Vision Applications," pp. 1–120, 2021, [Online]. Available: <https://stars.library.ucf.edu/cgi/viewcontent.cgi?article=1674&context=etd2020>
- [46] J. He, L. Ding, L. Jiang, Z. Li, and Q. Hu, "Intrinsic dimensionality estimation based on manifold assumption," *J. Vis. Commun. Image Represent.*, vol. 25, no. 5, pp. 740–747, 2014, doi: 10.1016/j.jvcir.2014.01.006.

## APPENDIX

### Appendix A

Model: "*encoder*"

Layer (type)	Output Shape	Param #	Connected to
encoder_input (InputLayer)	[(None, 76800)]	0	
dense (Dense)	(None, 512)	39322112	encoder_input[0][0]
z_mean (Dense)	(None, 7)	3591	dense[0][0]
z_log_var (Dense)	(None, 7)	3591	dense[0][0]
z (Lambda)	(None, 7)	0	z_mean[0][0] z_log_var[0][0]

Model: "*decoder*"

Layer (type)	Output Shape	Param #
z_sampling (InputLayer)	[(None, 7)]	0
dense_1 (Dense)	(None, 512)	4096

dense_2 (Dense)	(None, 76800)	39398400
-----------------	---------------	----------

---

Model: "vae\_mlp"

---

Layer (type)	Output Shape	Param #	Connected to
encoder_input (InputLayer)	[(None, 76800)]	0	
encoder (Functional)	[(None, 7), (None, 7 39329294		encoder_input[0][0]
decoder (Functional)	(None, 76800)	39402496	encoder[0][2]
dense (Dense)	(None, 512)	39322112	encoder_input[0][0]
z_log_var (Dense)	(None, 7)	3591	dense[0][0]
z_mean (Dense)	(None, 7)	3591	dense[0][0]
tf.__operators__.add (TFOpLambd	(None, 7)	0	z_log_var[0][0]
tf.math.square (TFOpLambda)	(None, 7)	0	z_mean[0][0]
tf.cast (TFOpLambda)	(None, 76800)	0	encoder_input[0][0]

tf.convert_to_tensor (TFOpLambda) (None, 76800)	0	decoder[0][0]
tf.math.subtract (TFOpLambda) (None, 7) tf.math.square[0][0]	0	tf.__operators__.add[0][0]
tf.math.exp (TFOpLambda) (None, 7)	0	z_log_var[0][0]
tf.keras.backend.binary_crossentropy (None, 76800)	0	tf.cast[0][0] tf.convert_to_tensor[0][0]
tf.math.subtract_1 (TFOpLambda) (None, 7)	0	tf.math.subtract[0][0] tf.math.exp[0][0]
tf.math.reduce_mean (TFOpLambda (None,))	0	tf.keras.backend.binary_crossentropy
tf.math.reduce_sum (TFOpLambda) (None,)	0	tf.math.subtract_1[0][0]
tf.math.multiply (TFOpLambda) (None,)	0	tf.math.reduce_mean[0][0]
tf.math.multiply_1 (TFOpLambda) (None,)	0	tf.math.reduce_sum[0][0]
tf.__operators__.add_1 (TFOpLambda) (None,) tf.math.multiply_1[0][0]	0	tf.math.multiply[0][0]
tf.math.reduce_mean_1 (TFOpLambda) ()	0	tf.__operators__.add_1[0][0]
add_loss (AddLoss) ()	0	tf.math.reduce_mean_1[0][0]



## Appendix B

$n_{obj} = 1$ )

Epoch 1/30

1/1 [=====] - 5s 5s/step - loss: 53235.7969 - val\_loss: 51995.0664

Epoch 2/30

1/1 [=====] - 3s 3s/step - loss: 51982.3711 - val\_loss: 50580.4492

Epoch 3/30

1/1 [=====] - 4s 4s/step - loss: 49935.6094 - val\_loss: 48657.3555

Epoch 4/30

1/1 [=====] - 3s 3s/step - loss: 46916.2148 - val\_loss: 46267.6094

Epoch 5/30

1/1 [=====] - 3s 3s/step - loss: 42624.2656 - val\_loss: 44415.7617

Epoch 6/30

1/1 [=====] - 3s 3s/step - loss: 37744.6445 - val\_loss: 42191.8516

Epoch 7/30

1/1 [=====] - 3s 3s/step - loss: 31683.0508 - val\_loss: 39471.6953

Epoch 8/30

1/1 [=====] - 3s 3s/step - loss: 26136.4121 - val\_loss: 35336.4258

Epoch 9/30

1/1 [=====] - 3s 3s/step - loss: 19950.3008 - val\_loss: 32509.6250

Epoch 10/30

1/1 [=====] - 3s 3s/step - loss: 15603.5273 - val\_loss: 30817.3633

Epoch 11/30

1/1 [=====] - 3s 3s/step - loss: 12991.1025 - val\_loss: 28107.9551

Epoch 12/30

1/1 [=====] - 3s 3s/step - loss: 11525.2168 - val\_loss: 27300.5332

Epoch 13/30

1/1 [=====] - 3s 3s/step - loss: 10258.0361 - val\_loss: 26925.7695

Epoch 14/30

1/1 [=====] - 3s 3s/step - loss: 10002.6670 - val\_loss: 24032.8105

Epoch 15/30

1/1 [=====] - 3s 3s/step - loss: 9057.2881 - val\_loss: 22797.1426

Epoch 16/30

1/1 [=====] - 3s 3s/step - loss: 8168.9683 - val\_loss: 21747.7754

Epoch 17/30

1/1 [=====] - 3s 3s/step - loss: 7802.6787 - val\_loss: 20363.0078

Epoch 18/30

1/1 [=====] - 3s 3s/step - loss: 7328.7329 - val\_loss: 18668.9082

Epoch 19/30

1/1 [=====] - 3s 3s/step - loss: 5795.3906 - val\_loss: 16842.5801

Epoch 20/30

1/1 [=====] - 3s 3s/step - loss: 5836.6973 - val\_loss: 17342.2949

Epoch 21/30

1/1 [=====] - 3s 3s/step - loss: 5551.6475 - val\_loss: 14684.7812

Epoch 22/30

1/1 [=====] - 3s 3s/step - loss: 5343.6455 - val\_loss: 13805.6758

Epoch 23/30

1/1 [=====] - 3s 3s/step - loss: 4531.8213 - val\_loss: 13867.9277

Epoch 24/30

1/1 [=====] - 3s 3s/step - loss: 4473.7549 - val\_loss: 11144.1865

Epoch 25/30

1/1 [=====] - 3s 3s/step - loss: 3514.7615 - val\_loss: 13449.4316

Epoch 26/30

1/1 [=====] - 3s 3s/step - loss: 3653.3240 - val\_loss: 12141.6895

Epoch 27/30

1/1 [=====] - 3s 3s/step - loss: 3006.4727 - val\_loss: 10912.0625

Epoch 28/30

1/1 [=====] - 3s 3s/step - loss: 2948.6716 - val\_loss: 9550.7070

Epoch 29/30

1/1 [=====] - 3s 3s/step - loss: 3008.6287 - val\_loss: 10350.0449

Epoch 30/30

1/1 [=====] - 3s 3s/step - loss: 2765.4270 - val\_loss: 9765.5166

$n_{obj} = 2$ )

Epoch 1/30

1/1 [=====] - 6s 6s/step - loss: 53236.8906 - val\_loss: 51291.9102

Epoch 2/30

1/1 [=====] - 3s 3s/step - loss: 51259.2227 - val\_loss: 48761.5781

Epoch 3/30

1/1 [=====] - 4s 4s/step - loss: 47680.2070 - val\_loss: 45207.3125

Epoch 4/30

1/1 [=====] - 3s 3s/step - loss: 41943.5430 - val\_loss: 39612.7305

Epoch 5/30

1/1 [=====] - 3s 3s/step - loss: 34023.3867 - val\_loss: 32291.3926

Epoch 6/30

1/1 [=====] - 3s 3s/step - loss: 25421.1465 - val\_loss: 24293.9258

Epoch 7/30

1/1 [=====] - 3s 3s/step - loss: 16498.4121 - val\_loss: 16751.2109

Epoch 8/30

1/1 [=====] - 3s 3s/step - loss: 10530.4375 - val\_loss: 11060.1074

Epoch 9/30

1/1 [=====] - 3s 3s/step - loss: 7347.5083 - val\_loss: 7561.1616

Epoch 10/30

1/1 [=====] - 3s 3s/step - loss: 6035.4966 - val\_loss: 4504.5283

Epoch 11/30

1/1 [=====] - 3s 3s/step - loss: 5502.6377 - val\_loss: 3062.5076

Epoch 12/30

1/1 [=====] - 3s 3s/step - loss: 5030.0259 - val\_loss: 3117.6179

Epoch 13/30

1/1 [=====] - 3s 3s/step - loss: 4977.5479 - val\_loss: 2778.9155

Epoch 14/30

1/1 [=====] - 3s 3s/step - loss: 4802.9790 - val\_loss: 2795.1445

Epoch 15/30

1/1 [=====] - 3s 3s/step - loss: 4744.5249 - val\_loss: 2908.7900

Epoch 16/30

1/1 [=====] - 3s 3s/step - loss: 4237.6982 - val\_loss: 3127.5090

Epoch 17/30

1/1 [=====] - 3s 3s/step - loss: 4424.8589 - val\_loss: 3341.0354

Epoch 18/30

1/1 [=====] - 3s 3s/step - loss: 3868.5579 - val\_loss: 3359.7937

Epoch 19/30

1/1 [=====] - 3s 3s/step - loss: 3728.4609 - val\_loss: 3463.2798

Epoch 20/30

1/1 [=====] - 3s 3s/step - loss: 3555.6399 - val\_loss: 3546.3813

Epoch 21/30

1/1 [=====] - 3s 3s/step - loss: 3749.8279 - val\_loss: 3532.8201

Epoch 22/30

1/1 [=====] - 3s 3s/step - loss: 3500.8779 - val\_loss: 3601.2014

Epoch 23/30

1/1 [=====] - 3s 3s/step - loss: 3402.3782 - val\_loss: 3626.4619

Epoch 24/30

1/1 [=====] - 3s 3s/step - loss: 3317.7883 - val\_loss: 3570.9568

Epoch 25/30

1/1 [=====] - 3s 3s/step - loss: 3250.6924 - val\_loss: 3545.5454

Epoch 26/30

1/1 [=====] - 3s 3s/step - loss: 3102.5674 - val\_loss: 3481.0356

Epoch 27/30

1/1 [=====] - 3s 3s/step - loss: 3085.5837 - val\_loss: 3507.0210

Epoch 28/30

1/1 [=====] - 3s 3s/step - loss: 2843.8728 - val\_loss: 3470.5547

Epoch 29/30

1/1 [=====] - 3s 3s/step - loss: 2867.5286 - val\_loss: 3407.4988

Epoch 30/30

1/1 [=====] - 3s 3s/step - loss: 2838.5520 - val\_loss: 3366.6033

$n_{obj} = 3$ )

Epoch 1/30

1/1 [=====] - 6s 6s/step - loss: 53239.6250 - val\_loss: 50763.7617

Epoch 2/30

1/1 [=====] - 3s 3s/step - loss: 49901.6094 - val\_loss: 45027.2461

Epoch 3/30

1/1 [=====] - 3s 3s/step - loss: 41381.0781 - val\_loss: 34857.2734

Epoch 4/30

1/1 [=====] - 3s 3s/step - loss: 28479.6016 - val\_loss: 22745.0723

Epoch 5/30

1/1 [=====] - 3s 3s/step - loss: 16471.2070 - val\_loss: 12709.7031

Epoch 6/30

1/1 [=====] - 3s 3s/step - loss: 10509.1162 - val\_loss: 8355.0957

Epoch 7/30

1/1 [=====] - 3s 3s/step - loss: 9388.9043 - val\_loss: 7875.4312

Epoch 8/30

1/1 [=====] - 3s 3s/step - loss: 9251.9189 - val\_loss: 8465.6279

Epoch 9/30

1/1 [=====] - 3s 3s/step - loss: 9066.0947 - val\_loss: 9338.4043

Epoch 10/30

1/1 [=====] - 3s 3s/step - loss: 8978.0410 - val\_loss: 10024.9355

Epoch 11/30

1/1 [=====] - 3s 3s/step - loss: 8679.7676 - val\_loss: 10426.0205

Epoch 12/30

1/1 [=====] - 3s 3s/step - loss: 8266.5254 - val\_loss: 10779.3535

Epoch 13/30

1/1 [=====] - 3s 3s/step - loss: 7829.2554 - val\_loss: 10683.1523

Epoch 14/30

1/1 [=====] - 3s 3s/step - loss: 7516.0229 - val\_loss: 10193.1426

Epoch 15/30

1/1 [=====] - 3s 3s/step - loss: 6925.5044 - val\_loss: 9668.0166

Epoch 16/30

1/1 [=====] - 3s 3s/step - loss: 6381.8892 - val\_loss: 9025.6084

Epoch 17/30

1/1 [=====] - 3s 3s/step - loss: 5852.4326 - val\_loss: 8454.7012

Epoch 18/30

1/1 [=====] - 3s 3s/step - loss: 5456.8892 - val\_loss: 8020.8584

Epoch 19/30

1/1 [=====] - 3s 3s/step - loss: 5081.9458 - val\_loss: 7523.7666

Epoch 20/30

1/1 [=====] - 3s 3s/step - loss: 4599.2842 - val\_loss: 7123.8188

Epoch 21/30

1/1 [=====] - 3s 3s/step - loss: 4180.5332 - val\_loss: 6772.4717

Epoch 22/30

1/1 [=====] - 3s 3s/step - loss: 3859.9612 - val\_loss: 6476.9990

Epoch 23/30

1/1 [=====] - 3s 3s/step - loss: 3507.8386 - val\_loss: 6200.0796

Epoch 24/30

1/1 [=====] - 3s 3s/step - loss: 3320.3005 - val\_loss: 5967.4634

Epoch 25/30

1/1 [=====] - 3s 3s/step - loss: 3117.9573 - val\_loss: 5720.6138

Epoch 26/30

1/1 [=====] - 3s 3s/step - loss: 2770.0369 - val\_loss: 5585.1304

Epoch 27/30

1/1 [=====] - 3s 3s/step - loss: 2552.7100 - val\_loss: 5454.1401

Epoch 28/30

1/1 [=====] - 3s 3s/step - loss: 2510.9912 - val\_loss: 5267.7324

Epoch 29/30

1/1 [=====] - 3s 3s/step - loss: 2344.9021 - val\_loss: 5392.6865

Epoch 30/30

1/1 [=====] - 3s 3s/step - loss: 2278.0603 - val\_loss: 5625.5610



### **Student's and other's contributions to project**

In the course of conducting this final project, it is important to acknowledge the collaborative nature of scientific endeavours. While I, the student, have been actively engaged in various aspects of the project, it is crucial to recognize the contributions and support received from individuals who have played a significant role in its execution.

#### **1. Supervisor's Contributions:**

- Theme and Research Direction: My supervisor provided invaluable guidance in defining the project's theme, aim, rationale, and overall purpose. Their expertise and insights were instrumental in shaping the research's objectives.
- Literature Review: My supervisor actively contributed to the literature review process by offering valuable suggestions, helping to identify key references, and ensuring the coherence of the research context.
- Progress Monitoring: Throughout the project, my supervisor closely monitored the progress, offering continuous feedback and direction adjustments as needed.
- Meeting Coordination: The supervisor facilitated regular meetings to discuss project developments, ensuring effective communication and project management.
- Technical Problem Solving: In instances where technical challenges arose, my supervisor provided essential guidance and support, particularly in resolving computer-based issues and coding difficulties.

#### **2. Module Organiser's Contributions:**

- Foundation Building: The module organizer's assistance was instrumental in building the foundational knowledge and skills required for the project. They provided guidance on how to approach and execute the project effectively.
- Skills Development: Their support focused on enhancing our skills, enabling us to tackle the project's challenges with confidence.

#### **3. External Reference:**

- I referred to an external resource, specifically the following website: [https://github.com/emilio-cartoni/REAL2021\\_starter\\_kit](https://github.com/emilio-cartoni/REAL2021_starter_kit). This external source contributed to my understanding of certain aspects related to the project.