Generating Synthetic Training Images for Deep Reinforcement Learning of a Mobile Robot

Sumeth Yuenyong and Qu Jian

***Abstract*— This paper proposes the use of variational autoencoder (VAE) to generate synthetic training images for deep reinforcement learning of a mobile robot. Deep reinforcement learning typically requires millions of interactions with the real world in order to learn good control policies, which is impractical for robotic tasks. Using synthetic images generated by a VAE, one may be able to reduce the number of interactions by running a deep reinforcement learning algorithm off these images, instead of real ones captured by the camera. Our experiment shows, for this particular task, that the VAE can generate synthetic images which are almost non-discernible from those obtained by direct reconstruction of real images.**

***Index Terms*—variational autoencoder; deep reinforcement learning; machine learning**

I. Introduction

DEEP reinforcement learning that combines reinforcement learning and deep neural network together had demonstrated tremendous success in making a computer learn how to play Atari video games at human or super-human performance. The method, called DQN uses no teaching signal other than the raw game screens and the game score and is able to learn good control policies for a wide rage of Atari games while using the same architecture and hyper-parameters for each [9], [10]. However, despite great success in the domain of video games/simulation (e.g. Atari/Mujoco physics simulator), deep reinforcement learning using only high-dimensional raw image input have had relatively limited success when it comes to real-world domain such as robotic/mobile agent tasks [1], [8], [11], [13], [14]. The main reason for this is the very large number of iterations needed by state-of-the-art deep reinforcement learning algorithms to converge. For example, in [10], the DQN network was trained using 50 millions images. While this may be fine for learning to play a video game, where one can just leave the game emulator running for days non-stop, for real-world robotic tasks it is simply impractical. This had forced researchers take another approach when faced with real-world reinforcement learning task, which is to reduce the dimension of the training images by various techniques, developed models for state transition in this low dimensional space and then use model reference adaptive control (MRAC) to train the agent such as in [11], [13]. While this approach works fine and require much fewer iterations than something like DQN, it is limited to tasks where it is possible to provide a reference image of where one wants the agent to move to, i.e., they are limited to simple direct movement tasks such as swinging the pendulum by 180 or place the hand of the robot arm in a designated area only.

Generative Models [4] refers to a class of neural network architectures trained to produce outputs that looks similar to, but not actually part of, the training set used to train them. Since real-world data is limited for robotic deep reinforcement learning, one possible way that may be able to get around this issue is to use generative models that had been trained with real-world images of the agent moving about its environment to generate additional training images without having to actually interact with the real world. The rationale of this approach is that it takes relatively little data to train a generative network to produce realistic-looking images compared to training the DQN network, therefore, only a much smaller amount of real-world interactions will be needed.

Recently, a particular type of generative network called Variational Autoencoders (VAE) [7] had gained popularity due to its generative performance and a solid theoretical foundation rooted in Bayesian inference. In this work, we demonstrate that VAE can be used to generate images of a mobile robot moving in its environment that look like actual real-world images captured by the camera (after some image processing had been applied); with the ultimate goal of being able to greatly reduce the number of real-world interaction that a mobile reinforcement learning would have to make during training.

II. Variational Autoencoder

The task of the VAE is to generate outputs X in the high-dimensional image space such that each output looks like it came from a training set, e.g., generating images of hand-written digits after being trained with the MNIST database [2]. Its structure is similar to the standard autoencoder [5], which has an encoder part that transforms the input into the lower-dimensional "latent" space and a decoder part that reconstructs it back. Unlike the standard autoencoder however, the derivation of the VAE starts from the decoder. Figure 1 shows the decoder part of the VAE - a network that transform a point in the latent space to the image space. The latent representation z is sampled from a multivariate distribution $P(z)$, then passed through a decoder network parametrized by some parameter vector θ_d which produces the output X that will look similar to the training set after training.

A detailed tutorial to VAE can be found in [3], which we briefly summarize here. VAE is derived from maximizing the

Sumeth Yuenyong is with the Department of Computer Engineering, Faculty of Engineering, Mahidol University. 25/25 Phutthamonthon 4 Rd., Salaya, Nakhon Pathom, 73170 Thailand, email: sumeth.yue@mahidol.ac.th

Qu Jian is with the School of Science and Technology, Shinawatra University. 99 Moo 10 Bang Toey, Sam Khok District, Pathum Thani 12160, Thailand

probability $P(X)$ of observing X that is similar to the training set, given the sampled latent representation z

$$P(X) = \int P(X|z;\theta_d)P(z)dz. \tag{1}$$

The prior in the latent space $P(z)$ is restricted to be of the form $\mathcal{N}(\mu, \Sigma)$, where μ is a vector of means and Σ is a diagonal covariance matrix. We have no access to the posterior $P(X|z;\theta_d)$ and must resort to using variational inference technique [12] in order to maximize (1). Variational inference works by constructing another distribution $Q(z|X;\theta_e)$ which is implemented by the encoder network. It is fed with real images and produce the mean vector μ and the diagonal elements of Σ that defines the prior $P(z)$ in latent space that when sampled from, allows the decoder to generate realistic-looking X. During training of the VAE two losses are minimized: the latent lost, which is the KL divergence between $Q(z|X;\theta_e)$, i.e., the distribution parametrized by the outputs of the encoder, and the standard multivariate normal distribution; the second is the reconstruction loss, which is the mean squared error or cross-entropy between the reconstructed image and the input image.

$$\begin{aligned} L_{\text{total}} &= L_{\text{latent}} + L_{\text{reconstruction}} \\ &= \text{KL}(\mathcal{N}(\mu,\Sigma)||\mathcal{N}(\mathbf{0},\mathbf{I})) + ||X - f(z)||^2, \end{aligned} \tag{2}$$

where f denotes the output of the decoder given some latent representation z.

Finally, for the entire VAE network to be trainable by stochastic gradient descent, the "reparameterization trick" is used so that z can be sampled from $\mathcal{N}(\mathbf{0},\mathbf{I})$ and shift the sampling operation (which is not differentiable) from the middle of the signal-flow graph to an input node instead (see Figure 4 in [3] for details). Once trained, the VAE can be used to generate additional synthetic images by first feeding it with a batch of real images from the training set so that μ and Σ are defined, then we can sample points from a standard multivariate normal and then transform those points such that it is as if they were sampled directly from $P(z)$ by

$$z = \mu + \epsilon\sqrt{\text{diag}(\Sigma)}, \tag{3}$$

where ϵ and z respectively are vectors representing a point sampled from the standard multivariate normal and a point under the $P(z)$ distribution. Here, with a slight abuse of notation, we use $\text{diag}(\Sigma)$ to mean a vector that holds the diagonal components of Σ. The transformed point z is fed through the decoder network in order to produce the output X. Alternatively, VAE can also be used in the encoding mode, where the mean vector μ outputted by the encoder is taken to be the latent representation of an input image.

III. Experimental Setup

The environment for our mobile robot agent is a black plastic sheet with white circles for obstacles and a white line as the goal. The objective of this task is the following: the agent starts from the left edge of the board, and must drive to the goal line, while avoiding the obstacles along the way, as shown in Figure 2. The closer the agent gets to the finish line, the higher the reward. If it hits an obstacle, the award is greatly reduced. This task cannot be achieved using MRAC based methods such as [1], [11], because no reference image can tell the agent to avoid the obstacles.

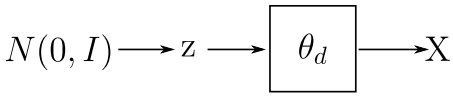

Fig. 1: The decoder part of the VAE. A point is drawn from the standard multivariate normal, transformed into z and passed through a decoder network parametrized by θ_d to produce the output image X.

In order to track the movement of the agent, a camera is placed looking down from the top, producing bird-eye view images. The specific camera we had chosen is Sony's PS3Eye camera, because it is very affordable and unlike other webcams at this price range is capable of recording at 120 fps. High frame rate is desirable because we can get more images in the same amount of time and avoid any possible motion blur caused by the movement of the agent.

The agent is a small RC car with two wheels driven by two stepper motors and controlled from a PC via Bluetooth. A picture of the agent (without batteries installed) is shown in Figure 3. The agent can perform 4 possible discrete actions: move forward, turn left/right (in place) and go back. The wheel RPM that the agent performs each action is hard-coded at 60 RPM in the Arduino board on the agent. The top of the agent is covered with a black plastic plate with an isosceles triangle painted on top that looks like an arrow pointing in the forward direction of the agent. The purpose of the cover is to hide all the circuits underneath which has nothing to do with the state of the agent in its environment: namely the x-y coordinate in pixels and the direction that the arrow/triangle is pointing to. By hiding the body of the agent under the plastic plate, the VAE only has to generate the white triangle representation of the agent, without the complicated but unnecessary features underneath.

Furthermore, we also applied some image processing to the raw camera images captured which can contain a lot of noise. Specifically, each image is converted to grayscale, thresholded to black-and-white and resized to 80 by 80 pixels. An example of a processed image that is part of the training set for the VAE is shown in Figure 4. This image, while heavily simplified, contains all the information that a reinforcement learning algorithms needs; the position/orientation of the agent is clearly discernible, as well as the positions of each obstacles and the position of the finish line. It is also free from noise that can cause problems for a reinforcement learning algorithm.

The summary of the steps in our experiment is shown in Figure 5. The agent was controlled manually through a PC and driven systematically around the environment in the lawn mower style pattern, stopping each time it has traveled about half its length to make a 360 turn in-place. This pattern sweeps through the different states of the agent in this environment. We kept driving in this pattern until the batteries ran out. Total recording time was about 1 hour and we obtained close to

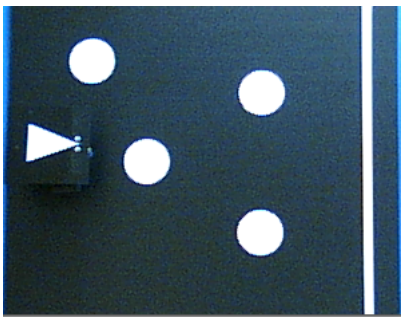

Fig. 2: The agent on the playing field as seen by the camera.

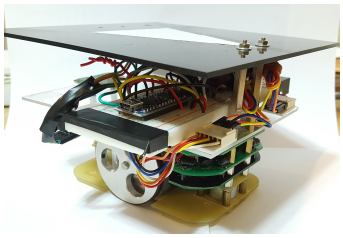

Fig. 3: The mobile agent.

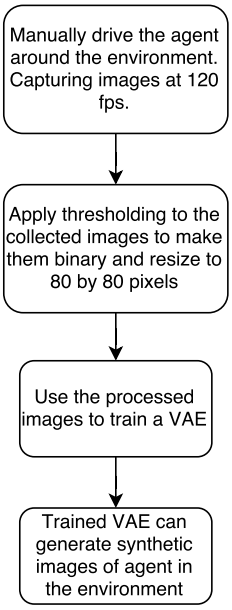

Fig. 5: The steps of our experiment.

100,000 images. We processed the images and then trained a VAE with the following architecture and parameters: the encoder network has three fully-connected layers with 2500 units in the first layer, 500 units in the second layer and 20 units (the dimension of the latent space) in the last layer; the decoder network is also fully-connected with 500 units in the first layer, 2500 in the second and 6400 in the last layer (the dimension of the output image is 80 by 80 pixels). Each unit in both the encoder and decoder networks has relu activation function, except for the last layers which have linear and sigmoid activation functions for the encoder and decoder, respectively. The other hyper parameters were: batch size of 100, learning rate of 0.001, the number of training epochs was 250 and the optimizer used was Adams [6]. The code was implemented in Tensorflow.

IV. Experiment Results

We trained the VAE using around 80,000 preprocessed images on a PC with a GTX 980Ti GPU. Total training time was around 45 minutes. Because Tensorflow by default reserves the entire GPU memory when it runs, we did not measure the GPU memory usage. GPU utilization as reported by the nvidia-smi utility was around 85%. The training loss curve as calculated by (2) is shown in Figure 6.

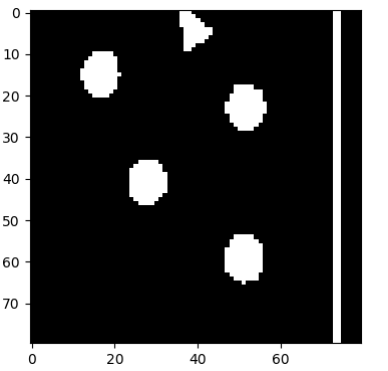

Fig. 4: An example of a processed image, ready to use to train the VAE.

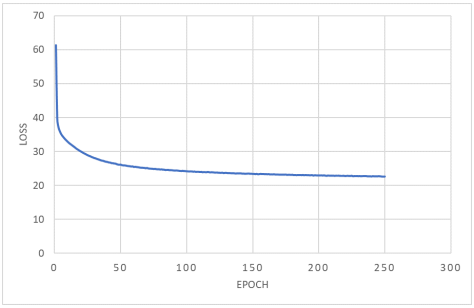

Fig. 6: The learning curve during training of the VAE.

Figure 7 shows the result of running the trained VAE in the "reconstruction" mode. It is fed with inputs from a separate test set, there was no sampling in the latent layer and the output of the encoder network is directly fed into the decoder network. Therefore ideally the output of the whole network should look exactly like the input. It can be seen from Figure 7 that the output of the VAE is very close to the input, with only minor blurring of the agent, but its position is clearly discernible, as well as the direction that the arrow is pointing to. This shows that the latent layer is a good compact representation of the input.

Figure 8 shows the images obtain by running the VAE in generating mode. First a batch of real images from the test set is fed into the encoder network to obtain μ and Σ from the output of the last layer. Then points are sampled from the standard multivariate normal and then transformed to points under $P(z)$ using (3), the transformed points are then fed into the decoder network to produce the synthetic images, some

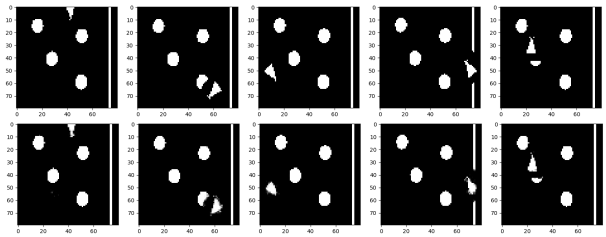

Fig. 7: (Top) The input images into the VAE and (bottom) the reconstructed images.

of which are shown (chosen at random) in Figure 8. It can be seen that even for a human it is hard to tell the difference between the reconstructed and the generated images.

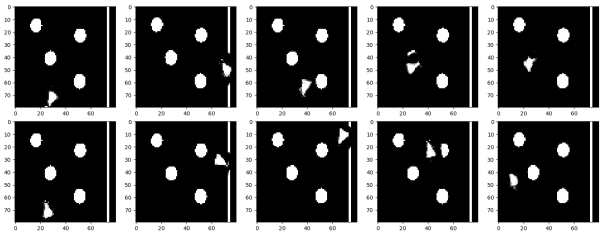

Fig. 8: Synthetic images created from running the VAE in the generating mode.

V. Conclusion

In this paper, the use of VAE to generated synthetic images for a mobile robot reinforcement learning task was proposed. The trained VAE was able to produce realistic-looking images that is hard to discern from images reconstructed directly from real inputs. Although the generated images look very similar to real images, there were some images in which the shape of the agent was not clear, i.e., it was difficult to tell which direction the agent is pointing too. In this regard, it seems possible to improve the results presented here further by adjusting the network architecture and hyper parameters, and experimenting with convolutional structures to see if that would improve the results. Another possible way of improving the result may be simply to train with a larger set of images. We think this may be the case because looking at the result in Figures 8, the circles and the line were always sharp, unlike the agent which sometimes was blurry. This maybe because the VAE had seen thousands of images with the line and circles in the same positions In contrast, the number of images with the agent at any particular orientation and location in the field is likely to be few compared to the size of the training set. Since the training time here was just 45 minutes, much larger training set would still be feasible to train using the hardware we currently have available.

Acknowledgment

This research was supported by the Thailand Research Fund's New Faculty Member Grant (grant number: MRG5980221).

References

[1] J.-A. M. Assael, N. Wahlström, T. B. Schön, and M. P. Deisenroth. Data-efficient learning of feedback policies from image pixels using deep dynamical models. *arXiv preprint arXiv:1510.02173*, 2015.

[2] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[3] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[5] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[8] S. Lange, M. Riedmiller, and A. Voigtlander. Autonomous reinforcement learning on raw visual input data in a real world application. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[11] N. Wahlström, T. B. Schön, and M. P. Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251*, 2015.

[12] M. J. Wainwright, M. I. Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[13] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, pages 2728–2736, 2015.

[14] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*, 2015.

Sumeth Yuenyong received his B.Eng. in Electrical Engineering and M.Eng in Information and Communication Technology for Embedded Systems; from Sirindhorn International Institute of Technology (SIIT) in 2004 and 2010, respectively. He received the Japanese Government Scholarship (Monbukagakusho) to study in Japan and obtained a M.Eng and a D.Eng in Communication and Integrated Systems from Tokyo Institute of Technology in 2011 and 2014, respectively. Currently he is with the Department of Computer Engineering, Faculty of Engineering, Mahidol University. He received a New Faculty Member grant from the Thailand Research Fund in 2016. His research interests include: signal and image processing, deep learning and embedded systems.

Qu Jian is a lecturer with the School of Information Technology, Shinawatra University, Thailand. He received Ph.D. with Outstanding Performance award from Japan Advanced Institute of Science and Technology, Japan, in 2013. He received B.B.A with Summa Cum Laude honors from Institute of International Studies of Ramkhamhaeng University, Thailand, in 2006, and M.S.I.T from Sirindhorn International Institute of Technology, Thammast University, Thailand, in 2010. His research interests are natural language processing, intelligent algorithms, machine learning, machine translation, information retrieval and image processing.