World Models

Nick Sawhney

May 9, 2021

1 Intro

The intuition behind World Models comes from the ways that humans perceive the world around them. In order to process the vast amount of information we receive from our senses, we form abstract models of the world, and act based on information from those models. Furthermore, our brains *predict* the future based on these models – this is what allows us to have extremely fast reflexes in certain situations, like avoiding potential car crashes or hitting fast-moving baseballs. The authors of World Models use this as inspiration, combining various models – one which provides an abstraction of "sensory input" (a model of the world) as well as a prediction (a model of the abstract representation of the future) – with a controller model to perform reinforcement learning tasks such as driving a virtual car in a virtual environment. The authors also show that, once the environment and predictive models are trained, the controller model can be trained within data generated by the sensory and predictive models, allowing the intelligent agent to be "trained within its dreams" but still perform well within the intended environment

2 Models

The first model to be trained is a Convolutional Variational Autoencoder, or CVAE, also referred to as V in the original paper. The job of V is to turn a large amount of 2D pixel information into a single smaller vector z which represents the information in the 2D image. This allows the information from the Car Racing (or any other) environment to be compressed into a

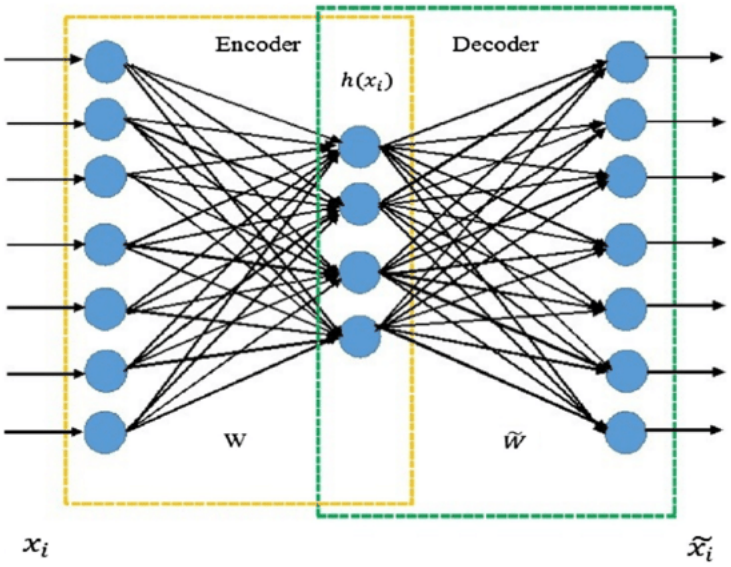

Figure 1: Basic VAE architecture. For the purposes of this paper, replace $h(x_t)$ with z_t

smaller single vector. This is achieved by training an "encoder" network with convolutional and dense layers to convert from pixel data into the latent z space. This z vector is then fed to a "decoder" network which converts it back into the original image. The reproduction is not identical, since some information is lost in the compression into the latent space. However, it has been shown that z can encode the important information about the environment. This compressed representation allows for decreased complexity for the controller model. A basic representation of the architecture is shown below. Importantly, in the World Models paper convolutional layers were used in to encode pixel data from the OpenAI Gym environment, followed by dense layers. The decoder uses deconvolutional layers to reproduce the original image. To train the VAE, rollouts of the environment with a random policy are collected, with the pixel values and actions recorded and saved. These are used to train the V model. The loss function of V minimizes the difference between the reconstructed image from the decoder and the original image. This ensures that the information in z represents important features of the environment.

Next, a Recurrent Neural Network is trained to predict the future values of z based on its current value – in other words to predict z_{t+1} given z_t. In the World Models paper, this model, referred to as M, outputs a probability density function $p(z)$ as a mixture of Gaussian distributions instead of directly attempting to predict z. This type of model is called a Mixture Density

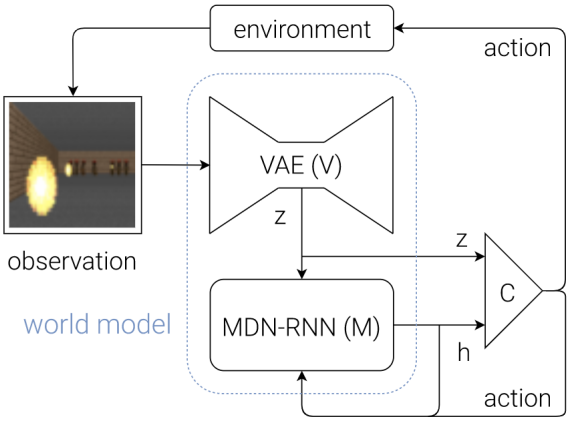

Figure 2: World Model Architecture

Network because it outputs a mixture of probability distributions. The RNN also takes as parameters a_t, an action taken at time t, as well as h_t, a vector representing the "hidden state" of the RNN. The model is trained on outputs from the encoder from the collected episode rollouts.

Finally, a simple model C is trained to be the "controller", responsible for determining the action taken by the agent in the environment at each time step. The controller uses the hidden state of the RNN h_t and the latent representation of the environment z_t. Since the RNN was trained to predict future zs, h_t encodes an approximation of the future. Since the VAE was trained to encode a representation of the environment in z, the controller also has an approximation of the current state of the environment. By this step, most of the complexity needed to encode the environment and predictions has already been trained into V and M, therefore C is able to be a very simple neural network with a single layer, defined as:

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

where W_c and b_c are weight and bias parameters of C.

The authors describe a number of practical benefits to keeping C simple while keeping complexity in V and M. V and M are models which can be efficiently trained using backpropogation on GPU-accelerated machines, and C being a simple linear model also allows for increased efficiency during training as well as unconventional training methods. To train the controller,

the authors use Covariance-Matrix Adaptation Evolution Strategy, a numerical optimization process which combines optimization and stochasticity to allow the controller to both optimize performance and explore new strategies. This exploration-vs-exploitation tradeoff is at the heart of optimizing controllers in Reinforcement Learning.

3 Results

The authors train two versions of the controller, one using only z_ts encoded by the VAE and the other using both z_t from the VAE and the predicted z_{t+1} from the RNN. They show that when the prediction is not included, the car "wiggles" more, struggling to find a smooth path through the racetrack. However, when predictions of the future are included, the driving is much smoother, indicating that the prediction of the future helps the controller drive more like a human would.

4 Training within Dreams

Finally, the authors suggest a strategy for training where the VAE and RNN produce the environment within which the controller can be trained and show that a model trained within a generated "dream" environment can be used within the real environment. First, the VAE is used to generate a z_t, then the RNN is used to generate successive z_{t+1} to create an episode for the controller to use for training. They find that the policy learned inside the real environment functions adequately inside the dream environment, and also that policies learned inside the dream environment can be transitioned back into the real environment. This means the VAE and RNN can be used in combination to generate more training data, and is a testament to the representational power of a latent vector and its future predictions.