UNIVERSITY
OF SUSSEX

AN EXPLORATION OF TRANSFERRING KNOWLEDGE IN WORLD MODELS

NAME: BENAT C. UNDERWOOD
CANDIDATE NUMBER: 215816
SUPERVISOR: CHRISTOPHER BUCKLEY

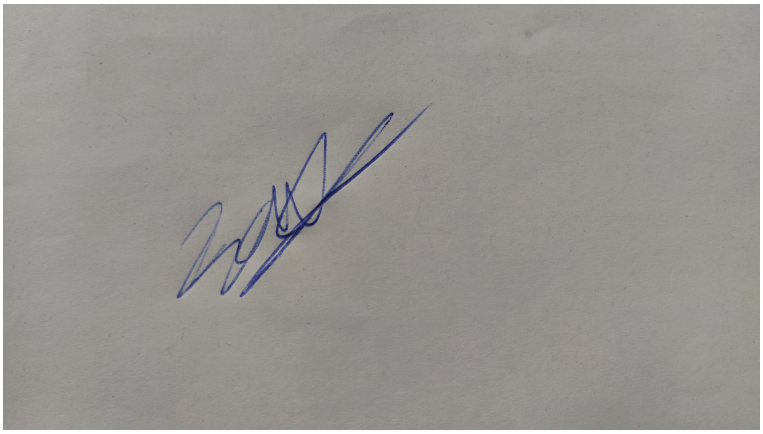
BSC COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

SCHOOL OF INFORMATICS AND ENGINEERING

2022

This report is submitted as part requirement for the degree of Computer Science & Artificial Intelligence at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged. I hereby give permission for a copy of this report to be loaned out to students in future years (delete as necessary).

Signature for statement of originality:



Benat Christopher Underwood Quintana

Acknowledgements

I would like to extend my sincere thanks to Dr Christopher Buckley, for his constant support, advice and guidance, in addition to supervising my final year project.

Summary

This project explores the utility of Transfer Learning (TL) in a model-based Deep Reinforcement Learning (DRL) algorithm (World Models [1]) when performing specified OpenAI tasks [2].

DRL algorithms use large artificial neural networks in order to achieve optimal performance when training an agent to perform a task in an environment. These neural networks are capable of handling data from high dimensional input spaces, which allows the Reinforcement Learning (RL) algorithm to learn a suitable policy for a complicated task, e.g learning from the screen pixel representation of a video game to control the player and 'win' the game.

The main problem with DRL algorithms is that they typically require a vast amount of data and time to train. Consequently they require powerful GPUs and CPUs in order to be practical to use under time constraints. Even with these assets, an AI may take hours or even days to train in order to achieve optimal results. World models is a well-known example of a model based DRL algorithm. This model consists of two neural networks, one to extract the features from the environment (Vision (V) model), and the other to predict the next state of the environment (Memory (M) model) based on the extracted features. It also includes a Controller (C) model , which is in charge of training the agent based on the outputs of both neural networks. These neural networks were trained on a very large dataset of image frames from a video sequence of a video game, and used powerful computer components, to train the networks as fast as possible.

In this project we were interested to see whether we could implement the World Model in such a way as to make it practical to experiment with on the resources available to us. Therefore, our main objective is to reduce the training time, amount of data, and computational requirements to train this model, in addition to achieving a high performing agent. The aim of our project was twofold. Firstly, to update and modify the World Models code, as the libraries to create the model were deprecated, and designed to run on a 64 core CPU, which we don't have access to. We did this as a means to better understand the inner workings of the model, as well as to make it executable on low-end computers. Secondly, we explored whether TL could be used in World Model components to reduce training data and time across OpenAI tasks. TL consists of transferring the knowledge of a pretrained neural network, to the network we have to train, in order to perform the task at hand.

We succeeded in implementing the World Models algorithm so as to make it practical to explore TL across OpenAI tasks on the computing resources available to us. We explored the use of TL in both the V and M components of the World Model when performing the following OpenAI tasks CarRacing-v0 [3], LunarLander-v2 [4] and BipedalWalker-v2 [5]. Our explorations suggest that TL can reduce training time and data required in the V component while achieving similar performance compared to training the model from scratch for the CarRacing-v0 task. However, the performance of the neural network for the V model, showed that when using two different TL methodologies to train the network, it performed slightly worse compared to training from scratch. With the predictive model component of World Models our explorations found no benefits for TL. The performance of the Controller Model, is not directly affected by the performance of the V & M model neural networks and it is able to learn to drive an agent around a virtual racing track, as long as the neural networks return clear observations and predictions of the environment. In addition to the findings above, we observed that the deep learning (DL) technique used in the V model failed to locate the spaceship for the LunarLander-v2 module regardless of whether the neural network employed was trained from scratch or transferred. Our discussion identifies limitations in our explorations and suggests reasons for some of the failures and ways these might be addressed. We also suggest some ways to improve the performance in future experiments.

Contents

Introduction

1.1 Motivation

The main objective of this project, is to modify a model-based Reinforcement Learning (RL) [6] algorithm, in order to run it on any PC without requiring powerful computer components to reduce training time. We intend to do this by rewriting the code, so that the model can be ran on any low-end desktop/laptop computer instead of a Virtual Machine, which is how the original implementation of World Models was designed [7]. In addition to this, we compared the effects of training using Transfer Learning (TL) [8], against training the system from scratch, to demonstrate when can it be used to train more efficiently and achieve similar or better results compared to training the model from scratch, by using less training data and time, thus, reducing computational cost.

1.2 The Problem Area

The most common way to train a robot, or an agent in a video game to complete a specific task is to use one of the three Machine Learning paradigms called RL. Through RL, the agent learns how to solve the task at hand by exploring the world, which allows it to learn new actions, and exploiting it, by using its acquired knowledge of positively rewarding actions to get closer to completing the task.

To deal with the large amount of information in the agent's world, researchers decided to combine RL along with Deep Learning (DL) [9], also known as Deep Reinforcement Learning (DRL) [10], so that the agent is able to process the information in the environment as well as perform actions in it. Unlike simple RL tasks, such as balancing a pole on a cart [11], where the observation space from the environment is an array of the cart's position, cart velocity, pole angle and pole angular velocity, and the action space is pushing the cart right or left, DRL uses artificial neural networks [12] to complete complex tasks, e.g completing a level in a video game. The use of neural networks enables the agent to deal with the high dimensional input from the environment, and learn suitable actions from these inputs.

However, one of the main issues with DRL is that it involves a large quantity of training data, time and computational resources to train the artificial neural network efficiently and quickly [13]. This commonly leads to purchasing expensive computer equipment and using large amounts of energy, which not everyone has access to in the research community.

For this reason, we decided to research the impact of TL [8], on a famous DRL algorithm called World Models [1]. The objective of this investigation, is to reduce the time and amount of data to train the neural networks in the World Models by using TL techniques, in order to prove that the same, or even better performance can be achieved, compared to training the full World Models from scratch.

Our second aim is to simplify the World Models code, in order to be able to execute it on any personal computer. For the original implementation of the World Models code [7], the researchers purchased powerful computer engines from Google Cloud Virtual Machine [14], where they had a access to a powerful GPU and a 64 core CPU to run the World Models code in parallel, as a means to decrease the execution time. We aim to change the code, for the purpose of being able to run it on a any laptop or desktop computer, which not all have multiple core CPUs and powerful GPUs.

For our main aim, we will explore the TL theory on OpenAI gym [2] environments, which were designed to create RL algorithms as they are easy to implement and contain a full range of different tasks. Firstly, we implement the World Model code into the CarRacing-v0 [3] environment, which consists of driving a virtual car around a track in the fastest time possible. This environment was also used in the World Models paper. Additionally, we will be using the LunarLander-v2 [4] environment, where the objective is to land a spaceship on to a platform, and finally the BipedalWalker-v2 [5] environment, where the agent has to learn to walk a robot on an uneven terrain.

1.3 World Models

According to Jay Wright Forrester, the father of system dynamics, humans developed a mental model of the world, as we are not able to perceive all of its information, as a consequence of our limited senses [15]. Explicitly, due to the large amounts of information in our surroundings, the human brain has learnt to filter out the insignificant details from it [16, 17], and create an abstract representation of both the spatial and temporal information of the environment, thus, an internal model of the world.

Research based on this internal model [18, 19], also suggests that humans have a predictive model, where we are able to predict future sensory data given the current state of the environment. Evidence of the predictive model can be observed while playing baseball, where the time to decide where to hit the ball is shorter than the time the visual signals take to reach from our eyes to the brain, therefore, the batter is able to instinctively predict when and where the ball will go [20].

The main concept behind the World Models algorithm is to implement both the mental and predictive model into the agent. By creating an abstract representation of the environment, the agent uses this depiction to predict future states of the world, based on what it has previously seen given the current state of the environment, which aids the agent when performing actions.

As shown in many RL papers [21, 22], agents have been shown to benefit information of past and present states of the environment, to make predictions of the future, they do this by using a Recurrent Neural Network (RNN) [23].

However, in order to create a RNN which is able to learn rich spatial and temporal representations of data, the RNN must be large. This is a problem for RL, as these algorithms are often bottlenecked by the credit assignment problem [24]. In RL problems, typically the reward of the feedback of the simulation, is given at the end, meaning we do not know which actions performed by the agent were positive or negative. The credit assignment problem tries to tackle this problem, by figuring out which actions performed by the agent were positive or negative, hence it keeps track of all of the actions taken. This a problem for RL algorithms, as they must learn millions of weights to keep track of all of the actions, which often takes days to train. Hence, smaller networks are used as they iterate faster and manage to find a good policy during training.

Unlike traditional RL algorithms, World Models model uses a large RNN network to only focus on training the agent's abstract perspective and predictive depictions of the world, in an unsupervised manner. It then uses a smaller controller model to learn to perform the task using the World Models. In other words, the small controller is trained to focus on the credit assignment problem on a small search space, whilst not sacrificing capacity and expressiveness via the larger world model. By training the agent through the lens of its world model, it can learn a highly compact policy to perform the task.

This approach is the first model to be considered to "solve" [25] the CarRacing-v0 task, as it completed the task much faster compared to other implementations. World Models has also demonstrated to be an effective approach when solving tasks from the Arcade Learning environments [26], such as Sonic the Hedgehog [27] and Dreamer-V2 [28].

1.4 Transfer Learning

Briefly explained, TL is the process of acquiring knowledge from one task or domain, and then transferring that knowledge to solve a new similar task [8]. The concept was inspired by the way humans learn new tasks, where we have acquired the innate ability to transfer our skills and knowledge to help us learn new concepts easier [29, 30].

The idea of introducing this into AI, is beneficial, as most engineers are often lacking the amount of data and time when having to design a new system with a different feature space and probability distribution. To be able to transfer knowledge on how to solve a task from one domain to the new one would be desirable, as this would save us time from collecting the training data and rebuilding a model.

A detailed explanation of how TL transfers knowledge from one domain to another in machine learning is demonstrated with the following equations:

Let's consider Source Domain, SD, to be a tuple $\{_{\mathrm{x}}$,P(X)$\}$, where $_{\mathrm{x}}$ represents the feature space and P(X) the marginal probability of the dataset, thus:

$$SD = \{_{\mathrm{x}}, P(X)\}$$

Equation 1.4.1

In addition, we define Source task ST as $\{_{y}, P(Y|X)\}$, where $_{y}$ is the label and P(Y|X) as the conditional probability, therefore:

$$ST = \{_{y}, P(Y \mid X)\}$$

Equation 1.4.2

Using Equation 1.4.1 and Equation 1.4.2, we define TL as a process to improve the outcome of target objective P(Y|X) for target task TT in the target domain DT using knowledge from ST in SD. However, this only works if both domains and tasks, are equal, i.e SD = DT and ST = TT. TL does not work if either SD $\neq$ DT or ST $\neq$ TT .

Within SD $\neq$ DT, if $_{x^{DS}} \neq _{x^{DT}}$, means that there is a feature space mismatch, for example if the task is to perform image classification for dogs and cats, the domain source might be black and white images, while the target domain are RGB images. Moreover, if P(XDS) $\neq$ P(XDT), there is a marginal probability mismatch, meaning that the probability distribution for both datasets are very different, e.g in a spam vs non spam email classifier, DS might be a corporate email dataset, which have less words and often include attached documents, and the DT dataset might be personal emails, which contain more words and rarely include attached files.

Moving to ST $\neq$ TT, when $_{y^{DS}} \neq _{y^{DT}}$, both label datasets are different, for example the Source task had to make predictions for two different classes, whereas the Target task had to make predictions for multiple classes. Finally, if P(YST|XST) $\neq$ P(YDT|XDT) the is a conditional probability mismatch. This can be observed better, by following the example of the spam vs non-spam classification model. Compared to corporate emails, personal emails tend to be more likely to be spam, hence, there is a different probability distribution for both tasks [31].

Additionally, TL also fails when the dataset for the domain target is larger than the domain source. This occurs because the weights of the network haven't been trained enough to transfer knowledge to complete the new task [8, 31].

There are three different methodologies when transferring knowledge from one model to another. Firstly we have feature extraction [32]. The way feature extraction is done, is by removing the final layer of the neural network, which is typically the layer with the role of transforming the features into the objective at hand, for instance classification, and replace it with our own additional fully connected layers or classification layer. The feature extraction layers are then frozen, in other words, the weights are not updated, whereas our own layers are during training.

The second methodology is Fine-Tuning [33]. Similar to feature extraction, Fine-Tuning consists of freezing some of the layers, for instance, only setting the weights to be untrained for the first two layers, while the rest of the layers, pretrained or our own, are updated.

Finally, we have the pretrained model [34], which simply consists of initialising the weights of the new network, with the values of the weights of the pretrained neural network to solve the new task at hand.

Notwithstanding the substantial difficulties involved in implementing TL, we decided to learn more about these and how to overcome them by exploring opportunities for TL in the World Models algorithm for different tasks. TL, has shown to achieve better initial and final performance, in addition to saving training time of the network. These achievements can be observed in a wide variety of applications, such as in healthcare, where researchers wanted to create a pediatric pneumonia diagnosis model [35], which used TL to initialise the models weight parameters by using chests X-ray images, not necessarily linked to pneumonia, and in Natural Language Engineering [36], to train a multi classification model on very different types of text, to more similar texts.

We structured our report by firstly explaining the reason behind our motivation, and describing the structure of our model, in addition to the training techniques we will be using (Section 1). Next, we briefly explain that our project does not require any ethical approval, and that we will be following the BCS Code of Conduct when performing our experiments(Section 2). We then explain how we designed the World Models and implemented TL into the model (Section 3), followed by the results we received when using and not using TL to train the model (Section 4). Finally, we discuss our results (Section 5) and suggest further work which can be done to improve our research (Section 6).

Professional considerations

This project has been aligned with the BCS - The Chartered Institute for IT Code of Conduct* and complies with all sections, which are relevant to any IT project. This project also abides by all specified ethical requirements indicated in the application of the Ethical Compliance Form for Undergraduate and Postgraduate projects attached in the appendix.

*BCS - The Chartered Institute for IT Code of Conduct available at: `https://www.bcs.org/membership/become-a-member/bcs-code-of-conduct/` (last accessed on the 9th of May 2022)

Methods

3.1 Collecting data for World Models

Our training data consists of video sequences from OpenAI gym environments, which have been recorded frame by frame. These images have been collected from 10000 rollouts from the LunarLander-v2, BipedalWalker-v2 and CarRacing-v0 environments, where in each rollout the agent was taking actions based on a random policy. For each rollout, we recorded the first 1000 image frames of the simulation, or until the agent had finished the task (either the agent died or completed the task successfully), along with the actions taken by the agent and overall reward of the simulation. All images were scaled down to the size of 64x64 pixels (see Figure 3.1), to be used as the training data for our World Models.

In the paper, the method used to gather the data was by running 1200 rollouts simultaneously on each core in a 64-core CPU, as a means to collect the data as quickly as possible. Because it was running on a virtual machine (Ubuntu 16.04), the researchers could run the code, without displaying the window, as they were running it on an X virtual frame buffer [37]. They did this, as OpenAI gym contains a memory leak error, which causes a memory overflow, terminating the process early.

However, because we want our implementation to be simple to use and executable on a low end PC, we removed this implementation, by simply executing one rollout at a time, until we got to 10000 rollouts. In order to avoid the code from crashing, we simply closed the window after the simulation had ended, and started a new one.

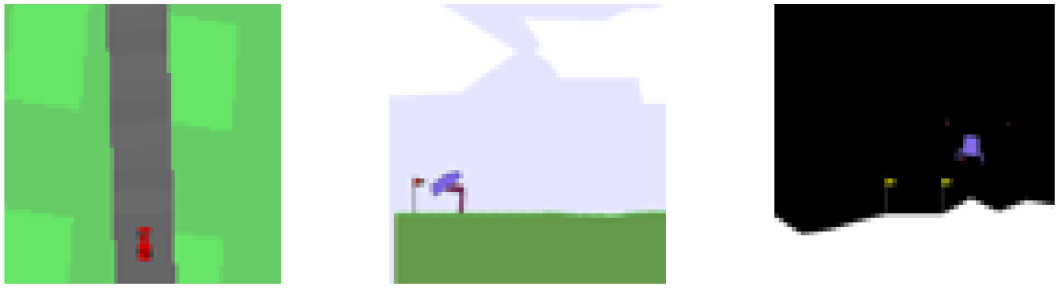


Figure 3.1: CarRacing-v0, BipedalWalker-v2 and LunarLander-v2 environments from OpenAI gym scaled down to 64x64x3 pixel images. The agents performed random actions on the environment, where every action was recorded and used as training data for the World Models.

3.2 Structure of the agent

The structure of our agent consists of the World Model, and a Controller (C) model. As explained in Section 1.1, the World Model is composed of two models, the Vision Model (V), also known as the internal model and the Memory Model (M) also known as the predictive model. The V Model, is in charge of encoding the observation from the environment, where the original observation has been scaled down to a size of 64x64 pixels, into an abstract and compressed representation of it. The role of the M Model, is to predict future states of the environment based on the historical observation from the current state of the world. Finally, the agent has a Controller Model (C), which performs the actions in the environment based on the representations of the world returned by the V and M model. An illustration of how the agent is built can be seen in Figure 3.2.

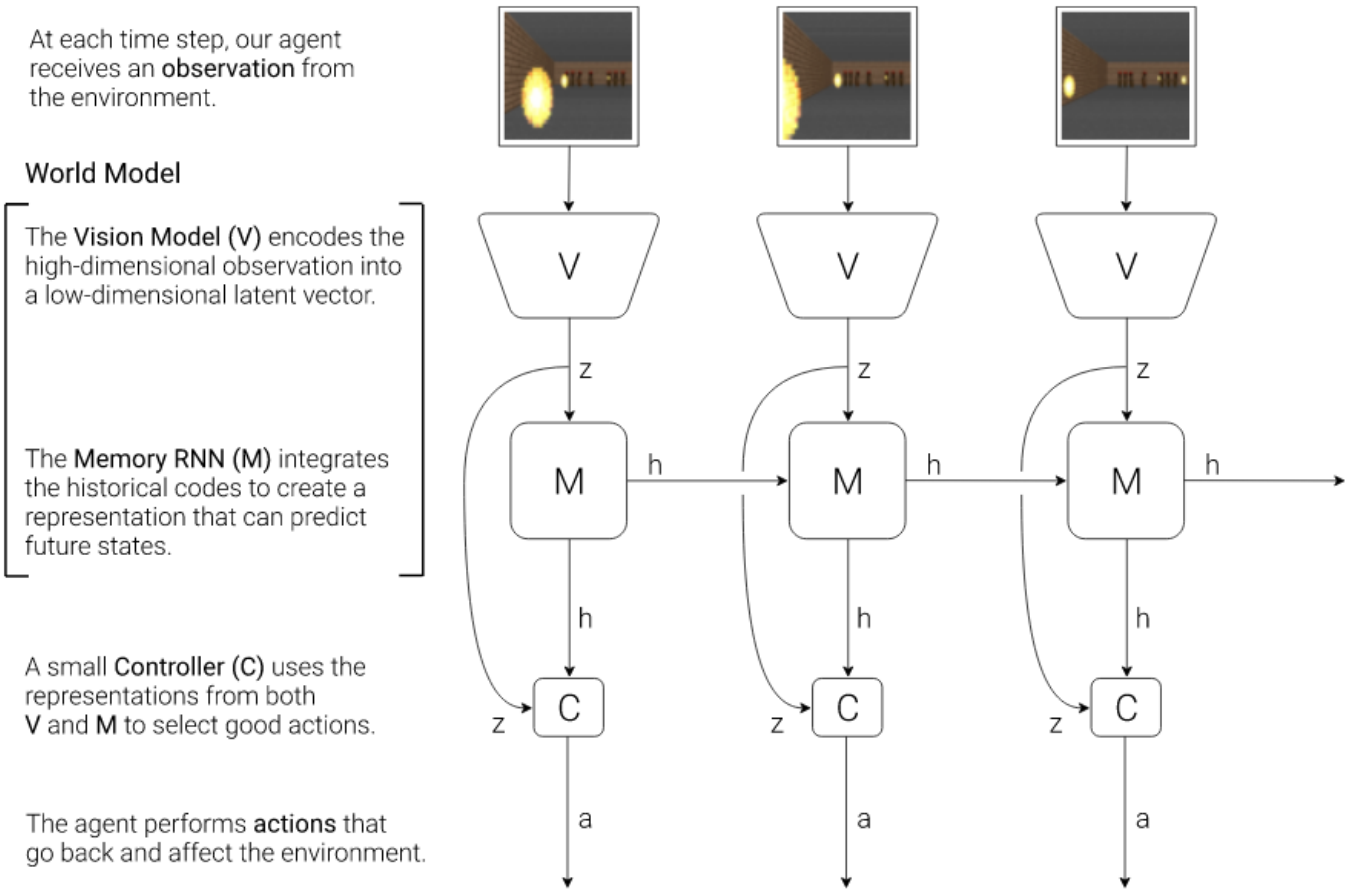


Figure 3.2: (From Ha, D. and Schmidhuber, J., 2018) [1] Structure of the World Model agent. The current observation is encoded by the V model, which is used as input for the M model. The M model then predicts the future state of the environment given the current and historical observations. Finally, the prediction of the M model is concatenated with the present observation, as input for the C model, which performs the actions on the environment.

3.2.1 The Vision Model

The observation given by the environment is a 64x64x3 image frame that is part of a video sequence. This input has 64x64x3=12.288 dimensions, which we want to decrease as this would be too many dimensions for both our M and C model to process. The role of the V model is to reduce this large input size, by learning a dataset of these scaled images from the environment, as a means to create an abstract and compressed representation of the current observation of the world. In other words, the V model transforms the perception of the environment, like the human internal model explained in Section 1.1.

To create this internal model, we first normalise the image frames, by changing their current pixel intensity values into a decimal number between 0 and 1. The next step is to use a Convolutional Variational Autoencoder (CVAE) as our V model (See figure 3.2), to abstract and compress the data. The CVAE, is divided into three components, the encoder, which purpose is to encode each frame it receives at time step t into a 32 dimensional latent vector z_t, the bottleneck which is used to sample the latent vector from a factored Gaussian distribution [38] $N(\mu, \sigma^2)$, where the mean (μ) and diagonal variance(σ^2) comes from the values of the 32 dimension latent vector, and finally a decoder, which is used to reconstruct the latent vector into an abstract representation of the original image (See Figure 3.4).

After setting up the network, we train the CVAE on 10 epochs, a batch size of 100 and a learning rate of 0.001, to minimise the difference between the original image and the reconstructed one. We do this by reducing the CVAE loss, which is calculated by adding the reconstruction loss and Kullback Leibler (KL) Divergence [39].

Once the encoder has been trained, we take all of the sampled μ and σ^2 latent vectors and add it to our dataset, which will be used as the input for our Memory Model.

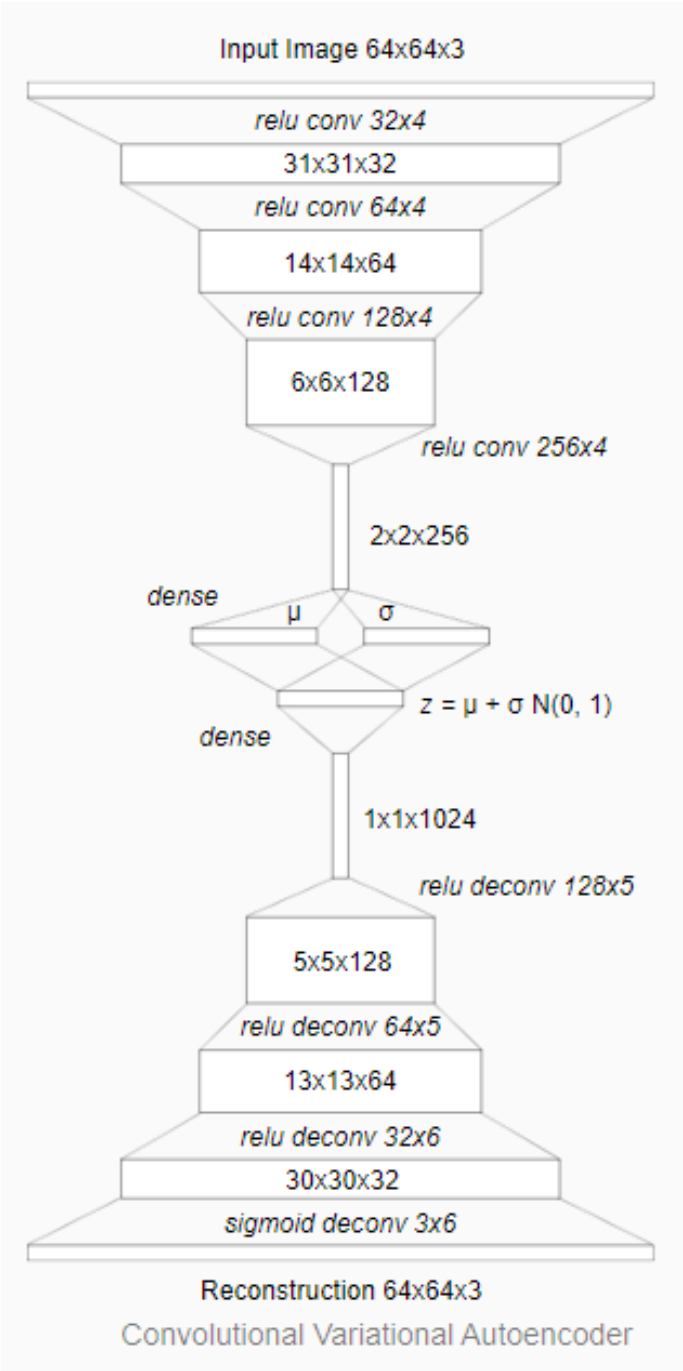


Figure 3.3: (From Ha, D. and Schmidhuber, J., 2018) [1] The CVAE consists of 4 convolutional layers which are part of the encoder, 2 dense layers as out bottleneck, and 4 deconvolutional layers for the decoder. The encoder and bottleneck is in charge of encoding the image into a vector, whereas the decoder is required to visualise the reconstructed image and training the V model.

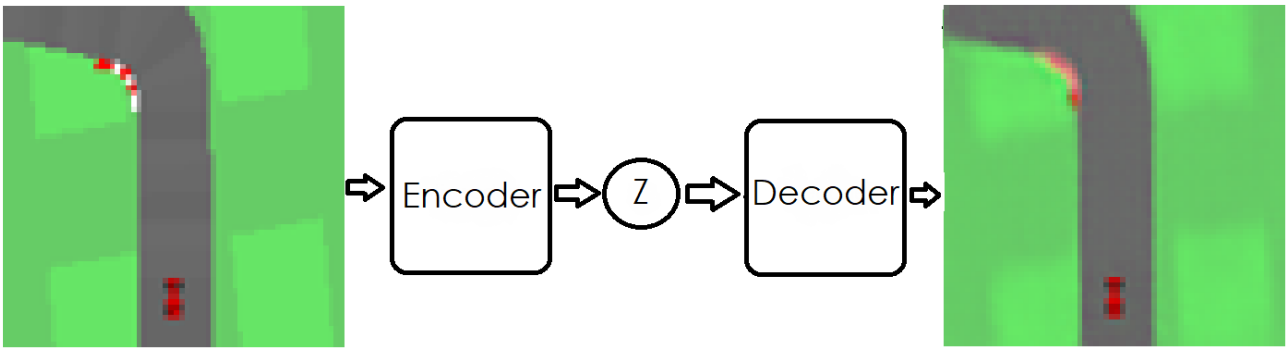


Figure 3.4: 64x64 scaled image from the CarRacing-V0 environment. The environment is encoded into a latent vector for the World Models. This latent vector can then be decoded and reconstructed to observe and abstract representation of the agent's world.

3.2.2 The Memory Model

The Memory model is in charge of predicting the future state z_{t+1} given a sample from a Gaussian distribution $N(\mu, \sigma^2)$ of current state z_t returned by the V model. However, because the environments we are using are stochastic, we train our M model to output a probability density function $P(z_{t+1})$ instead of a deterministic prediction. In order to do this, we use a mixed density network merged with a recurrent neural network (MDN-RNN) [40, 41], which enables us to model the RNN to predict $P(z_{t+1}|a_t, z_t, h_t)$, where a_t is the action taken at current time step t, and h_t is the hidden state of the RNN at t. Then the MDN is in charge of calculating the uncertainty of the RNN prediction, which can be controlled, by adjusting a temperature parameter τ [42]. An illustration of the Memory Model structure can be seen in the figure below.

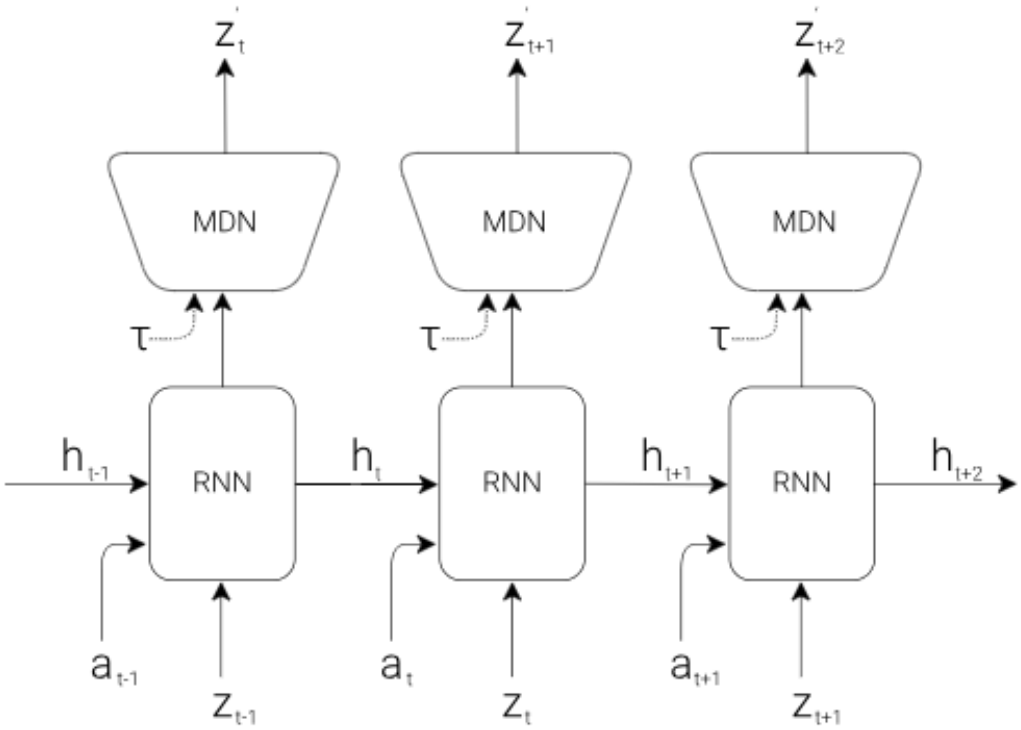


Figure 3.5: (From Ha, D. and Schmidhuber, J., 2018) Structure of MDN-RNN. RNN takes latent vector z_t, a_t and output of previous RNN hidden state h_t to predict z_{t+1}. We can adjust the uncertainty of z_{t+1} by exploring the hyper-parameter τ.

The Memory model is trained on 4000 times steps with a batch size of 100, and a learning rate of 0.001 with decay. The RNN we will be using is a Long-Short term Memory model (LSTM), as these have shown to perform well for time series predictions [43]. It outputs a 256 dimensional vector, which contains the next possible states of the current observation. This vector is then concatenated with the 32 latent vector from the environment, where the values of these vectors will be used as inputs for the C model.

3.2.3 The Controller Model

The C model is responsible for taking the best actions in the environment, in order to acquire as much cumulative reward as possible. It is a simple perceptron [44] and trained separately from the World Model. It uses the output from the z_t vector and hidden state h_t from the M model to perform the action in the environment, as shown by this equation: $a_t = W_c[z_t\ h_t] + b_c$, where W_c is the weight matrix and b_c the bias vector.

The C Model is trained by using an Evolutionary Strategy (ES) [45, 46], in order to optimize the parameters of the model. To evolve the parameters of C, we chose to use the Covariance Matrix Adaption (CMA) [47] algorithm, since it works well with models which have many parameters as input. We will import our CMA for this experiment from the Evolutionary Strategy library [48], in order to simplify our work. Below we show the pseudo code for training the C Model in an OpenAI gym environment, which returns an image frame of the environment as the observation. For our BipedalWalker-V2 and LunarLander-V2 environments, because the observation of the environment is an array, we have to modify the pseudocode below to get the rendered image from the video sequence, and use it as our observation space of the agent.

```
procedure ROLLOUT()
    for generation in generations do
        for individual in population do
            obs ← env.reset()
            h ← rnn.initialState()
            done ← False
            totalReward ← 0
            while done = False do
                z ← vae.encode(obs)
                a ← controller.getAction(z, h)
                obs, reward, done ← env.step(action)
                totalReward += reward
                h ← rnn.forward(a, z, h)
            populationReward[individual] ← totalReward
        controller.setModelParameters(CMA(populationReward))
    return controller
```

Figure 3.6: Pseudocode for training the C model on an OpenAi gym environment. In a generation, we run a rollout for each individual to perform random actions in the environment. After each individual has performed its actions, we get the best performers from the rollouts, and create a new population, based on the best performers of the environment. We repeat this process until there are no more generations.

Our C Model was trained for 20 generations for the BipedalWalker-v2, CarRacing-v0, and LunarLander-v2. For all three tasks we trained on a population size of 32, instead of 64 as done in the experiments in the paper, where each individual would run the simulation twice. The reason we chose a smaller population size compared to the paper, is because similar to section 3.1, the researchers used the 64 core CPU to train the C model in parallel, in order to achieve the results faster. They did this by using a Message Parsing Interface (MPI) package [49] in order to run each simulation in each CPU core. Because one of the aims of this project is to create portable code so it can be ran on any PC, we rebuilt the code for the C model, and trained it on a population of 32, because we would receive our results faster compared to if we trained on a population of 64.

3.3 Transfer Learning

We decided to use TL on both the V and M model, to study whether the agent could still perform the tasks when training the World Models on less training time and data. We will not implement TL on the C model, as this would require to add an additional layer to the linear model, because the CarRacing-V0, LunarLander-V2 and BipedalWalker-V2 environments, all have a different number of outputs. We want to make the training of the C model as simple as possible, so that the model complexity resides in the World Models.

We use two TL methodologies for the World Models. Fine tuning, and using a pretrained model. We won't be using feature extraction, as we won't be adding additional layers to any of our models.

3.3.1 Using a pretrained Convolutional Variational Autoencoder on the Visual Model

For the pretrained model, we decided to first train the V model for a certain OpenAi gym environment from scratch, for example the LunarLander-V2 environment, and use this pretrained model to initialise the weights and biases of the CVAE, to reconstruct the observation of the environment we want to solve, for instance CarRacing-V0. When performing this experiment we will train the CVAE on half of the dataset, 5000 rollouts, as we want to demonstrate that we can achieve better results and save time when training from a pretrained model, compared to training from the start.

3.3.2 Fine-Tuning the Visual Model

When fine-tuning the V model, we initialised the weight of the network, in the same way we did for section 3.3.1. However, instead of retraining the full pretrained network, we will freeze, also known as setting the weights to be untrained, some of the layers of the CVAE to observe whether this will perform better or worse, compared to training the network from scratch. For our experiments in the CVAE, we will first freeze the full encoder and train it for 5000 rollouts, and then freeze the first two layers of the

encoder, and again train it on 5000 rollouts, so we can compare the results against training the CVAE from scratch and using a pretrained model. An example of Fine Tuning can be observed in Figure 3.7.

Moreover, we tested that the weights of the encoder of our model have been frozen, when we look at the summary of the model layers. The summary of the model, for both, when we freeze only two layers, and when we freeze the full encoder, can be observed in Figure 3.8 and Figure 3.9.

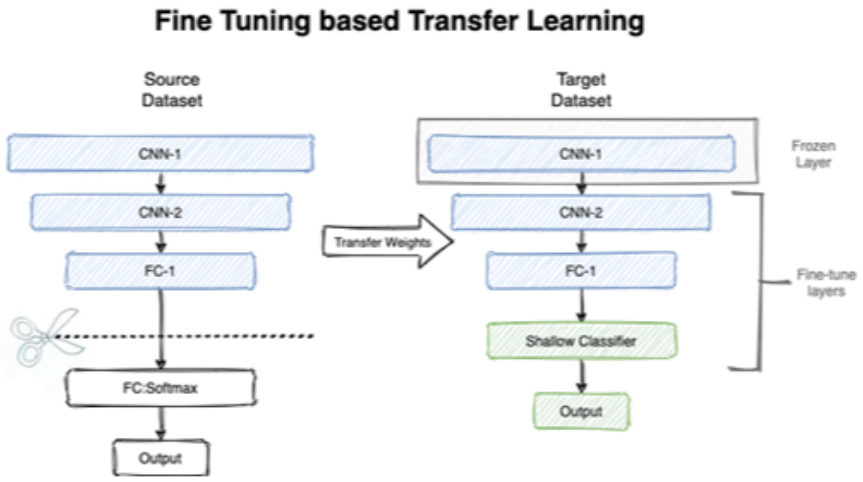


Figure 3.7: (From Sarkar D, Bali R. and Ghost T, 2018) [31] An example of how Fine Tuning is used in a Convolutional Network. The first Convolutional layer is set to be untrained throughout the training of the network, whereas the rest of the layers weight's and biases' are updated.

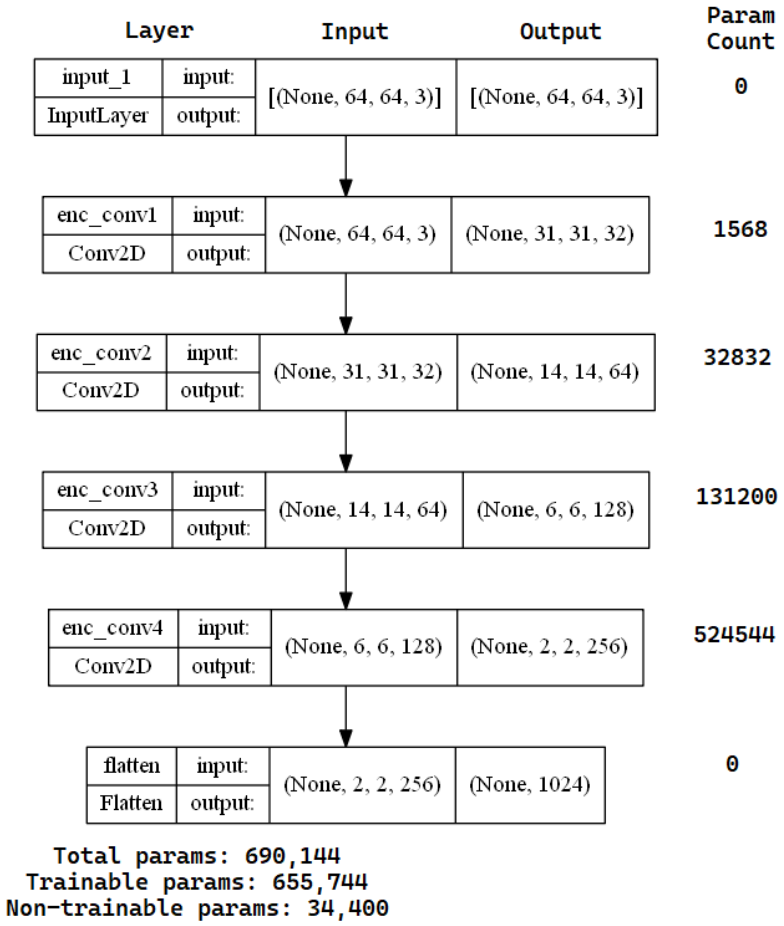


Figure 3.8: Freezing 2 layers of the encoder. We can clearly observe that only the two first layers have been frozen, because if we add up their parameters 1.568+32.832=34.400 parameters to be untrained.

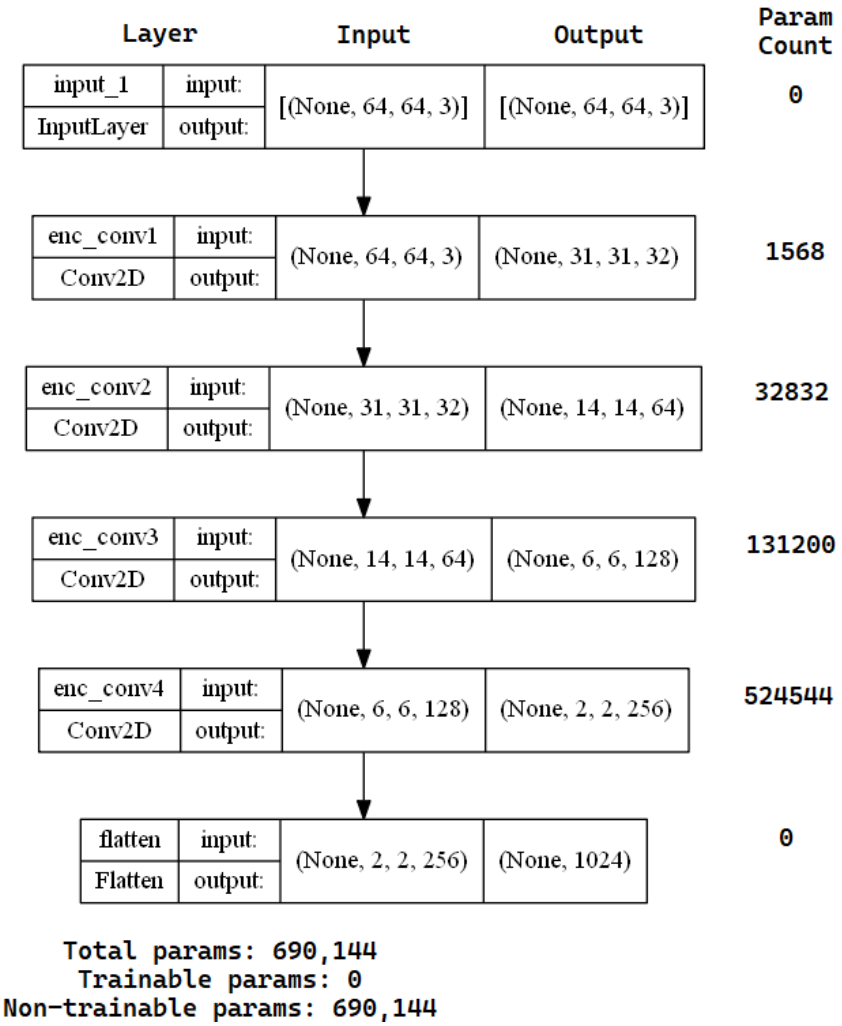


Figure 3.9: Freezing the full encoder. We can clearly see from the figure that all layers have been frozen, as there are not parameters to be trained.

3.3.3 Transfer Learning on the M model

For the Memory model, unlike the Vision Model, to perform TL we have to remove the Input Layer of the Long-Short Term Memory (LSTM) network, as the input shape is different for different environments. The reason this is different, is because apart from inputting the vectors from the V model, the M model alos takes the action space of the environment as input, which is different in each OpenAI gym environment. To solve this problem, we replaced the input layer of the environment with the correct shape for the new environment we want to perform the task on. Because the number of weights for both, the pretrained model and the new model are different, we initialise the weights and biases of the input layer to have a random values

For simplicity's sake, the mean (μ) and diagonal variance(σ^2) from the V model, will be from the V model trained from scratch, where it has been trained on 10000 rollouts and 10 epochs.

3.3.4 Using a pretrained Mixture Density Recurrent Neural Network on the Memory Model

Similarly to the Visual Model, we first trained our M model for the BipedalWalker-v2 environment, and transferred the weights and biases into our new M model. However, instead of reducing the training data by half, we will train the model on 2000 timesteps, instead of 4000.

3.3.5 Fine tuning the Memory Model

When fine tuning the Memory Model, we first froze the LSTM layer and left the MDN layer unfrozen, and vice versa. The training timesteps was also halved for this part of the experiment.

We also test that our Long-short term memory model and our Mixed Density Network have been frozen, by looking at the model description, which is is llustrated in Figure 3.10 and 3.11.

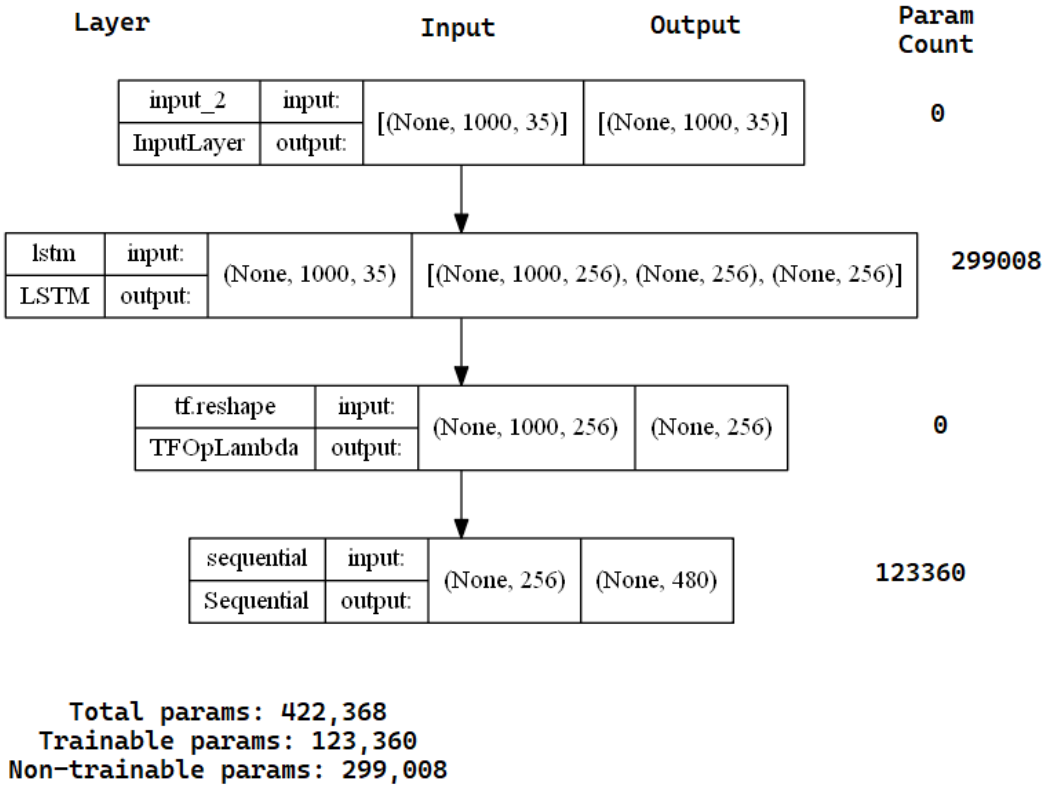


Figure 3.10: Freezing the Long-Short term Memory model. We can observe that all of the parameters for the LSTM have been set to be untrained during training, as the LSTM model has 299008 parameters which are set to be untrained.

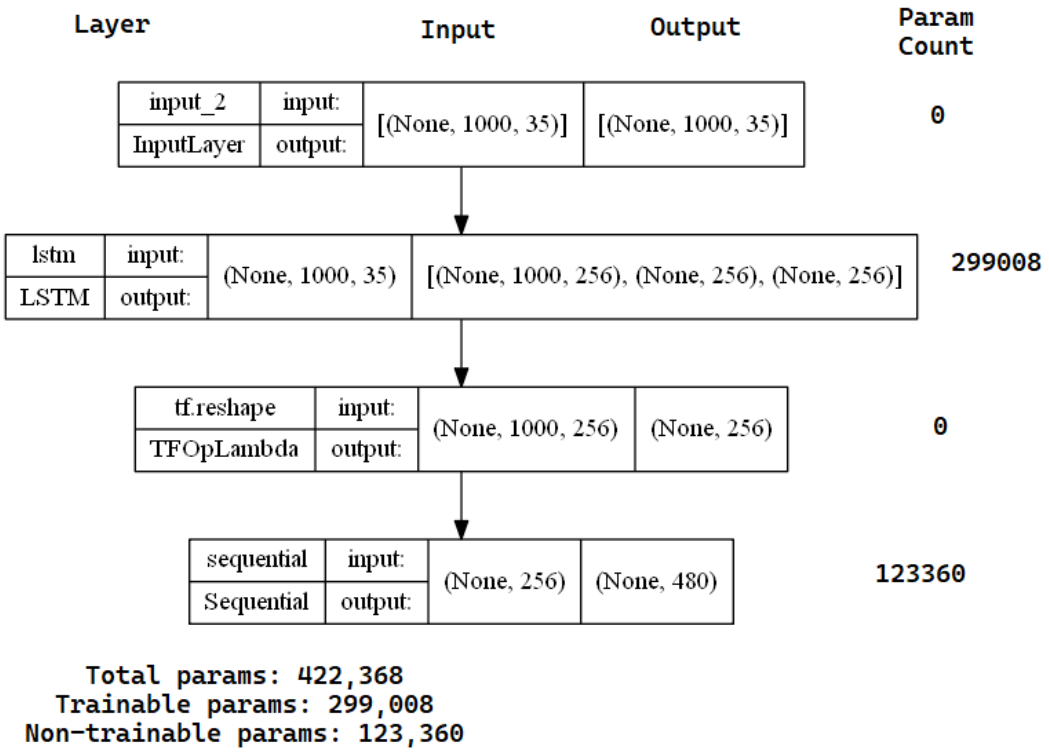


Figure 3.11: Freezing the Mixture Density Network. We can see that the MDN will not be retrained when we run the Memory Model. We can clearly observe that 123360 parameters will not be updated when the M model is training.

Results

In this section, we present the results of our investigations. First, we demonstrate that our V and M model work as expected when using different TL techniques, by plotting the loss for every epoch, we can see that it is gradually decreasing. In addition to this, we show the reconstructed image frame for the CarRacing-v0 task from a V model, which had been trained on scratch and different TL techniques. Then, we show the performance of our C model in different OpenAi environments. We illustrate the best performing individual at the task, as well as the performance of the mean population and worst performing agent when using different training techniques. We averaged the results for the performance of the population over 5 consecutive runs, where we plot the performance mean and standard deviation for each generation.

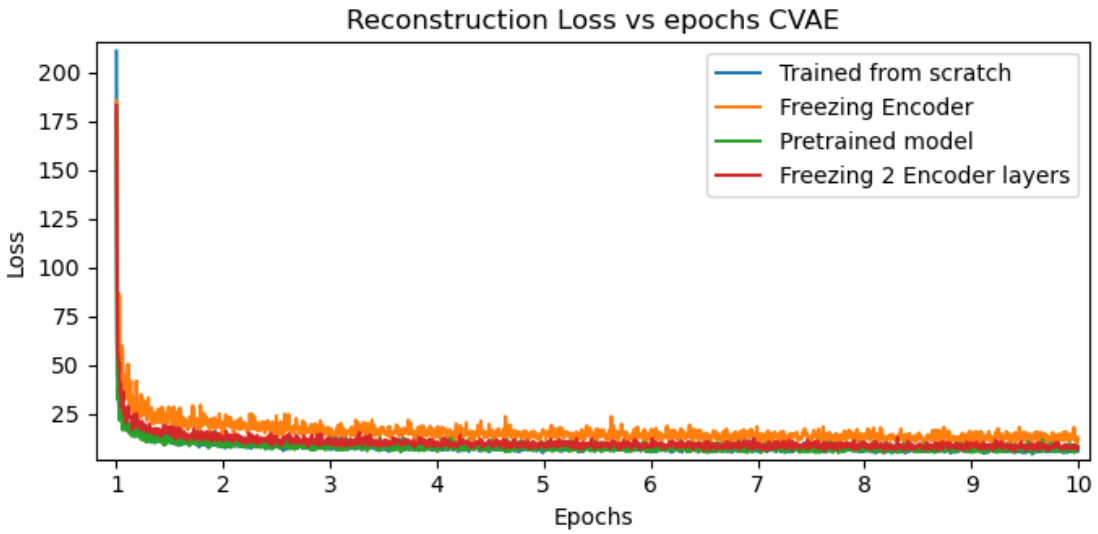
4.1 Vision Model results

The purpose of the V Model, is to encode the high dimensionality observation space of the environment, into a latent vector. This abstraction of the environment has to be as clear as possible, so that the C model can perform accurately. The V model was trained to learn image frames from the CarRacing-v0 task. We compared its performance training it from scratch against training it using different TL techniques, from a V model which had previously been trained to learn image frames from the BipedalWalker-V2 task.

We observe in Figure 4.1 and 4.2, when freezing all of the layers of the encoder in the CVAE, even if its starts at a lower reconstruction loss it performs worse when reducing its loss overtime, compared to the other TL methodologies and training from scratch. Nonetheless, when only freezing the first two layers of the encoder, we achieve better results, however it doesn't perform as well as compared when training from scratch. With the pretrained model however, not only do we start at a lower loss, but we also achieve a very similar loss rate compared to training the V model from scratch.

In terms of the KL divergence, we observe that the training method which minimises the loss the most, is when we train the model from scratch, however it is not much difference to the loss rate when using a pretrained model. Finally, we can observe how well the training techniques reconstruct the environment, by observing Figure 4.3.

a)



b)

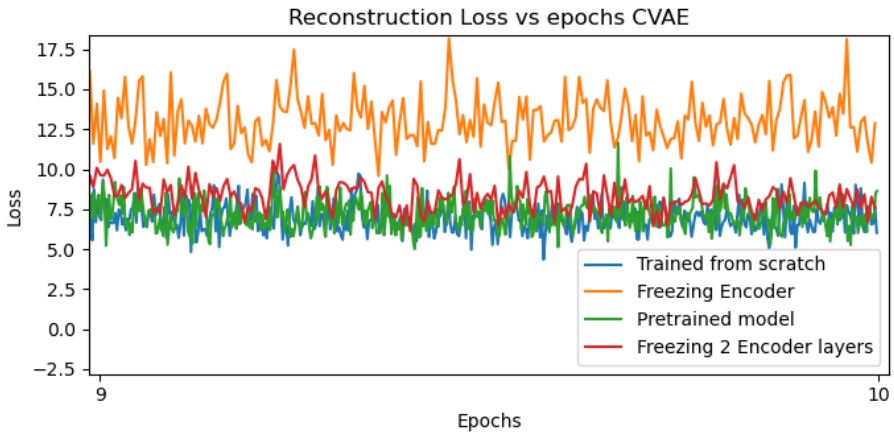
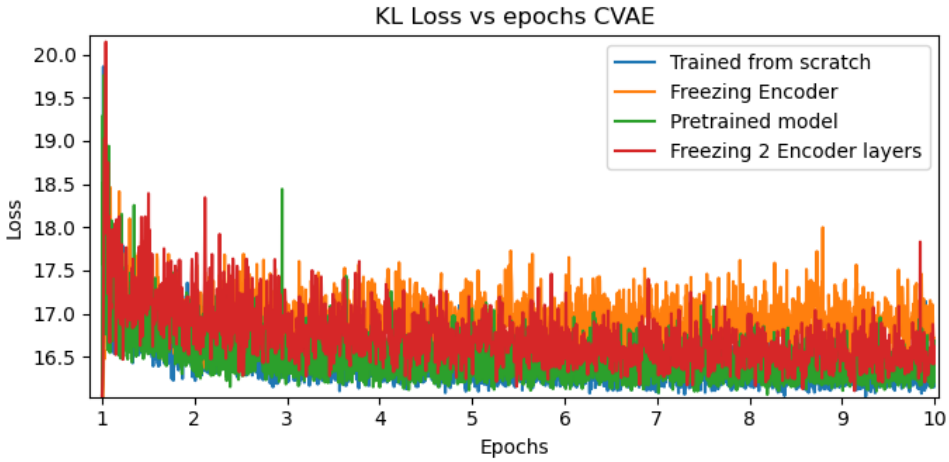


Figure 4.1: CVAE reconstruction loss across 10 epochs (**a**), when trained from scratch and different TL methodologies. We clearly observe that the loss across epochs for both trained from scratch and using a pretrained model is the same. When freezing 2 layers from the encoder, there isn't much differnce either compared to the former 2, however, when we decide to freeze the full encoder, we achieve the worst results. We get a clearer view for the differences in TL and training from scratch, when we zoom into the graph on the 9^{th} and 10^{th} epoch (**b**).

a)



b)

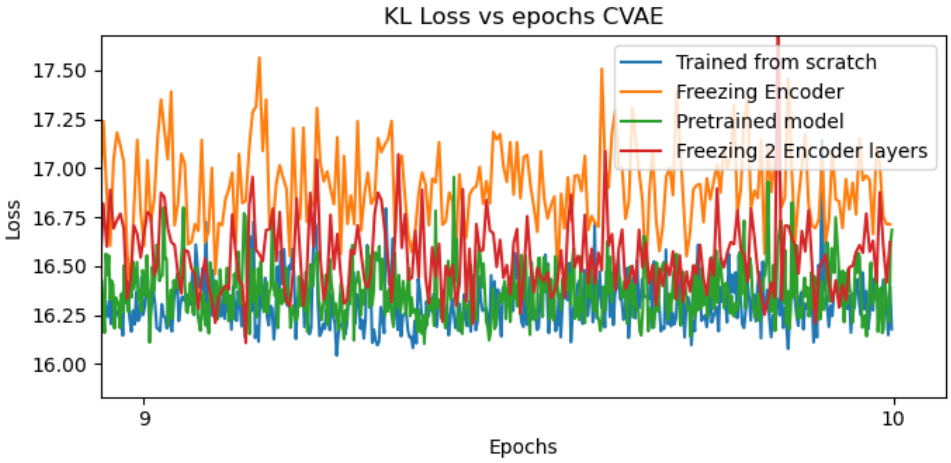
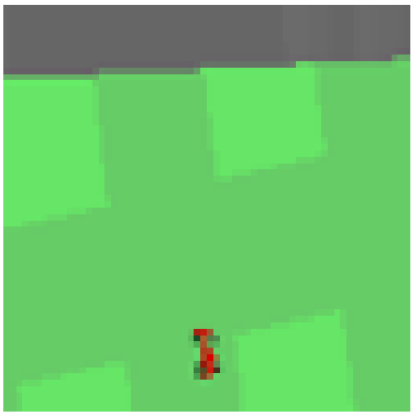
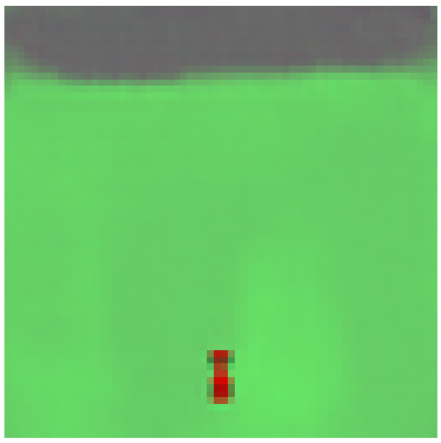


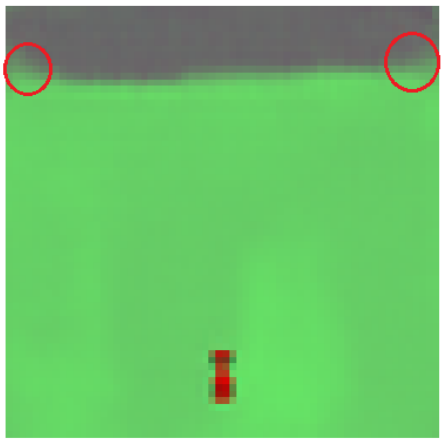
Figure 4.2: CVAE Kullback–Leibler divergence across 10 epochs (**a**), when trained from the beginning and TL methodologies. Whilst the model trained from scratch achieves the lowest loss rate, the pretrained model is not far behind. When we decide to freeze parts or the full encoder however, there is more divergence when trying to reconstruct the image. We can clearly observe this when we observed the rate between the 9^{th} and 10^{th} epoch (**b**).



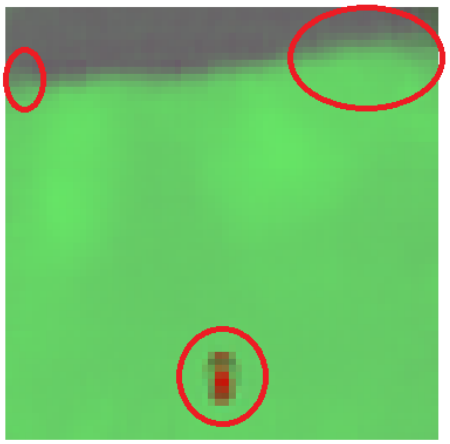
Original image frame



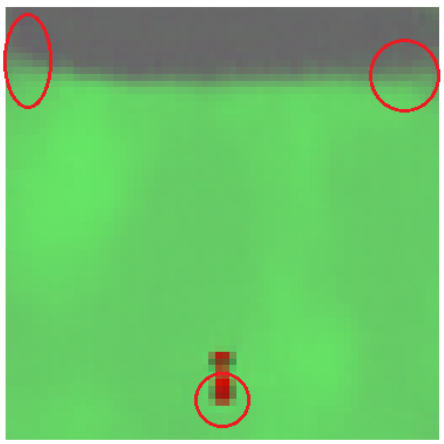
Trained from scratch



Pretrained model



Freezing encoder



Freezing 2 layers in the encoder

Figure 4.3: CVAE reconstruction of the environment with different training techniques. None of them are able to reconstruct the image perfectly, as clearly seen from the original image frame, the car is slightly positioned at a different orientation compared to the reconstructed images, however, we can observe that we get the same reconstruction from the environment when we train the model from scratch and using a pretrained model.

We achieve similar loss graphs when training the model from scratch and using different TL techniques for the BipedalWalker-V2 and LunarLander-V2 environments. However, when visualising the reconstruction image for the LunarLander-V2 task we observed that the CVAE is not able to locate the spaceship when creating an abstract representation of the environment when trained on scratch, nor trained using different TL techniques (See Figure 4.4).

Original image frame Scratch

Figure 4.4: Original image from the LunarLander-v2 environment, and the reconstruction from the CVAE. As we can clearly observe, the CVAE has left out the spaceship, when reconstructing the image, no matter which TL methodology we used nor training from scratch. Because the V model is not able to rebuild the spaceship, the World Models won't be able to complete this task, as it will not be able to locate the position of it.

Apart from visualising the loss rate when we use and do not use TL, Figure 4.5 illustrates the differences in training time across ten epochs, for different types of training. We ran this model on an NVDIA GTX 1060 Super GPU [50].

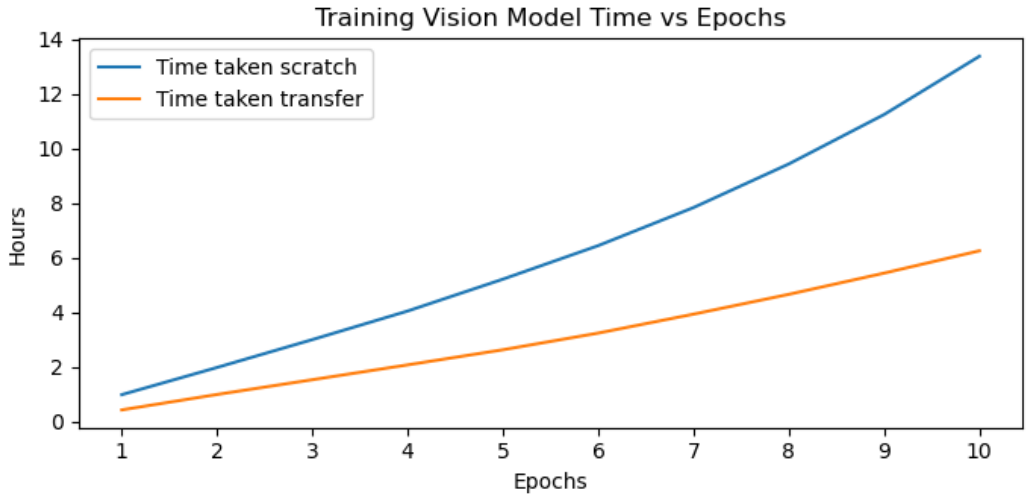


Figure 4.5: Training the V model when using different training techniques across ten epochs and hours. We can clearly see that when training using TL, the model learns twice as fast compared to training from scratch.

4.2 Memory Model results

The purpose of the M model, is similar to the V Model, except that it has to create a clear future representation of the environment, based on the current observation.

For the M model, we again transferred the knowledge from a trained M model on the BipedalWalker-V2, to the task we want to perform, in this case the CarRacing-V0 task.

As we can observe from Figure 4.6, when we train the model from scratch it quickly decreases its loss to almost 0. When using TL methodologies however, even if the pretrained model gradually gets closer to 0 loss, it never does as well as from scratch and doesn't start at a lower rate. We also observe, that if we freeze the layers, our loss seems to get stuck at a local minima prior to reaching 2000 timesteps.

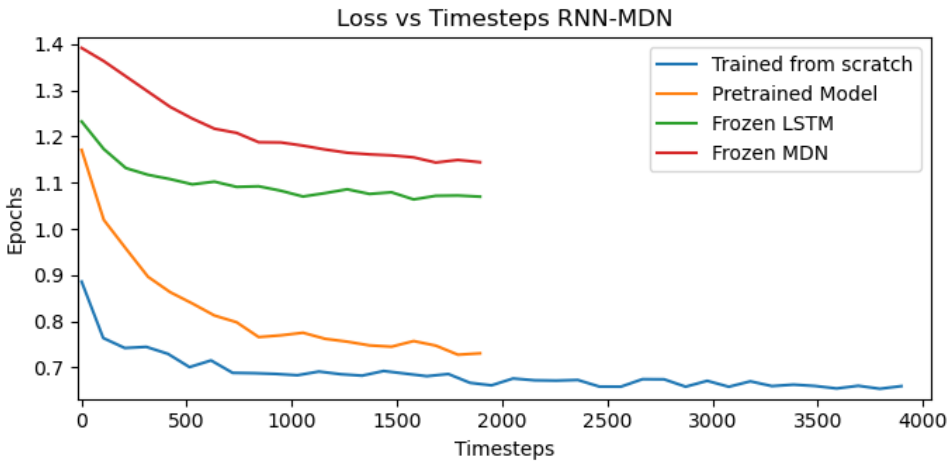


Figure 4.6: Loss vs time steps for RNN-MDN, where we trained the model on 10k images, and halved the time steps for the models which were trained using TL. We can clearly observe that the best training method for the M model, is training the whole model from scratch, as it is the closest to approximate 0 loss. When we use different TL methodologies however, we see that the M model starts to converge on the 2000th, meaning it is unable to reduce any more loss.

4.3 Agent Performance

We tested our C model on all environments to observe if TL did or did not improve the performance of the agents. In other words, we wanted to observe whether TL could achieve more cumulative reward in shorter time compared to training from scratch on different OpenAI task.

Firstly, we decided to observe the CarRacing-v0 agent performance. We can observe that the mean population is slowly improving, when trained from scratch as can be observed from Figure 4.7. We didn't leave it training for longer, as this would require days, in order to achieve optimal results.

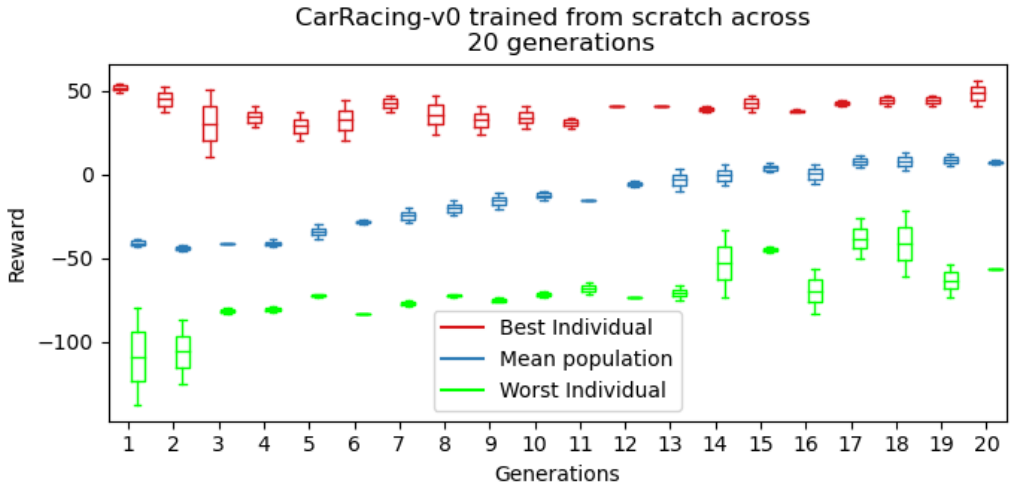


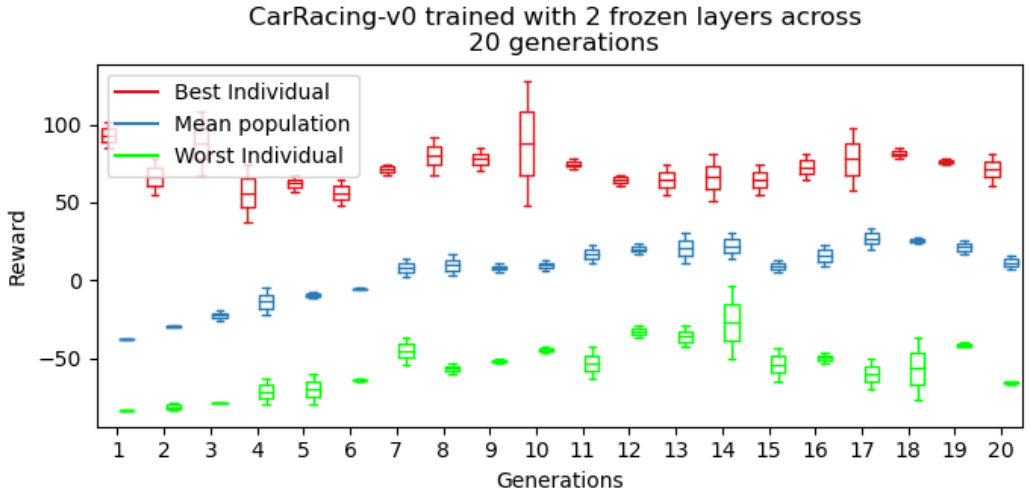
Figure 4.7: CarRacing-v0 performance when training the World Models from scratch on a single run. We decided to plot the best individual, mean population and worst individual from the CMA Evolutionary Strategy. We can clearly observe that the performance of the population is slowly improving.

However, this also seems to be true when training the model on TL when freezing the full encoder and two layers of the encoder in the CVAE, as seen in Figure 4.8.

Moreover, when we compare the performance of the mean population for each training technique, averaged on 5 runs, we observe that when trained the V model on TL methodologies, we achieved similar results compared to training the C model from scratch, see Figure 4.9.

We also tested the performance on the BipedalWalker-v2 task and LunarLander-v2 task, having trained the V model and M model from the beginning and using TL techniques on the V model, from a pretrained V model to learn images from the CarRacing-v0 task (See figures 4.11, 4.12, 4.13). Nonetheless, our results show that across 20 generations, the walker for the task has not shown to improve its performance, nor has the LunarLander-v2. As shown in Figure 4.15 and 4.16, the performance rate for the mean population does not improve, and the performance is quite random for each epoch.

a)



b)

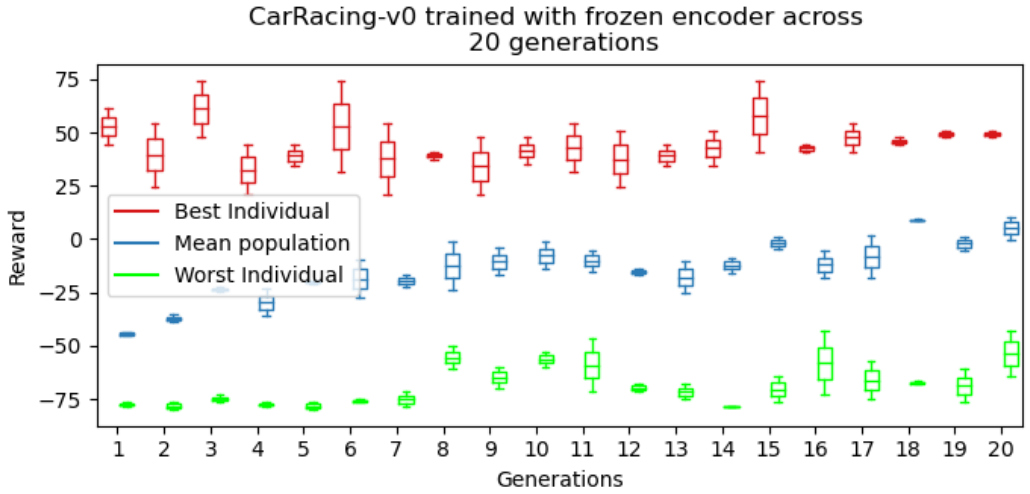


Figure 4.8: CarRacing-v0 performance when training the World Models from freezing 2 layers from the encoder (**a**) and freezing the full encoder (**b**) on a single run. We decided to plot the best individual, mean population and worst individual from the CMA Evolutionary Strategy. We can clearly observe that for both TL techniques the mean population, best and worst individual seem to be improving their performance.

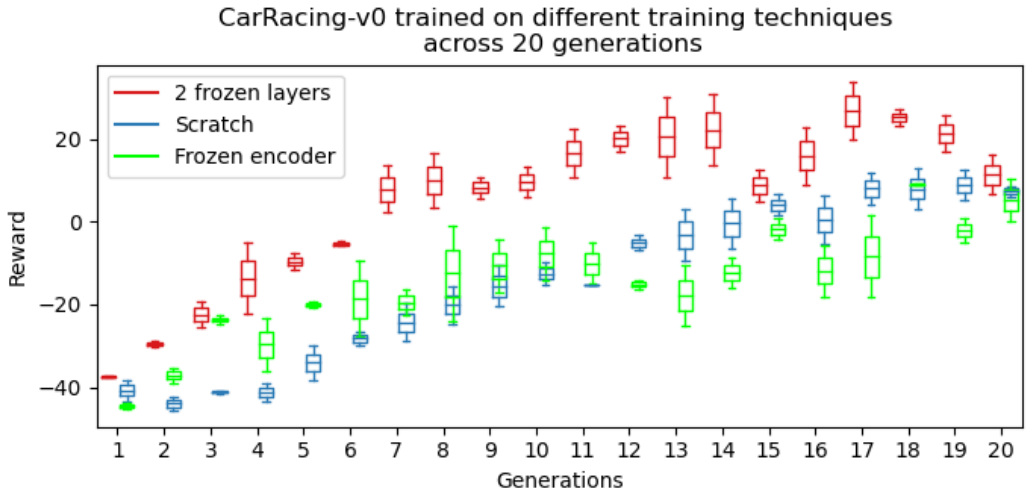
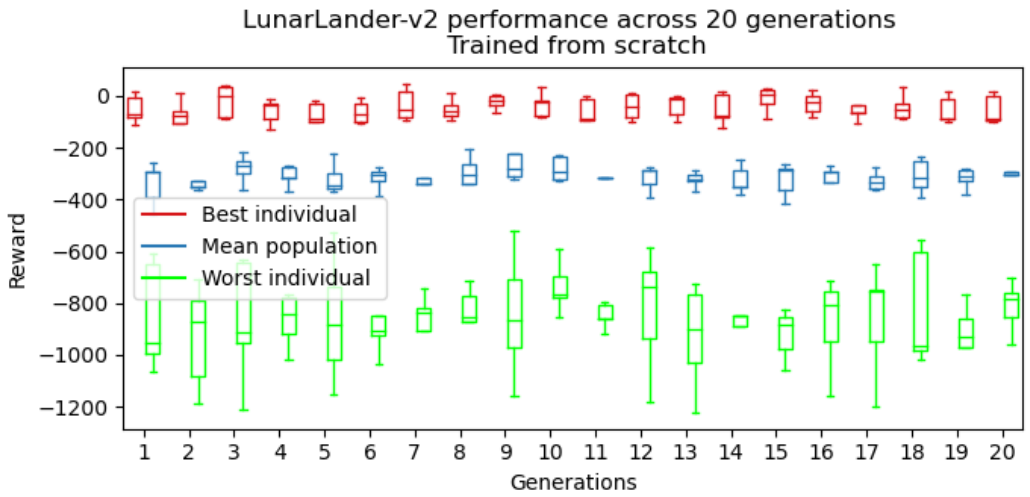


Figure 4.9: Comparison on the the reward achieved across 10 generations for the agents in the CarRacing-v0 task. We can clearly observer that no matter which training method we use, the agent seems to be performing similarly on each generation.

a)



b)

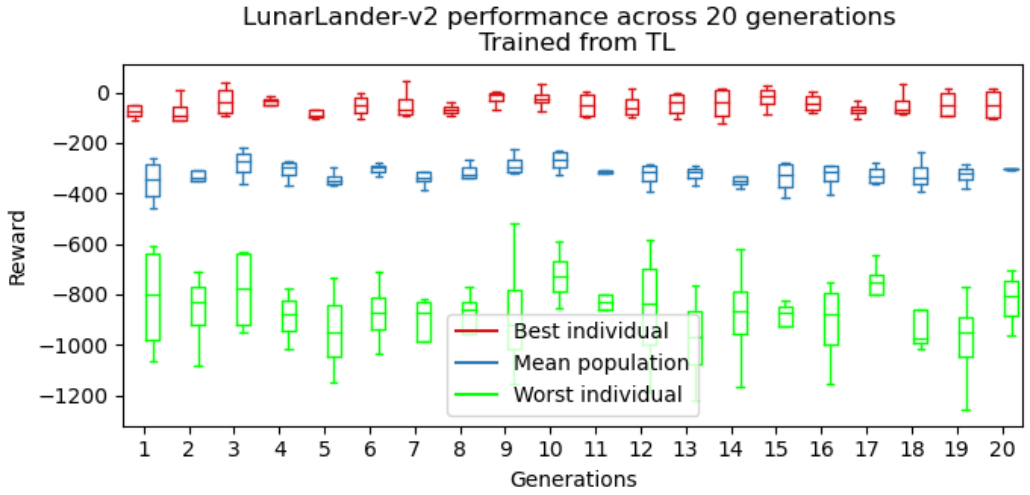
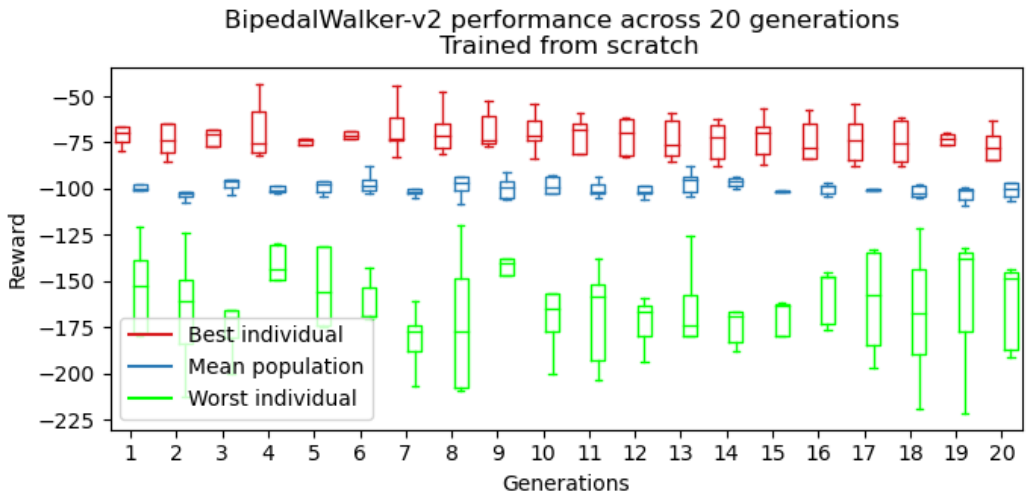


Figure 4.10: LunarLander-v2 performance across 20 generation. We plot the performance of the best individual, mean population and worst individual from the CMA ES, trained from scratch (**a**) and using TL (**b**).

a)



b)

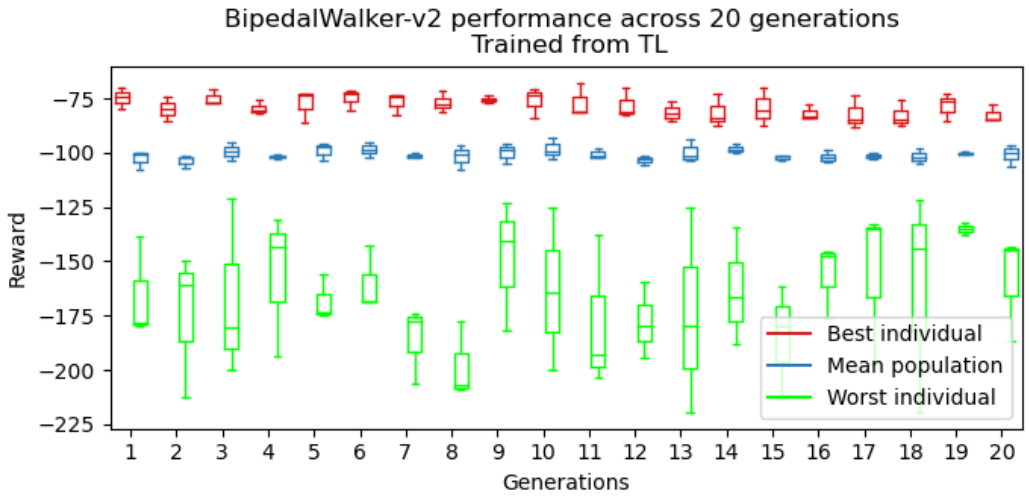


Figure 4.11: BipedalWalker-v2 performance across 20 generation. We plot the performance of the best individual, mean population and worst individual from the CMA ES, trained from scratch (**a**) and using TL (**b**).

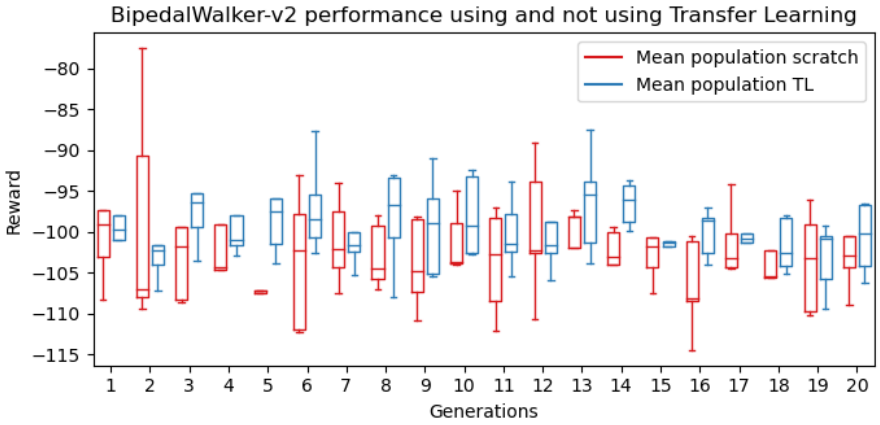


Figure 4.12: BipedalWalker-v2 performance across 20 generation. We can clearly observe that the mean population for both training techniques is very random for each generation, meaning that the agent is not able to learn a suitable policy to solve this task.

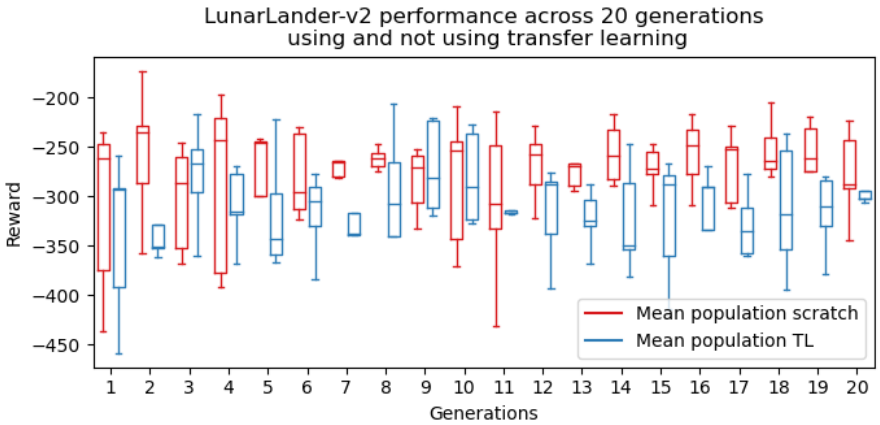


Figure 4.13: LunarLander-v2 performance across 20 generation. We can clearly observe that the mean population for both training techniques does not seem to either increase or decrease across 20 generations. We believe that the agent is not capable of learning a suitable policy to complete this task.

Discussion

5.1 Overview

This project consisted of two objectives. Firstly, to rewrite the World Models code, to make it simpler and executable on any computer, as well as adding additional OpenAI gym environments to complete new tasks. This objective took far more time than expected, as most of the libraries to create the neural networks were deprecated, and the code for the C Model had to be changed to a great extent, as it was originally designed to run on each CPU core in parallel. However, by rewriting the code it enabled us to fix various bugs, as well as improving the functionality of the model. Secondly, our main objective, was to study the effects of different TL techniques when training the World Models algorithm. We aimed to train the World Models to perform different OpenAI gym task, and compare their performances when training on less data and time using different TL techniques against training from scratch with all of the data and more timesteps.

We proved that by using a pretrained model on the V model, without fine-tuning the layers, to learn the image frames for a new task, performed as well compared to training the model from scratch. However, this was demonstrated to be false in the case of the M Model, where training it from scratch achieved the best performance, as it was the only training method to gradually decrease its loss to almost 0 units.

In terms of the agent's performance, we only decided to test the performance of TL on the V model, as when we used TL on the M model, it did not appear to be beneficial for the C model. We observe that for the CarRacing-v0 task, the cumulative reward improves for the mean population, worst and best individual across 20 generations, no matter which training method we use. Having trained the V model through TL, the performance of the C model seemed to be similar compared to training the V model from scratch. This suggests that as long as there is an abstract representation of the environment, no matter how blurry the reconstruction is, it is still able to perform the task effectively. However, in the case of the Bipedal Walker-v2 and LunarLander-v2, we see that its performance does not improve over time no matter which training method we used, this suggests that the agent is not able to learn a suitable policy to solve any of these tasks, using World Models.

Apart from exploring the effects of TL, we also discovered that the V Model is not able to predict the coordinates of the spaceship for the LunarLander-v2 environment, as when it reconstructs the image for this environment it seems to deem the spaceship as an unimportant feature, thus it does not include when decoding the $_z$ vector.

5.2 Agent limitations

We believe that one of the reason the World Models is not able to land the spaceship for the LunarLander-v2 task, nor get the robot walking for the BipedalWalker-v2 is due to how these models were originally designed. Originally, these models return the observation of the environment as an array of the agent's coordinates and velocity, which obviously we cannot use as input for the World Models. That is why, we had to modify the code, in order to return us the image frame of the environment, use this as the observation space of the world, and encode it for our C model. Fortunately, this worked to train the World Models on these OpenAI environments, nevertheless the C model was not able to find rewarding actions and learn from them, as a means to get closer to completing the task.

In addition to this, for the CarRacing-v0 task, we had to tweak some outputs, in order for our agent to perform better actions, e.g if the linear model predicted the car to come to a full stop, instead of allowing the agent to do so, we programmed it to decrease its speed. The same is done in Dylan Djian's work [27], where the author designed the World Models code to only output the two most useful actions, the ones which could get the agent closer to solving the task, and only perform the action, if it was above a certain threshold. We weren't able to find these heuristics for our two environments, specially

the BipedalWalker-v2, where we had to figure out a way to keep the agent balanced when moving one of its legs.

5.3 World Models limitations

5.3.1 Vision Model

As demonstrated in Figure 4.4, when given the task to reconstruct the LunarLander-v2 environment, the V Model always seems to leave out the spaceship, no matter which training method we use. We believe that the reason for this is because the location of the spaceship is so random, that the CVAE is not able to reconstruct the spaceship in the same position as in the input image frame, as it has learnt from images where the spaceship was at very different locations in the environment. The CVAE learns from 10000 rollouts where the spaceship is in a different location every time, both inside and outside the screen, and unlike the CarRacing-v0 and BipedalWalker-v2 environment, where the camera is always focused on the position of the agent, the view we have of the world, is of a fixed position in the environment. According to Philgren, Sandin and Liwicki (2020) [51], the CVAE are trained on element-wise loss, which disregards high level structures from the image, and doesn't include them when encoding the image. We can clearly see this occurring in the LunarLander environment, because it appears randomly in different location, the CVAE disregards this as an important feature.

In the World Models paper, they also explain this problem when training the V model on the VizDoom [52] environment. Because it was trained in an unsupervised manner, it does not know which features will be useful, to complete the task at hand. For instance, for the VizDoom environment the V model returned detailed brick tile patterns, which are unimportant when solving the task. This was opposite to what happened in our experiment, where the V model left out the most important feature of the task, nevertheless, it was able to reconstruct the mountainous terrain outside of the landing platform perfectly.

5.4 Transfer Learning Limitations

As clearly shown in Figure 4.5, TL does not work as well as compared to training the M model from scratch. On reflection, we think this is probably because of the marginal probability mismatch, between the source task and the target task, as explained in section 1.2. This occurs in the M model, as part of its input is the action space of each environment, where each OpenAI gym task has a different number of action spaces. For example the CarRacing-v0 task has three possible actions, accelerating, steering left or right and stopping, whereas the LunarLander-v2 environment only has two actions, the force of the engine and steering the spaceship left or right. Because of this, the probability distribution for the outputs of both tasks are different, hence, TL does not work well for these kind of tasks.

Conclusion & Further Work

Our research illustrates that whilst TL can save us time and resources when training a model-based RL algorithm, its best to set only a few layers from the pretrained model to be untrained, or none, as most weights must be adjusted to the task we want to perform at hand. Even though, for the CarRacing-v0 environment, we acquire the same results compared to when we train the network from scratch on a rollout of the environment, we suggest only using TL when we have limited data and time, which was not the case for our experiments, as we could collect data easily by using a random policy on an agent and recording the performed actions.

In terms of the World Models, we find it to perform accurately when training on environments which return image frames as the observation output. However, we believe there can be an improvement to the V Model, which we suggest in section 6.1.

6.1 Improving the Vision Model

For tasks where the object of interest is not tracked by the camera, and consequently mainly centered in the terrain of the environment, for instance the LunarLander-v2 environment. We suggest modifying the CVAE from the V Model to be trained with perceptual loss. As shown by Philgren, Sandin and Liwicki (2020) [51], researchers used this technique to reconstruct the LunarLander-v2 environment, which they succeeded by just using 1400 images from the environment, where the spaceship was inside the image frame.

Another approach to improve the V Model, is instead of encoding the image into a latent vector by using a CVAE, to use a Generative Adversarial Network (GAN) combined with an encoder to convert the image into a latent vector. These techniques are more novel, and better at encoding the image into a latent vector as shown in investigations, such as the Bidirectional Generative Adversarial Networks (BiGANs) [53] and the VAE-GAN [54].

6.2 Using different environments

Because the only task which returns us an image frame environment from OpenAI gym is the CarRacing-v0 environment, we suggest exploring the Arcade Learning Environment (ALE) or ROM environments, as these return a pixel representation of the world. Still, these tasks are typically quite hard as they contain a large action space. This is why we suggest a simpler alternative to this problem where you can create your own OpenAI gym environment. This is simple, as when executing the simulation is the same for all OpenAI gym tasks, plus the code can be created to the user's desire, even returning the observation as an image frame, rather than an array of values.

References

[1] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] CarRacing-v0 from the Box2D environment OpenAI gym, link = https://github.com/openai/gym/blob/master/gym/envs/box2d/car_racing.py, note = Accessed: 2022-04-22.

[4] LunarLander-v2 from the Box2D environment OpenAI gym, link = https://github.com/openai/gym/blob/master/gym/envs/box2d/lunar_lander.py, note = Accessed: 2022-04-22.

[5] BipedalWalker-v2 from the Box2D environment OpenAI gym, link = https://github.com/openai/gym/blob/master/gym/envs/box2d/bipedal_walker.py, note = Accessed: 2022-04-22.

[6] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[7] World Models Experiments Otoro blog, link = https://blog.otoro.net/2018/06/09/world-models-experiments/, note = Accessed: 2022-04-25.

[8] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[9] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

[10] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[11] CartPole-v1 from the classic control environment OpenAI gym, link = https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py, note = Accessed: 2022-04-22.

[12] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[13] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[14] Compute Engine Pricing, link = https://cloud.google.com/compute/all-pricing, note = Accessed: 2022-04-25.

[15] Jay W Forrester. Counterintuitive behavior of social systems. *Theory and decision*, 2(2):109–140, 1971.

[16] Le Chang and Doris Y Tsao. The code for facial identity in the primate brain. *Cell*, 169(6):1013–1028, 2017.

[17] R Quian Quiroga, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005.

[18] Nora Nortmann, Sascha Rekauzke, Selim Onat, Peter König, and Dirk Jancke. Primary visual cortex represents the difference between past and present. *Cerebral Cortex*, 25(6):1427–1440, 2015.

[19] Gerrit W Maus, Jason Fischer, and David Whitney. Motion-dependent representation of space in area mt+. *Neuron*, 78(3):554–562, 2013.

[20] Hirson B. Tracking fastballs. *Science Update Interview*, 2013.

[21] Paul J Werbos. Learning how the world works: Specifications for predictive networks in robots and brains. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics, NY*. IEEE, 1987.

[22] David Silver. Lecture 8: Integrating learning and planning, 2015.

[23] Jiirgen Schmidhuber. Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments. 1990.

[24] Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

[25] Open AI gym environments leaderboard, link = https://github.com/openai/gym/wiki/leaderboard, note = Accessed: 2022-04-24.

[26] Arcade Learning Environments, link = https://github.com/mgbellemare/arcade-learning-environment, note = Accessed: 2022-04-22.

[27] Retro AI competition, World Models implementation, author= Dylan Djian link = https://github.com/wassname/world-models-sonic-pytorch, note = Accessed: 2022-04-22.

[28] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[29] John D Bransford, Ann L Brown, Rodney R Cocking, et al. *How people learn*, volume 11. Washington, DC: National academy press, 2000.

[30] Rachael D Seidler. Neural correlates of motor learning, transfer of learning, and learning to learn. *Exercise and sport sciences reviews*, 38(1):3, 2010.

[31] Dipanjan Sarkar, Raghav Bali, and Tamoghna Ghosh. *Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras*. Packt Publishing Ltd, 2018.

[32] Hideki Nakayama. Image feature extraction and transfer learning using deep convolutional neural networks. *IEICE Technical Report; IEICE Tech. Rep.*, 115(146):55–59, 2015.

[33] Grega Vrbančič and Vili Podgorelec. Transfer learning with adaptive fine-tuning. *IEEE Access*, 8:196197–196211, 2020.

[34] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

[35] Gaobo Liang and Lixin Zheng. A transfer learning method with deep residual network for pediatric pneumonia diagnosis. *Computer methods and programs in biomedicine*, 187:104964, 2020.

[36] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials*, pages 15–18. arxiv, 2019.

[37] Xvfb virtual framebuffer X server for X Version 11, link = `https://www.x.org/releases/x11r7.6/doc/man/man1/xvfb.1.xhtml`, note = Accessed: 2022-04-25.

[38] Nathaniel R Goodman. Statistical analysis based on a certain multivariate complex gaussian distribution (an introduction). *The Annals of mathematical statistics*, 34(1):152–177, 1963.

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[40] Christopher M Bishop. Mixture density networks. 1994.

[41] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[42] David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.

[43] Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99:650–655, 2021.

[44] Stephen I Gallant et al. Perceptron-based learning algorithms. *IEEE Transactions on neural networks*, 1(2):179–191, 1990.

[45] Hans-Paul Schwefel. *Numerical optimization of computer models*. John Wiley & Sons, Inc., 1981.

[46] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[47] M Willjuice Iruthayarajan and S Baskar. Covariance matrix adaptation evolution strategy based design of centralized pid controller. *Expert systems with Applications*, 37(8):5775–5781, 2010.

[48] CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, author=Nikolaus Hansen, Youhei Akimoto, and Petr Baudis, link = `https://github.com/cma-es/pycma`, year=2019, note = Accessed: 2022-04-24.

[49] MPI for Python, link = `https://mpi4py.readthedocs.io/en/stable/intro.html#what-is-mp`, note = Accessed: 2022-04-25.

[50] GeForce GTX 1060 — Specifications - Nvidia, link = `https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1060/specifications/`, note = Accessed: 2022-04-25.

[51] Gustav Grund Pihlgren, Fredrik Sandin, and Marcus Liwicki. Improving image autoencoder embeddings with perceptual loss. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[52] Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. Vizdoom competitions: Playing doom from pixels. *IEEE Transactions on Games*, 2018.

[53] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[54] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

Appendix A

Code

The code for the file is held within a zipped folder called 'World Models CandNo215816.zip'. The implementation of all code used in this study is held within multiple.py files, which must be run in order to train the algorithm from scratch. The data collected to run these files won't be included, otherwise, we wouldn't be able to submit our code.

The file paths used within the code were appropriate for our the locations of our files within our PC. These should be updated so that it can be ran on your PC. The file paths have been commented to show you where you should put your own file path.

This code was ran on Visual Studio using an anaconda environment. Make sure that you download all the necessary libraries, e.g Tensorflow, keras, numpy etc. To be able to run the code successfully.

Appendix B

Original Project Proposal

Candidate Number: 215816
Supervisor: Christopher Buckley

Working title - Exploring Transfer Learning in World Models

The world contains masses of information, however, we only perceive the information which is useful to us. Based on this information, humans create a mental model of their environment, enabling them to makes decisions and actions in the real world [1].

Building on this idea, David Ha and Jürgen Schmidhuber created their own virtual mental model (World Model), to solve a particular task in its own environment [2]. Nevertheless, one might consider how will the model perform on a similar task, will it be able to solve it faster than the previous task, having trained on it? What about a completely different task?

This project will explore the idea of Transfer Learning [3], specifically, how much knowledge has been transferred from the previous task, to the new task.

Aims

The main aim of this project is to demonstrate the impact of transfer learning on an OpenAi gym [4] task, using the World Models code as a basis [5].

Large AI systems require a vast amount of time and resources for training, making it very expensive and is harmful for the environment. Transfer learning can reduce both time and resources, which is what we are trying to achieve.

Objectives

Primary Objectives

- Research existing applications of Transfer Learning and integrate it into the World Models.
- Modify the World Models code to be able to execute the training and the testing on a local machine.
- Train the model on the CarRacing-V0 environment from OpenAI gym [4] and use Transfer Learning to solve the LunarLander-V2 environment from OpenAI gym [4].
- Give quantitative and qualitative results for Transfer Learning vs training from scratch.

Extension Objectives

- Compare 2 AI agents, one with self determined actions and one with random actions on a new task using Transfer Learning.
- Use transfer learning with World Models on a 2D video game instead of Open AI gym.

Relevance

This project involves programming in Python, plotting graphs on Matplotlib, using the OpenAi gym package to visualise the AI in the environment, and creating Neural Networks which has been taught in detail on my second year modules, specially in Acquired Intelligence & Adaptive Behaviour. Disregarding the technical side of the project, this investigation will also test my research skills, organisational skills and my quality of presentation of results.

This project will expand my knowledge on adaptive systems and machine learning, which could be useful when applying for a postgraduate course or research position.

Resources Required

- A powerful CPU and GPU Desktop/Laptop computer (Minimum CPU Intel® Core™ i5-9400F and GPU NVIDIA® GeForce® GTX 1660.
- The World Models code (Github, code and authors have been properly referenced).

References

[1] J. W. Forrester, “Counterintuitive behavior of social systems,”Theory and decision,vol. 2, no. 2, pp. 109–140, 1971.
[2] D. Ha and J. Schmidhuber, “World models,”arXiv preprint arXiv:1803.10122, 2018.
[3] L. Torrey and J. Shavlik, “Transfer learning,” inHandbook of research on machinelearning applications and trends: algorithms, methods, and techniques, pp. 242–264,IGI global, 2010.
[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, andW. Zaremba, “Openai gym,”arXiv preprint arXiv:1606.01540, 2016.
[5] D. Ha and J. Schmidhuber, “Recurrent world models facilitate policy evolution,”inAdvances in Neural Information Processing Systems 31, pp. 2451–2463, CurranAssociates, Inc., 2018.https://worldmodels.github.io.

Appendix C

Old Graphs

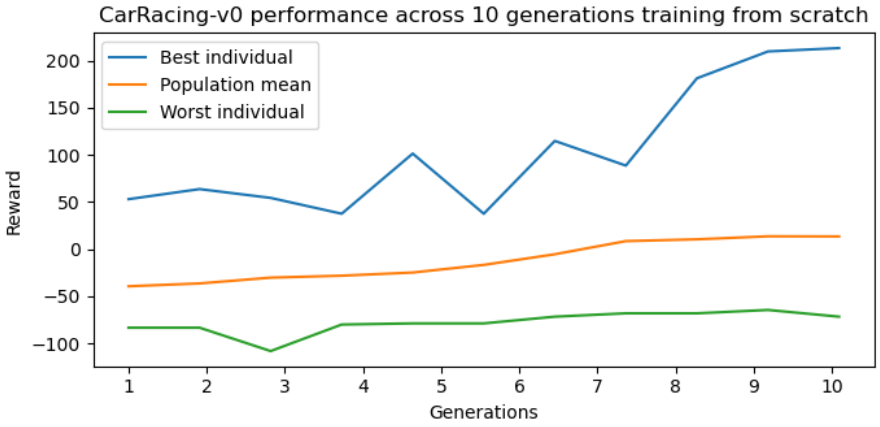
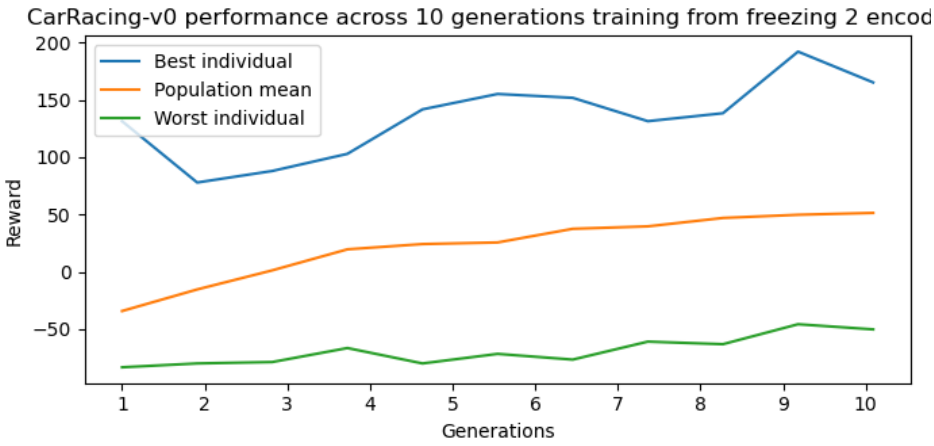


Figure 6.1: CarRacing-v0 performance when training the World Models from scratch on a single run. We decided to plot the best individual, mean population and worst individual from the CMA Evolutionary Strategy. We can clearly observe that the performance of the population is slowly improving.

a)



b)

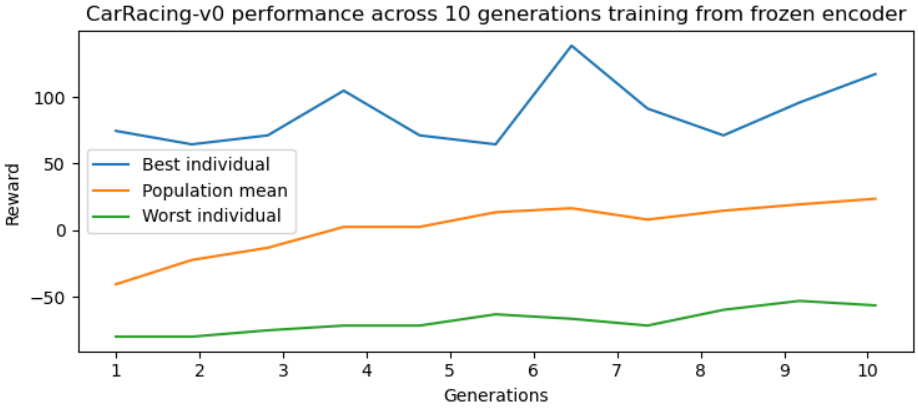


Figure 6.2: CarRacing-v0 performance when training the World Models from freezing 2 layers from the encoder (**a**) and freezing the full encoder (**b**) on a single run. We decided to plot the best individual, mean population and worst individual from the CMA Evolutionary Strategy. We can clearly observe that for both TL techniques the mean population, best and worst individual seem to be improving their performance.

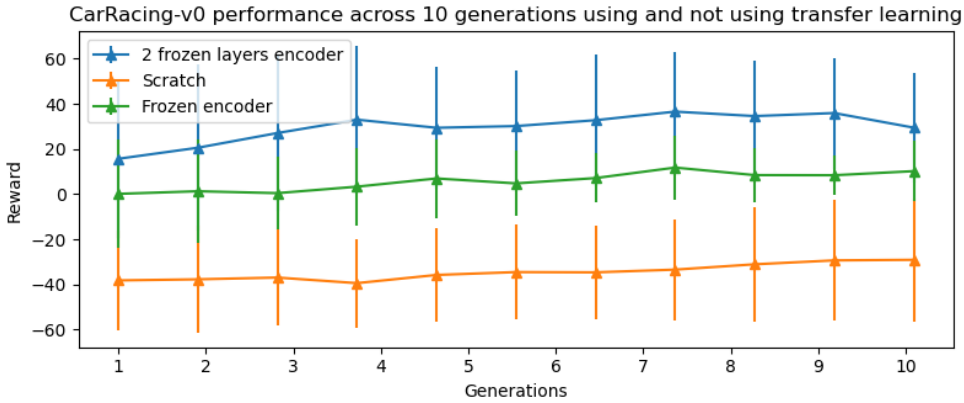
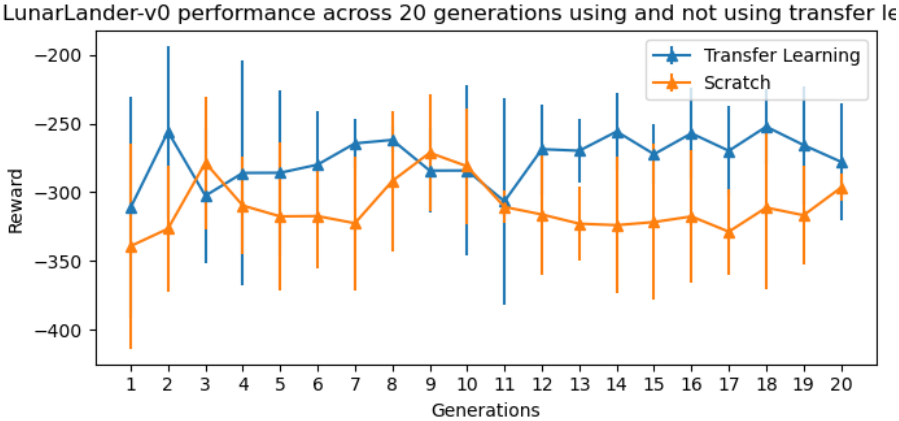


Figure 6.3: Comparison on the the reward achieved across 10 generations for the agents in the CarRacing-v0 task. We clearly observe that although the mean population is improving overtime, when using TL, we start acquiring higher reward.

a)



b)

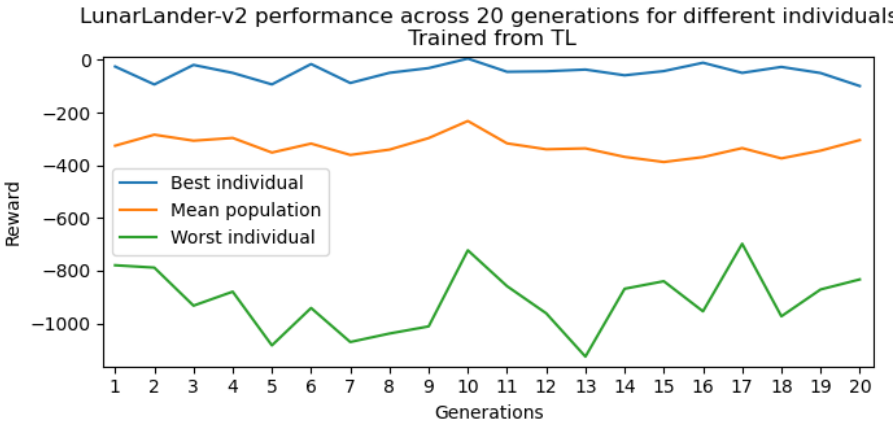
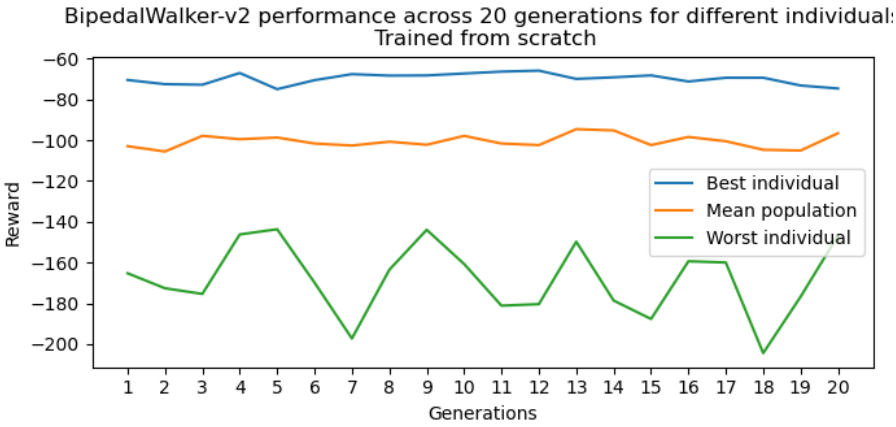


Figure 6.4: LunarLander-v2 performance across 20 generation. We plot the performance of the best individual, mean population and worst individual from the CMA ES, trained from TL.

a)



b)

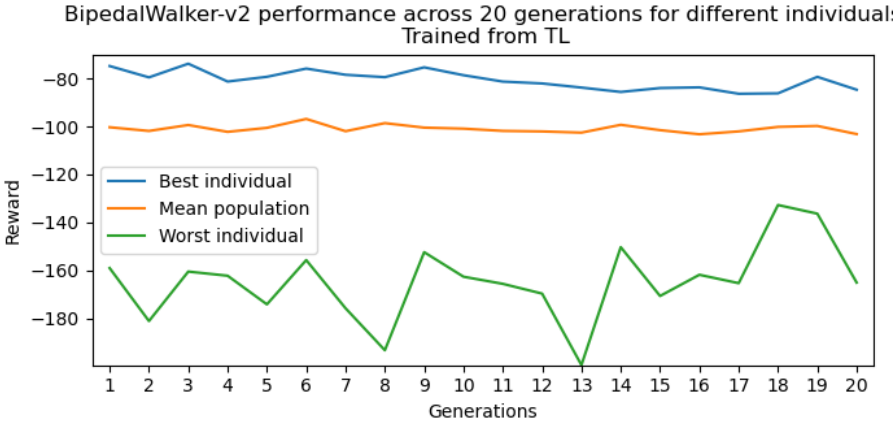


Figure 6.5: BipedalWalker-v2 performance across 20 generation. We plot the performance of the best individual, mean population and worst individual from the CMA ES, trained from TL.

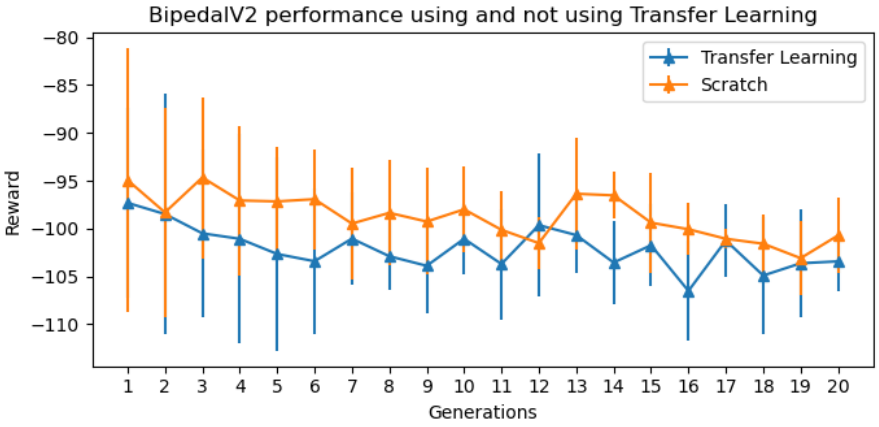


Figure 6.6: BipedalWalker-v2 performance across 20 generation. We can clearly observe that the mean population for both training techniques does not seem to either increase or decrease across 20 generations.

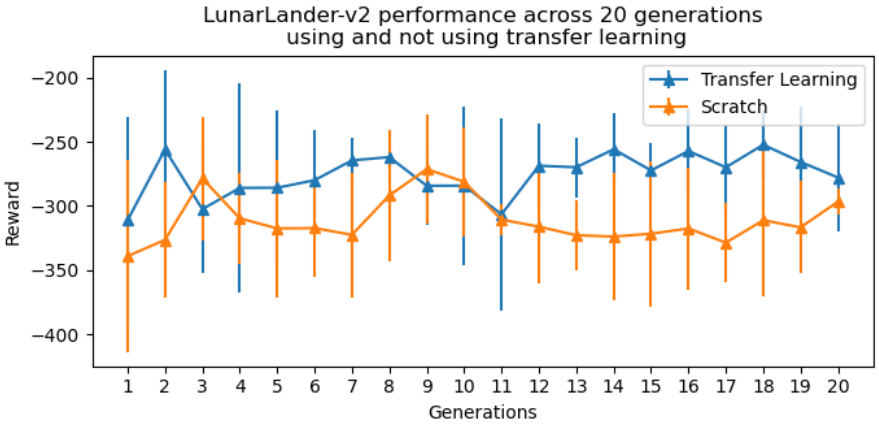


Figure 6.7: LunarLander-v2 performance across 20 generation. We can clearly observe that the mean population for both training techniques does not seem to either increase or decrease across 20 generations.

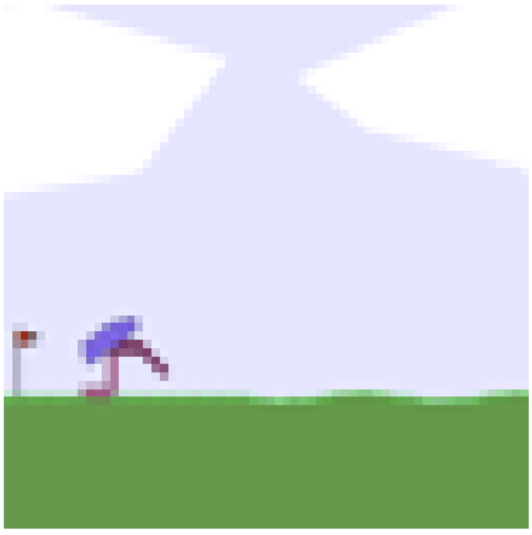


Figure 6.8: BipedalWalker-v2 image frame from environment.

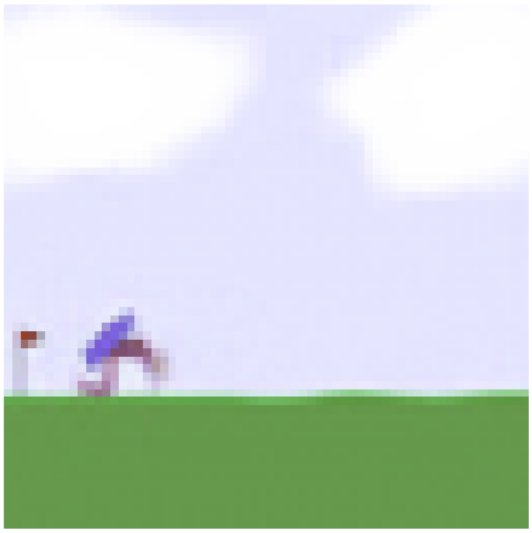


Figure 6.9: BipedalWalker-v2 image frame recosntructed from V model.

Appendix D

Ethical Review Application

Ethical Review Application (ER/BCU20/2) Benat Underwood

Parent Application:ER/BCU20/1

Project Title	Exploring Transfer Learning in World Models (COPY)
Status	Draft
Email	bcu20@sussex.ac.uk
Phone No.	07510761158
Applicant Status	UG
Department	Informatics
Supervisor	Buckley, Christopher L
Project Start Date	26-Sep-2021
Project End Date	10-May-2022
External Funding in place	No
External Collaborators	No
Funder/Project Title	
Name of Funder	
Project Description	World Models (Ha & Schmidhuber, 2018), code on github https://github.com/worldmodels/worldmodels.github.io, is an AI model which takes actions on a virtual environment according to what it sees in the environment. However, the main idea of this project is to explore a concept called transfer learning, a machine learning task which transfers knowledge from solving a source task to solve a new task, on this code. The main aim of the project is to use Transfer learning to minimise the training time of World Models and to prove that Transfer learning is more effective when training an AI.

Ethical Review Form Section A (ER/BCU20/2)	
Question	**Response**
>> Checklist	
A1. Will your study involve participants who are currently or potentially vulnerable or unable to give informed consent or in a dependent position (e.g. people under 18, people with learning difficulties, over-researched groups or people in care facilities)?	No
A2. Will participants be required to take part in the study without their consent or knowledge at the time (e.g. covert observation of people in non-public places), and / or will deception of any sort be used? Please refer to the British Psychological Society Code of Ethics and Conduct (or similar guidelines) for further information.	No
A3. Unless specifically and clearly consented (e.g. a media release form), will it be possible, through a research output, to identify participants in any way? (This does not include taking email details for participant prize draws or identifying participants from signed consent forms or holding identity encryption spreadsheets that are stored securely separate from the research data).	No
A4. Might the study induce psychological stress or anxiety, or produce humiliation or cause harm or negative consequences beyond the risks likely to be encountered in the everyday life of the participants?	No
A5. Is there a risk that the research topic might lead to disclosures from the participant concerning their beliefs, involvement in illegal actions or any other activities that may represent a threat to themselves or others?	No
A6. Will the study involve collecting any personal special category information* in a form that could allow the participant/ participants to be identified? [* identifiers relating to race, ethnic origin, politics, religion, trade union membership, philosophical beliefs, genetics, biometrics, health, sex life or sexual orientation]	No
A7. Will any drugs, placebos or other substances (such as food substances or vitamins) be administered as part of this study and will any invasive or potentially harmful procedures of any kind will be used?	No
A8. Will your project involve working with any substances and / or equipment which may be considered hazardous?	No
A9. Will your study involve the taking and/or storage of human tissue that falls under the Human Tissue Act (HTA)? http://www.sussex.ac.uk/staff/research/governance/erp_overview/humantissue	No
>> Risk Assessment	
A10. If you have answered Yes to ANY of the above questions, your application may be considered as HIGH risk. If, however you wish to make a case that your application should be considered as LOW risk please enter the reasons here. Researchers should note that SREOs or C-RECs may decide NOT to agree with the case that you have made.	

Ethical Review Form Section B (ER/BCU20/2)	
Question	**Response**
>> Data Collection and Analysis (Please provide full details)	
B1. PARTICIPANTS: How many people do you envisage will participate, who are they, and how will they be selected?	N/A
B2. RECRUITMENT: How will participants be approached and recruited?	N/A
B3. METHOD: What research method(s) do you plan to use; e.g. interview, questionnaire/self-completion questionnaire, field observation, audio/audio-visual recording?	I will only research papers related to the topic of my project. I won't collect research from participants as I won't need any.
B4. LOCATION: Where will the project be carried out e.g. public place, in researcher's office, in private office at organisation?	The research will be carried out in a private office and the computer labs from the University of Sussex School of Engineering & Informatics.
B5. PARTICIPANT WELLBEING: Will the study involve engaging participants in the discussion of potentially distressing or sensitive topics? (e.g. sexual activity, drug use, ethnicity, political behaviour, potentially illegal activities). If so, please set out how you will manage the well-being of participants.	N/A
>> Confidentiality and Anonymity	
B6. Will questionnaires be completed anonymously and returned indirectly?	N/A
B7. Will research data only be identifiable by a unique identifier (e.g. code/pseudonym)? If Yes, please explain how this will be attributed in B11a below.	N/A
B8. Will lists of identity numbers or pseudonyms linked to names and/or addresses be stored securely and separately from the research data? If Yes, explain how this will occur in B11a below.	N/A
B9. Will all place names and institutions which could lead to the identification of individuals or organisations be changed unless this is consented to explicitly in the consent form?	N/A
B10. Will all personal information gathered be treated in strict confidence and never disclosed to any third parties?	N/A
B11. Can you confirm that your research records will be held in accordance with data protection regulations? (http://www.sussex.ac.uk/ogs/policies/information/dpa)	N/A
B11a. Please explain how ANY identifiable personal and/or research data will be managed and securely stored ensuring that participants have given appropriate informed consent for this.	N/A
B12. Do you intend to use the research data for any purpose other than that for which consent is explicitly given? If so, please explain below	N/A
B12a. If you answered NO to any of the above in this section (or think more information could be useful to the reviewer) please explain here:	
>> Informed Consent and Recruitment of Participants	

B13. Will all respondents be given an Information Sheet and be given adequate time to read it before being asked to agree to participate?	N/A
B14. Will all participants taking part in an interview, focus group, observation (or other activity which is not questionnaire based) be asked to sign a consent form? If you are obtaining consent another way (such as verbally), please explain under B17 below.	N/A
B15. Will all participants self-completing a questionnaire be asked to show consent to participate by a specific and identifiable action? (Give details in B17 below)	N/A
B16. Will all participants be told that they can withdraw their participation at any time during the research and can ask for their data to be destroyed and/or removed from the project until it is no longer practical to do so?	N/A
B17. If you answered NO to any of the above in this section (or think more information will be useful to the reviewer) please explain here:	
>> Context	
B18. Is DBS (Disclosure and Barring Service) clearance necessary for this project? If yes, please ensure you complete the next question.	No
B19. Are any other ethical clearances or permissions (internal or external) required? Please see the help text (i) for further details.	No
B19a. If yes, please give further details including the name and address of the organisation. If other ethical approval has already been received please attach evidence of approval, otherwise you will need to supply it when ready. (You do not need to provide evidence of a current DBS check at this point).	
B20. Does the research involve any fieldwork - Overseas or in the UK?	No
B20a. If yes, where will the fieldwork take place? If undertaken overseas you must attach an OTSSRA form. In the event that the Foreign and Commonwealth Office has specific travel warnings in place for the country (ies) to be visited you will also need to provide a detailed risk assessment. https://www.gov.uk/foreign-travel-advice	
B21. Will any researchers be in a lone working situation?	No
B21a. If yes, briefly describe the location, time of day and duration of the lone working. What precautionary measures will be taken to ensure safety of the researcher(s)?	
>> Any further concerns	
B22. Are there any other ethical considerations relating to your project which have not been covered above?	No
B22a. If yes, please explain:	