Using World Models for Pseudo-Rehearsal in Continual Learning

Nicholas Ketz¹*, Soheil Kolouri¹, Praveen Pilly¹

¹Information and Systems Sciences Laboratory, Center for Human Machine Collaboration, HRL Laboratories, Malibu, California 90265
{naketz, skolouri, pkpilly}@hrl.com,

Abstract

The utility of learning a dynamics/world model of the environment in reinforcement learning has been shown in a many ways. When using neural networks, however, these models suffer catastrophic forgetting when learned in a lifelong or continual fashion. Current solutions to the continual learning problem require experience to be segmented and labeled as discrete tasks, however, in continuous experience it is generally unclear what a sufficient segmentation of tasks would be. Here we propose a method to continually learn these internal world models through the interleaving of internally generated rollouts from past experiences (i.e., pseudo-rehearsal). We show this method can sequentially learn unsupervised temporal prediction, without task labels, in a disparate set of Atari games. Empirically, this interleaving of the internally generated rollouts with the external environment’s observations leads to an average 4.5x reduction in temporal prediction loss compared to non-interleaved learning. Similarly, we show that the representations of this internal model remain stable across learned environments. Here, an agent trained using an initial version of the internal model can perform equally well when using a subsequent version that has successfully incorporated experience from multiple new environments.

1 Introduction

The power to simulate the dynamics of a given environment provides a learning agent the ability to not only re-experience past episodes, but also to generate potentially unseen experiences in preparation for encountering them. This idea has its foundations in model based reinforcement learning, however modern explorations have taken on many forms. One proposed approach seeks to learn an unsupervised temporal prediction model that compresses the complete history of \((\text{state}_t, \text{action}_t, \text{reward}_t, \text{state}_{t+1})\) transitions an agent experiences [Schmidhuber, 2015]. This ‘World Model’ is then provided to the agent as a tool to better inform its decision making process in various ways. Recent explorations of this theoretical framework has shown the approach is feasible at least with a single environment, and has illustrated the potential for using training based on simulated rollouts of possible episodes from the learned World Model in an entirely off-line fashion [Ha and Schmidhuber, 2018].

One critical aspect of this framework, which has yet to be fully addressed, is the need to continually learn in the potentially very different domains of the environment the agent is experiencing. Particularly when using neural networks, the World Model learned in this fashion would be subject to catastrophic forgetting when an incomplete history of all previous transitions are stored (which inevitably would be the case in a true lifelong learning scenario). Schmidhuber [2015] suggests some ideas for how to address this, however they remain under-specified and untested to date.

Modern approaches to the catastrophic forgetting problem propose to learn various forms of plasticity parameters such that once a set of weights are learned for a given distribution of input samples (e.g., a particular task) those weights are made static as new distributions of input samples arrive [Kirkpatrick et al., 2017; Zenke et al., 2017]. Intrinsic to these approaches is the necessity to segment and label collections of input samples into discrete sets or ‘tasks’. This is both difficult to do in the continuous flow of experience, and undesirably inflexible given the potential benefit of transferring learned knowledge from one task to another.

Previous work in psychology and neuroscience have developed theories for how mammalian brains can perform this type of continual learning. One of the more prominent computational theories is the idea of Complimentary Learning Systems [McClelland et al., 1995; O’Reilly et al., 2014], which posits that recent experiences are ‘replayed’ during both sleep and quiet rest. This allows the brain, or learning system, to interleave previous experiences so they can be slowly integrated into a more comprehensive and robust internal representation. This in-turn inspired connectionist models that used a form of this replay, referred to as ‘pseudo-rehearsal’, to preserve previous learned weights [Robins, 1995; Ans et al., 2004]. The utility of such a mechanism to modern learning systems has been previously posited [Kumaran et al., 2016], however the limit and scope of its applications have yet to be fully understood.

Here we explore a potential solution to preserve the un-
supervised learning of a World Model like architecture by adapting the strategy of pseudo-rehearsal [Robins, 1995; Ans et al., 2004]. Fundamentally this is accomplished by having the World Model generate rollouts of previous experiences that are then interleaved with new experiences, thereby making the input distribution reflect both past and present experiences. Critically, we show this method capable of preserving the previously learned experiences without the need for segmenting experience into discrete tasks. To do this we use a published version of the World Model architecture trained in sequence to capture the dynamics of a set of Atari 2600 games.

The major contributions of this paper are summarized as:

- Illustrating pseudo-rehearsal as a potential label-free approach to continual learning in pixel-based environments
- Providing foundations for continual learning in model-based reinforcement learning
- Exploring the relationship between proportion of interleaved pseudo-samples and performance retention

2 Related Work

The original ideas surrounding pseudo-rehearsal were first developed within a relatively simple connectionist framework that relied on input and output to be sparse, low dimensional binary patterns. In general, these methods used random input patterns to generate corresponding output patterns to create a set of input/output pairs that can be interleaved with new samples to preserve the current state of the network [Robins, 1995]. Later work explored the idea of capturing a whole sequence of experiences with a single pseudo-sample [Ans et al., 2004]. These methods were both inspired by and inspiring to theories of how biological systems continually

learn, namely the Complimentary Learning Systems theory. This theory posits that the hippocampus replays recent experiences so that the slower learning neo-cortex can consolidate that information into a more stable and robust representation [O’Reilly et al., 2014].

This idea of replay based preservation of past experiences has taken root in recent work looking to accomplish the goal of sequentially learning multiple datasets. Most of this has been focused on classification tasks, and uses a form of pseudo-rehearsal referred to as ‘deep generative replay’ [Shin et al., 2017]. The basic approach is the same, however the input/output patterns can be non-sparse continuous values, including pixel-based representations. The architectures used is a generative model of some type and is usually a Generative Adversarial Network [Goodfellow et al., 2014]. Pushing this approach further, and taking inspiration from the Complimentary Learning Systems framework, a dual memory system used a form of deep generative replay for continual learning on a digits classification tasks [Kamra et al., 2017]. Notably, this implementation showed increased accuracy and backward transfer compared to Elastic Weight Consolidation, however the apparent difference in tasks tested was relatively small compared to the arbitrary set of Atari games used in the current work.

The Compress and Progress framework [Schwarz et al., 2018] can continually train reinforcement learning agents in a series of complex pixel-based environments using a more scalable form of Elastic Weight Consolidation [Kirkpatrick et al., 2017] along with Policy Distillation [Hinton et al., 2015] to iteratively transfer learned policies into a single network. As mentioned above these plasticity based preservation methods require task labels to avoid catastrophic forgetting. Recent work, referred to as the RePR model (Reinforcement-Pseudo-Rehearsal), has shown that the pseudo-rehearsal approach is also viable in reinforcement learning [Atkinson et al., 2018]. Using a GAN based generative architecture paired with a modified DQN this approach used pseudo-rehearsal to iteratively maintain a single generative network trained on self-generated pseudo-samples and samples from an expert generative network trained on a specific task. For the integrated policy, similar to Progress and Compress [Schwarz et al., 2018], a single agent network was learned by using Policy Distillation to generate target output action distributions from both itself and an agent network trained on a specific task. This approach is very promising for iterative training in reinforcement learning, however, it segments experience into tasks to train expert generative agent networks, and can not provide temporally connected samples essential for n-step gradient based learning, e.g., backprop through time.

Within the World Model framework there has been a couple recent works that illustrate its effectiveness within reinforcement learning. In particular, Ha and Schmidhuber [2018] showed that this framework could be used, within a limited set of environments, to train evolved neural networks to achieve state of the art performance even when trained entirely on simulated rollouts. More recently, an approach that similarly learns a latent World Model used to inform action selection was able to learn simultaneously in 6 separate environments provided there were interleaved with each other.
Algorithm 1 Sequential Learning Using World Model Pseudo-Rehearsal

**Input:** $E$ set of potential environments, hyperparameters $n_{RealRollouts}, n_{SimRollouts}$

**Parameter:** network parameters $V$ and $M$

1. Random initialization of $V, M$, network parameters, initialize $M_{t-1}$ as empty
2. while error in $V$ is decreasing do
3.   optimize $V$ on random samples from all $E$ environments
4. end while
5. for $e_i$ in $E$ do
6.   while loss in $M_i$ is decreasing do
7.     optimize $M_i$ on $n_{RealRollouts}$ drawn from $e_i$ using random actions
8.   if $M_{t-1}$ is not empty then
9.     seed $M_{t-1}$ with $n_{SimRollouts}$ random points in latent space
10. optimize $M$ on $n_{SimRollouts}$ drawn from $M_{t-1}$
11. end if
12. end while
13. $M_{t-1} = M_t$
14. end for

[Hafner et al., 2018].

3 Continual Learning of World Models

Our main focus in this work is testing the hypothesis that interleaving pseudo-rehearsal based replays can be sufficient to preserve the learned knowledge within a World Models framework. Similar to Ha and Schmidhuber [2018], a perceptual Variational Auto Encoder ($V$) is trained to compress a high dimensional input (e.g. images) into a smaller latent space ($z$) while also allowing for a reconstruction of that latent space back into the high dimensional space. This latent space representation is then fed into a temporal prediction network ($M$) that is trained to predict one time step into the future.

The $V$ network learns to both encode and reconstruct observed samples into a latent embedding by optimizing a combination of reconstruction error of the samples from the embedding back into the original observation space, and the KL Divergence of the samples from the prior ($\mathcal{N}(\mu = 0, \sigma = 1)$) on the embedding space those samples are encoding into; this framework is generally known as a Variational Auto-Encoder (VAE) [Kingma and Welling, 2013]. In this work we use a convolutional VAE with the same architecture as Ha and Schmidhuber[2018], where the input image is passed through 4 convolutional layers ($32, 64, 128,$ and $256$ filters, respectively) each with a 4x4 weight kernel and a stride of $2$. Finally the output of the convolution layers is passed through a fully connected linear layer onto a mean and standard deviation value for each of the dimensions of the latent space, which is used to then sample from that space. For reconstruction a set of deconvolution layers mirroring the convolution layers takes this latent representation as input and produces an output in the same dimensions as the original input. All activation functions are rectified linear except the last layer which uses a sigmoid to constrain the activation between 0 and 1.

The $M$ network, also based on Ha and Schmihuber [2018], takes the latent space observation and passes it through a 256 unit LSTM layer [Hochreiter and Schmidhuber, 1997]. The LSTM output is then concatenated with the current action as input to a Mixture Density Network [Bishop, 1994], which passes the input through a linear layer onto an output representation that is the mean ($\mu$) and standard deviation ($\sigma$) used to determine a specific Normal distribution ($\mathcal{N}$), and a set of $G$ mixture parameters ($\Pi$) used to weight those separate distributions in each of $V$’s latent space dimensions. Two additionally output units are present, one for predicted reward and the other for the predicted episode termination probability. In total that is $3GL + 2$ output units, where $L$ is the size of the latent space.

Optimization of the LSTM and MDN is done comparing the output of the MDN to a $V$ encoded real observation ($z_{t+1}$) from 1 time step into the future predicated on the chosen action ($a_t$). Specifically, the combined LSTM (with hidden state $h_t$) and MDN, referred to as the $M$ network, models each dimension (indexed by $l$) of the next latent state as the weighted sum of $G$ Gaussians: $P_t(z_{t+1}|x = (z_t, a_t, h_t)) = \sum^G_{g=1} \Pi_g(x)(\mu_g(x), \sigma_g(x))$. Output units corresponding to each $\Pi_l$ set are normalized using a softmax, and each $\sigma_{g,l}$ unit is passed through an exponential function to ensure they are appropriate for the mixture model. This set of Gaussian mixture models can be optimized using the average log likelihood over latent dimensions as the loss function: $L_{GM}(\tilde{z}_{t+1}, x) = 1/L \sum^{D}_{l} - \log(\sum^{G}_{g=1} \Pi_g(x)\mathcal{N}(\mu_g(x), \sigma_g(x)))$. Additionally, reward and terminal state units are optimized using mean squared error ($L_{MSE}$) and binary cross entropy ($L_{BCE}$), respectively. Finally, the total loss optimized by the $M$ network is the average of the three losses $L_{total} = \frac{1}{3}(L_{MSE} + L_{BCE} + L_{GM})$.

The training of these two models is usually done in sequence ($V$ then $M$) and is entirely unsupervised (i.e. no label data is required). This framework allows for an online simulation of experiences as the $M$ network can predict one time step into the future, and can use it’s own predictions as the seed for the subsequent predictions provided some action. Sampling a specific prediction $\tilde{z}_{t+1}$ from the distribution determined by the $M$ network’s output is done by drawing a specific $g$ from each of the $\Pi_l$ distributions, which then determines a $\mathcal{N}(\mu_{g,l}, \sigma_{g,l})$ value to be used in the $l$th dimension of $\tilde{z}_{t+1}$. The samples from these simulations are used in this work as pseudo-samples, approximating the original experience, and interleaved with the real samples from the environment (see Figure 1). Similarly, the $V$ network can decode these $\tilde{z}_{t+1}$ predictions into the original input space allowing for full simulated rollouts of experiences in either the original input space or in the learned latent embedding, however, in this work pseudo-sample rollouts were confined to the latent space for efficiency.

Sequential learning can be done in the $M$ network by first
learning/providing an auto-encoder, $V$, that can embed high-dimensional samples from all potential environments into a sufficiently low-dimensional space. If the input dimensions are already sufficiently small this embedding may potentially be unnecessary, however it is as yet unclear what a sufficiently low-dimensional space would be. The first $M$ network is optimized on ‘task0’, with no interleaved pseudo-samples. After training has reached some criteria, this $M$ network is made static as $M_{i-1}$, so it can be used to generate pseudo-samples to be interleaved, batch for batch, with embedded samples from the real environment.

In the current experiments, preservation of the $M$ network was done only when a new environment is introduced, however within task iterations of $M$ may also prove beneficial. The optimal schedule for $M$ network preservation is yet to be determined, and is left for future work. The basic training algorithm for sequentially learning in this framework is outlined in Algorithm 1.

4 Experiments

Testing the sequential learning of the $M$ network was done by first generating 1000 rollouts from all potential tasks, which in this case were a set of 13 Atari games. Each rollout was generated using a series of randomly sampled actions with a minimum duration of 100 and maximum duration of 1000 samples. The first 900 of these rollouts, for each game, were used for training data and the last 100 are reserved for testing. All image observations were reduced to 64x64x3 and rescaled from 0 to 1, and all games were limited to a 6 dimensional action space: “NOOP”, “FIRE”, “UP”, “RIGHT”, “LEFT” and “DOWN”. Each game is run through the Arcade Learning Environment (ALE) and interfaced through the OpenAI Gym [Bellemare et al., 2013; Brockman et al., 2016]. All rewards were clipped as either -1, 0, or 1 based on the sign of the reward, the terminal states were labeled in reference to the ALE game-over signal, and a non-stochastic frame-skipping value of 4 was used.

All training images are then fully interleaved to train a VAE ($V$ network) that can encode into and decode out of a 32 dimensional latent space. Training was done using a batch size of 32 and allowed to continue until 30 epochs of 100,000 samples showed no decrease in test loss greater than $10^{-4}$. The learned embedding for this VAE can be seen in Figure 4.

Using this $V$ network to encode the original rollouts into the latent space, the $M$ network is then trained in sequence on the set of potential tasks. Here training is done using rollouts of length 32 in batches of 16, and is allowed to continue for up to 30 epochs of 100,000 samples unless 10 epochs show no decrease in test loss greater than $10^{-4}$. The weights saved on the epoch with the lowest test loss is used as the initialization for training in the next task.

After training, a set of simulated rollouts, or pseudo-samples, are generated by seeding the $M$ network with a random point in the latent space (each dimension sampled from the Normal Distribution with mean 0 and standard deviation 1), and allowed to rollout a simulated trajectory using its learned model of the environment. Here the prediction for the current step is used as the input for the next step, and continues in this way until a ‘done’ (game-over) state is predicted, or until 1000 time steps have been generated. These generated samples are then interleaved with the next task’s training set. Here, first a batch of 16 rollouts of 32 samples from the real environment provided gradient updates, and then a batch of 16 rollouts of 32 samples provided gradient updates from the pseudo-samples. Training duration and stopping criteria are not altered except that the loss used to determine stopping is now the sum of test losses across the real and pseudo-sample rollouts.

A baseline measure of catastrophic forgetting was established by performing the same training as described above with no pseudo-samples interleaved. Similarly, we only trained the $M$ network on a subset of the total tasks presented to the $V$ network to allow for the rollouts to potentially drift into other areas of the latent space occupied by untrained tasks.

4.1 Results

Performance (here the loss function optimized during training) was assessed on a held out test set of rollouts for each task. This assessment was done on all previous tasks after training was complete for a given task, and was normalized by the minimum loss achieved for each respective task, i.e. the loss after initial training within a given task. In this way a measure of the degradation of performance as a function of learned tasks was achieved. Figure 2 shows these performance curves using 6 very different Atari games as tasks. Solid lines show performance when simulated rollouts were interleaved during training, and dashed lines show performance when no interleaving of simulated rollouts occurred. Clear catastrophic forgetting is seen in the non-interleaved case while relatively little reduction in loss is observed when simulated rollouts were interleaved. The small plot in Figure 2 titled ‘loss over tasks’ zooms in on the unnormalized loss when using pseudo-rehearsal and illustrates the relatively small accumulation of increased loss over tasks. Similarly, Figure 2 plot title ‘task0 loss curve for each task’ shows the
Figure 3: Reconstruction of test rollouts from task0 using $M$ networks after training on a given task. Left grid shows simulated rollouts when no pseudo-rehearsal was used in training, right grid shows simulated rollouts when trained with pseudo-rehearsal, and far right column shows real rollout from the environment. Grid rows correspond to rollout images spaced 4 time steps (actual rollouts are spaced a single time step, 4 is used here for brevity), columns are rollouts generated after each round of sequential training on a given task.

Figure 3, reconstructions of test rollouts from the first trained environment (i.e., task0) are shown across sequential learning iterations. This figure provides a heuristic for translating the change in loss observed in Figure 2 into appreciable samples. Clear signs of catastrophic forgetting are seen in the reconstructed samples when pseudo-samples are not interleaved which translates to roughly 10x increase in loss compared to initial training. Similarly, a measurable amount of increased loss is seen even when using a pseudo-rehearsal, shown in Figure 2 ‘loss over tasks’ to be on the order of 10% over 5 tasks. The impact of this increased loss is seen in Figure 3 to be relatively unintrusive when reconstructed into the original input space. This increase does illustrate the potential for accumulating loss when scaling this method to 100s of tasks. However, on average across tasks and training iterations, an increase in loss of 4.52 $\pm$ 1.35 times the original loss is witnessed when not using pseudo-rehearsal (i.e. the average pairwise difference of without-pseudo-rehearsal minus with-pseudo-rehearsal loss measures in the main plot of Figure 2).

The simulated rollouts for each iteration of sequential training are shown in Figure 4. Here the latent space is visualized in 2 dimensions using a UMAP embedding [McInnes et al., 2018] derived from a random set rollouts totaling 10,000 samples from each task. Using this embedding a K-Nearest Neighbor (KNN) classifier [Pedregosa et al., 2011] is trained using the $V$ embedding of the same data, and then used to estimate a task label for pseudo-samples, which themselves are a random set totaling 10,000 samples taken from the rollouts interleaved during the experiment. As can be seen in the bottom plot of Figure 4 the pseudo-samples generated stay mostly within and distribute relatively well across trained tasks, even with no explicit task labels provided. This sampling, however, clearly does deviate from uniform, and one consequence of obscuring task labels during training is that forcing a particular sampling over previous tasks becomes more difficult. There was a significant relationship between percentage of pseudo-samples generated within a given task and the increase in loss within a iteration of the sequential learning experiment (Spearman rank correlation $r = -0.58$, $p = 0.02$), suggesting that the more pseudo-samples used within a given task the smaller the increase in error. One potential improvement from the current approach would be finding methods to enforce a smarter pseudo-sampling, however what the optimal sampling should be is an active area of research.

4.2 Reinforcement Learning

A short validation using the trained $V$ and $M$ networks as input to a A2C agent was done to assess the stability of the learned representations over sequential learning. The intuition behind this analysis is an agent trained on the concatenated output of a pair of $V$ and $M$ networks will assume some distribution of inputs in order to achieve some average return, i.e. the agent learns a policy and value function that is specific to the input patterns it was trained on. If those input patterns change, then it is likely the average episode return would decrease due to the agent being untrained on the current pattern of inputs. Here a PyTorch implementation of A2C [Shang tong, 2018] was modified to train an agent using the latent and hidden states ($z$ and $h$ from Figure 1) of the World Model. First an agent was trained for 20 million steps using the $M$ network trained only on task0, i.e. $M_0$. Then, average returns over 30 episodes were used to assess that agent’s per-
performance in each of the trained $M$ networks, i.e., $M_0$ to $M_5$. Results from this validation showed no deviation of average episode returns when using $M$ networks sequentially training from task1 through task5 (i.e., $M_1$, $M_2$, $M_3$, $M_4$, and $M_5$) compared to the original $M_0$ network the agent was trained on. This suggests that pseudo-rehearsal not only preserved the original loss in the $M$ network, but also sufficiently maintained the same representations such that the trained agent could rely on them through sequential training.

5 Conclusion and Future Work

Here we have shown that pseudo-rehearsal based methods are capable of supporting continual learning within the World Model framework. The scalability of this approach is not likely to exceed 10s of tasks given the current implementation and results. The size of the networks used would require augmentation if no learning facilitation becomes evident when training on a larger set of tasks. This augmentation process is non-trivial without destroying existing information stored in the network. A more sophisticated method of sampling from previous tasks would help improve retention of previous tasks and speed up training time.

In this work we assumed a latent representation that is capable of encoding and decoding the features of new tasks. Preliminary experiments investigating continual learning in the VAE have proven difficult as each iteration rearranges the latent space, destroying the learning in the $M$. Building in a prior to the loss function that preserve the already existing latent space is likely a promising approach.

Although it is indeed true our approach requires no explicit labeling of tasks our training scheme inherently makes use of them by preserving the world model when switching tasks. More work is required to establish how closely this preservation cycle needs to be related to task switches, and if there is any benefit of preservation within an ostensibly pre-defined task such as an Atari game.

Finally, the current framework could be used to drive a trained agent to continually collect more diverse samples than are possible given random actions, which can be used to further train the internal World Model. Continual learning within the agent itself provides its own challenges, however iterative Policy Distillation [Schwarz et al., 2018; Atkinson et al., 2018] and recent approaches relying entirely on the World Model for action selection [Hafner et al., 2018] are good candidate solutions.

Acknowledgments

We thank James McClelland, Amarjot Singh, Charles Martin, and Mohammad Rostami for helpful feedback in the development and analysis of this work, and Jeffrey Krichmar, Emre Neftci, Risto Miikkulainen, Andrea Soltoggio, and Jean-Baptiste Mouret for conceptual discussions surrounding the work. This material is based upon work supported by the United States Air Force & DARPA under contract no. FA8750-18-C-0103. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA.
References

[Ans et al., 2004] Bernard Ans, Stéphane Rousset, Robert M French, and Serban Musca. Self-refreshing memory in artificial neural networks: Learning temporal sequences without catastrophic forgetting. Connection Science, 16(2):71–99, 2004.

[Atkinson et al., 2018] Craig Atkinson, Brendan McCane, Lech Szymanski, and Anthony Robins. Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting. arXiv preprint arXiv:1812.02464, 2018.

[Bellemare et al., 2013] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, Jun 2013.

[Bishop, 1994] Christopher M. Bishop. Mixture density networks. Technical report, Citeseer, 1994.

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

[Goodfellow et al., 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[Ha and Schmidhuber, 2018] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Advances in Neural Information Processing Systems, pages 2455–2467, 2018.

[Hafner et al., 2018] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. arXiv preprint arXiv:1811.04551, 2018.

[Hinton et al., 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[Kamra et al., 2017] Nitin Kamra, Umang Gupta, and Yan Liu. Deep generative dual memory network for continual learning. arXiv preprint arXiv:1710.10368, 2017.

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

[Kirkpatrick et al., 2017] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, page 201611835, 2017.

[Kumaran et al., 2016] Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. Trends in cognitive sciences, 20(7):512–534, 2016.

[McClelland et al., 1995] James L McClelland, Bruce L. McNaughton, and Randall C’O’reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological review, 102(3):419, 1995.

[McInnes et al., 2018] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. The Journal of Open Source Software, 3(29):861, 2018.

[O’Reilly et al., 2014] Randall C O’Reilly, Rajan Bhattacharya, Michael D Howard, and Nicholas Ketz. Complementary learning systems. Cognitive science, 38(6):1229–1248, 2014.

[Pedregosa et al., 2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[Robins, 1995] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. Connection Science, 7(2):123–146, 1995.

[Schmidhuber, 2015] Jürgen Schmidhuber. On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. arXiv preprint arXiv:1511.09249, 2015.

[Schwarz et al., 2018] Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabiska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. arXiv preprint arXiv:1805.06370, 2018.

[Shangtong et al., 2011] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. The Journal of Open Source Software, 3(29):861, 2018.

[Shin et al., 2017] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In Advances in Neural Information Processing Systems, pages 2990–2999, 2017.

[Zenke et al., 2016] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. arXiv preprint arXiv:1703.04200, 2017.