Recurrent Predictive Processing For Image Classification

Alex Hanson, Ryen Krusinga, Julian Vanecek
{hanson,krusinga,jsv}@cs.umd.edu

***Abstract*—We propose a simple recurrent image classification model based on the paper "Recurrent Models of Visual Attention" [1]. The model uses a recurrent neural network to look around an image, extracting small attention windows called "glimpses," which it ultimately uses to make a classification. Our main contribution is the addition of a module which predicts future glimpses based on the current hidden state of the attention model. We show that a model trained using nothing but prediction loss can learn informative embeddings of MNIST digits. We then show that training a model with prediction loss and classification loss can outperform a model trained only with classification loss, at the task of classification. However, when stochasticity is added to the glimpses along with a reinforcement learning loss, the original model outperforms ours significantly.**

I. Introduction

Recently, several schools of thought in artificial intelligence and neuroscience have been converging on the same principle: to learn good unsupervised representations of a domain, you should predict future percepts. We will refer to this idea broadly as *predictive processing*. This is a common theme in modern reinforcement learning models, in which rewards are often related to the ability of a model to predict the future consequences of its actions[2] [3]. This is closely associated with the concept of attention modeling [4] [5], where agents learn to direct their information-gathering actions towards salient parts of the environment. Building off of the work of Minh [1] on image classification via recurrent visual attention, we hypothesized that attention-based supervised learning models can take advantage of predictive processing to more efficiently learn feature representations for classification.

II. Background

Ideas about predictive processing have been around much longer than modern deep reinforcement learning, in other fields and guises. In neuroscience and philosophy, the Predictive Processing Model [6] [7] [8] [9] suggests that the brain is continually making top-down predictions in order to inhibit its future perceptual inputs. Actions are viewed as a special kind of prediction that directly affects the world by changing the agent's pose and thus its next perceptual inputs. In other words, actions are self-fulfilling prophecies that make their own predictions come true. Prediction and input inhibition as a method of model learning have their roots in early control theory and cybernetics [10] [11] [12].

In traditional machine learning, the concept most closely associated with predictive processing is that of *autoencoding*: learning representations of data by squeezing them through a low-dimensional bottleneck and then attempting to reconstruct the original as closely as possible [13] [14]. This may be regarded as a special case of predictive processing in which the model tries to predict its current input only. Autoencoders are frequently used for tasks such as automated feature extraction [13]. Extracted features can then be used as inputs to other models, such as classifiers.

Later, machine learning borrowed the idea of attention from neuroscience. It has been shown

that models of salience can be added to supervised neural networks to highlight the portions of an image informing the classification [15]. Perhaps the most natural model of attention lies in recurrent neural networks, which can be used to drive percept windows around images like the saccades of a biological eye [16] [1] [17] [18]. We use a recurrent neural network ([1]) that produces actions corresponding to saccades of attention, and which eventually uses what it has discovered to make a classification.

III. Review of Reinforcement Learning

Reinforcement learning introduces the concept of *actions* and *rewards*. Actions are things that the agent can do to influence its environment, its future percepts, and its future rewards. Rewards are numerical signals r_i that inform the agent whether it is performing well or poorly at its task, with larger rewards corresponding to better performance. The goal of a reinforcement learning agent is to maximize its expected future reward

$$\mathbb{E}\left[R_t\right] = \mathbb{E}\left[\sum_{i=t}^{T} r_i\right]$$

from the current time t onward to the final time T, which may be finite or infinite. Future rewards are sometimes discounted with a multiplier γ^{i-t}, $\gamma \in (0, 1)$, that biases the agent more towards the present and eliminates convergence problems with infinite time horizons. Due to the short time horizons in this paper, we will only consider undiscounted rewards.

In timestep t, a recurrent reinforcement learning model obtains perceptual input x_t, generates action a_t, and receives reward r_t. Rewards are often sparse, occurring only after some complex goal with multiple steps has been completed. Examples of rewards could include game score in the case of Atari-playing agents [19] or correctness of image classification in the case of supervised models [1].

We argue that in reinforcement learning for classification, there is a natural kind of reward signal that could be exploited, namely, percept prediction accuracy. We hypothesize that an agent that is able to predict what it will see next has a strong model of its environment, and thus is better positioned to make a classification.

IV. Model

We take the RNN model from "Recurrent Models of Visual Attention" [1] and modify it to make predictions about future percepts. Our high level recurrent architecture is depicted in Figure 6. At each timestep t, an action a_t is emitted based on the error e_{t-1} from the previous step and the previous hidden state h_{t-1}. This action is a two-dimensional vector which dictates the next percept x_t by specifying the coordinates where the vision window should be centered, normalized to the interval $[-1, 1]$ via a tanh nonlinearity. The hidden state h_t is then computed from e_{t-1}, h_{t-1}, and a_t, then a prediction p_t is emitted based on h_t. Then the error e_t is computed by subtracting the predicted attention window from the actual window. At the end of a run, a final classifier makes a prediction of the class based on the hidden state. These update steps are represented in equations 1-4. The function f produces action vectors, g updates the hidden state, and q produces predictions. All functions are neural networks.

$$a_t = f(e_{t-1}, h_{t-1}) \tag{1}$$

$$h_t = g(e_{t-1}, h_{t-1}, a_t) \tag{2}$$

$$p_t = q(h_t) \tag{3}$$

$$e_t = x_t - p_t \tag{4}$$

We used several different loss functions. For classification, a negative log likelihood loss was used. On each prediction error step, a mean squared error loss was used. For actions, a reinforcement learning loss can optionally be used. If the reinforcement loss is not used, the actions are learned by BPTT via g and f (equations 1 and 2). We refer to this as the **deterministic** model, because the actions are not randomly sampled by the network. If

some random sampling is added, we refer to the model as **stochastic** and we add the REINFORCE loss expression as given the original paper [1]. This increases the log probability of actions with good rewards, as shown in expression 5.

$$\frac{1}{M}\sum_{i=1}^{M}\sum_{t=1}^{T}\nabla_\theta(\log \Pr(a_t|h_{t-1}))(R_t^i - b_t^i) \quad (5)$$

The value R_t^i is the cumulative reward from data point i at timestep t and later, while the value b_t^i is a baseline reward prediction by a regressor w, so that $b_t = w(e_{t-1}, h_{t-1})$ is the predicted future cumulative reward from the current timestep onward. The w network is trained with a mean squared error on the actual value of R_t^i, and is detached from the recurrence so that the gradient values are not backpropagated through time. The probability of an action a_t given h_{t-1} is the density function value of a gaussian centered around $f(e_{t-1}, h_{t-1})$ with a fixed standard deviation of 0.17.

A. Glimpses

We refer to the locations of the attention window over time as *glimpses*. Figures 1, 2, 3, and 4 show examples of glimpses over time on various images. Glimpses consist of a small 8x8 *center vision* window, with some optional number of additional peripheral vision windows surrounding it. The peripheral vision windows double in size going outward, but are then downsampled via average pooling to be the same size as the center vision window. This simulates the blurriness of peripheral vision. All vision windows are concatenated into one vector before being processed. If any window goes beyond the image boundaries, the out-of-bounds portion is padded with zeros. Figure 5 shows some example predictions of future glimpses made by the model. In the models trained on larger 60x60 images, we used an 8x8 center vision window with two extra peripheral windows. In the model trained on standard 28x28 MNIST, we used just one peripheral window.

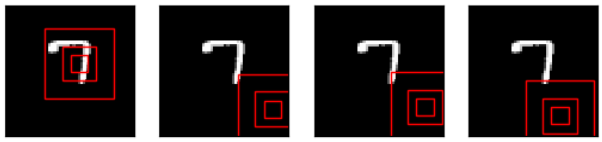

Fig. 1. The locations of glimpses for the original model, on a single translated MNIST digit.

For most of our experiments, the number of glimpses was fixed at 4, somewhat arbitrarily. For the first recurrent step of the model, no prior glimpse is available, and both the hidden state h_0 and the initial error e_0 are initialized to zeros. The initial action a_1 taken by the network can be interpreted as the part of the image that the network has learned is the best place to start looking.

B. Variants of backpropagation

For training, we used the Adam optimizer with an initial learning rate of 0.0003. We experimented with several methods of propagating errors from different parts of the model. The MSE loss gradients for the glimpse predictions p_t can either be detached from the rest of the model so that they don't influence the recurrent model, or included in the backpropagation through time. We decided that the prediction errors should propagate through time, as this is fundamental to the idea that prediction should affect how the recurrent model builds actions and represents information.

By contrast, we kept the reinforcement learning loss on the stochastic action model detached from the rest of the model, as this is what the original paper did. In both the deterministic and stochastic models, actions are fed into the recurrence, and are influenced by the prediction error loss and negative log likelihood classification loss.

V. Experiments

We performed a number of experiments to figure out the best combinations of losses,

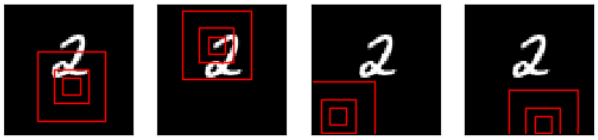

Fig. 2. The locations of glimpses for our model, on a single translated MNIST digit.

Fig. 3. The locations of glimpses for the original model, on a single translated and cluttered MNIST digit.

architectures, glimpse numbers, and so forth, not all of which we show in this paper due to uninteresting results. One such experiment was testing the effect of using an LSTM module as opposed to the original non-LSTM recurrent architecture. We found that the LSTM did not make much of a difference in this case, possibly because our network is not trained for very many recurrent steps. We also tried convolutional layers as opposed to fully connected layers for making glimpse predictions and glimpse embeddings. However, this also did not noticeably affect performance; the glimpse windows are small enough for fully connected layers to work just fine. Additionally, we tested various numbers of glimpses per data point, and various numbers of peripheral vision windows. More glimpses and more windows improved performance, sometimes so much that different models were indistinguishable in performance after training; we ultimately picked parameters that illustrated divergences in model performance.

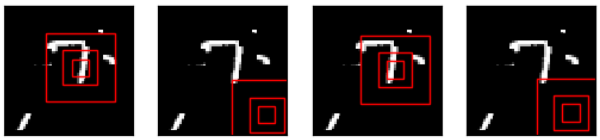

Fig. 4. The locations of glimpses for our model, on a single translated and cluttered MNIST digit.

Fig. 5. Examples of center vision window predictions.

As a sanity check experiment, we tested whether or not purely predictive models learn anything at all about their inputs. To do this, we trained a slightly altered model on standard MNIST, with only mean squared error loss and no classification loss or reinforcment learning loss. Instead of the network producing actions on its own, glimpse window centers in this case were sampled uniformly at random from the image, and then fed into the network along with the corresponding coordinates. Six glimpses were used in total. In order to reduce prediction error, this network needed to learn some informative representation of the input image. Figure 7 shows that over time, the network is indeed able to reduce its loss function; we conclude that it is indeed learning something about the input image that allows it to predict future percepts. We then used the trained network to extract embeddings of MNIST digits by running them with random actions and using the last hidden state h_T as the embedding. We were able to train a second network on these embeddings to predict the MNIST digit classes with 90% test accuracy. This shows that the hidden states of a trained pure predictive network represent something meaningful about the input.

Our next four experiments were designed to show the differences between our model and the original model, for both the deterministic and stochastic variations, on translated MNIST and translated MNIST with clutter. The translated MNIST dataset was produced by placing each MNIST digit in larger 60x60 window and randomly translating it. The translated MNIST with cluttered overlays randomly cropped 8x8 blocks from other MNIST digits on the translated MNIST dataset.

Figures 8 and 9 compare the original deter-

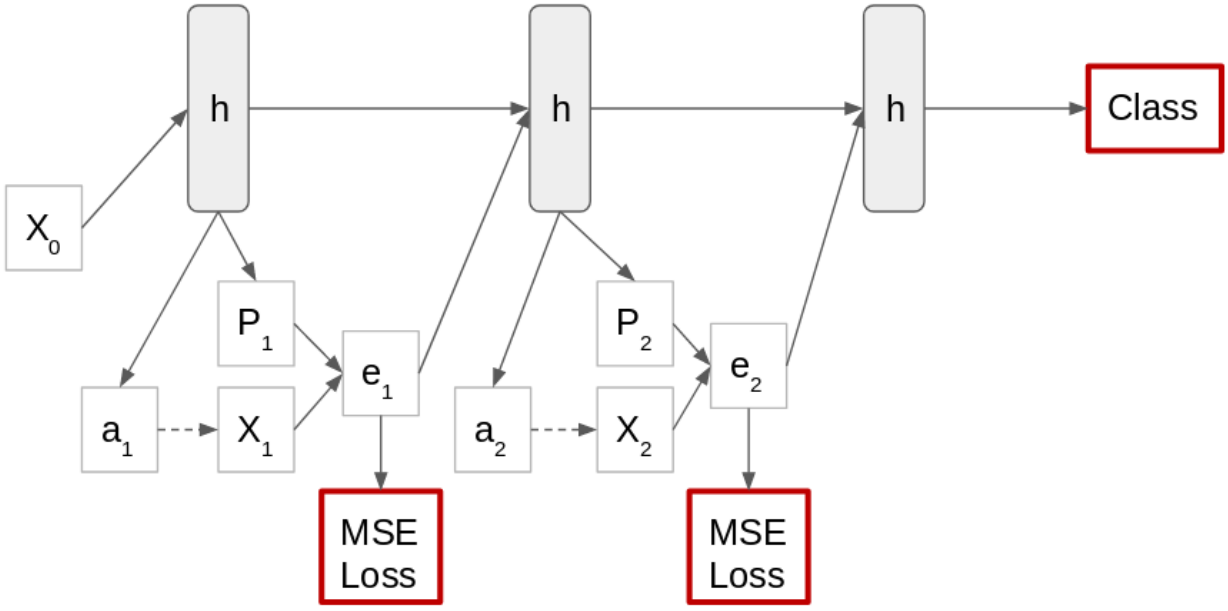

Fig. 6. Recurrent predictive processing model. Each timestep emits an action a_t and a prediction p_t. Prediction error is fed as input into the next timestep, as well as an embedding of the action a_t (not shown in the diagram).

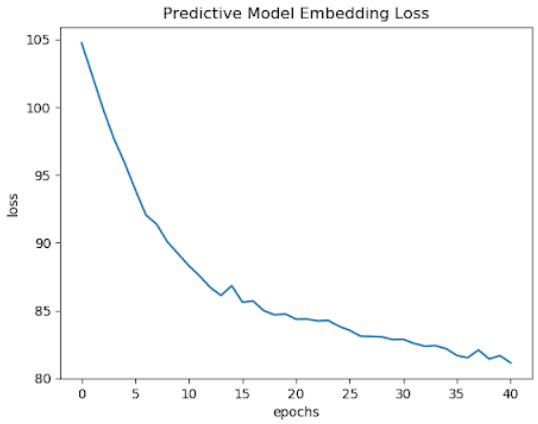

Fig. 7. Validation loss curve for a deterministic model trained on 28x28 MNIST with pure prediction loss and no classification loss.

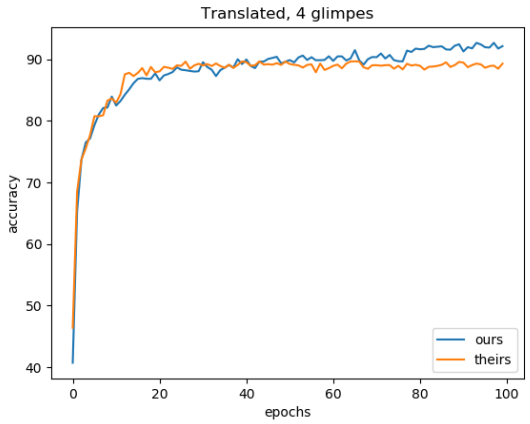

Fig. 8. Comparison of our deterministic model and original deterministic model validation accuracy on 60x60 translated MNIST.

ministic model with our model on the translated MNIST and the translated MNIST with clutter respectively. The original deterministic model glimpses around the image and then makes a classification, and is only trained with negative log likelihood (NLL) loss. Our model does the same, but uses prediction error instead of glimpses directly, and incurs a mean-squared-error prediction loss at every timestep, which is backpropagated through time along with the NLL loss. The rationale for propagating the prediction error through time is that the hidden state update to the model should be forced to learn structures that foster long term predictions. In both figures, our predictive model outperforms the original model by a

Fig. 9. Comparison of our deterministic model and original deterministic model validation accuracy on 60x60 translated and cluttered MNIST.

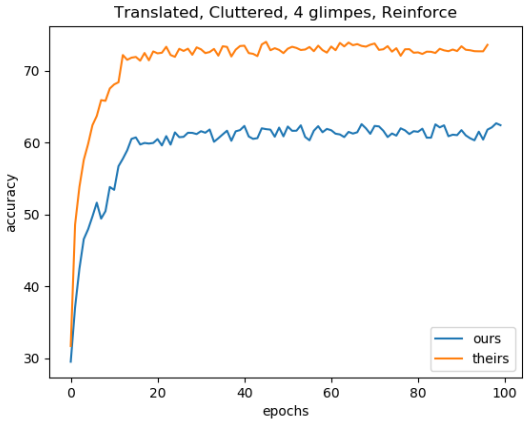

Fig. 11. Comparison of our stochastic model and the original stochastic model validation accuracy on 60x60 translated and cluttered MNIST.

Fig. 10. Comparison of our stochastic model and original stochastic model validation accuracy on 60x60 translated MNIST.

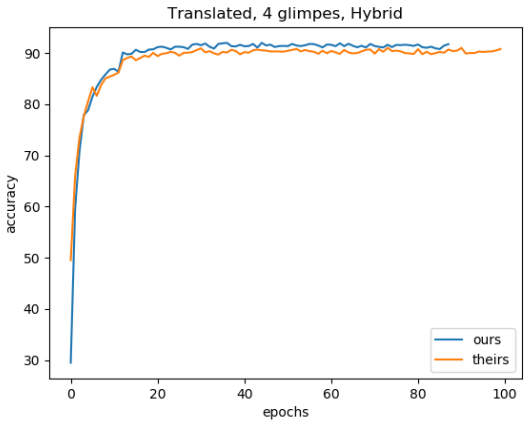

Fig. 12. Comparison of our hybrid model and original hybrid model validation accuracy on 60x60 translated MNIST.

few percentage points in terms of held out validation accuracy after 100 epochs. In figure 9 especially, the difference is clear; prediction seems to help deal with the clutter in this case. Further experimentation is needed to see if this superiority is robust.

Figures 10 and 11 show the original model (with reinforcement learning loss) vs. our stochastic model with reinforcements corresponding to negative prediction error rather than classification accuracy. The action model in this case is the same as that for the deterministic model, except that after being generated, small gaussian noise is added to the actions before performing them. This randomness allows the REINFORCE algorithm to calculate the log likelihood of actual actions given intended actions, and then increase the future probability of the actual actions when they corresponded to a good reward. In both figures, our model significantly underperforms the original model. This is likely because randomness of action affects ability to make accurate predictions. It is also quite possible that the negative prediction error rewards or network hyperparameters were not tuned well. We believe that with further

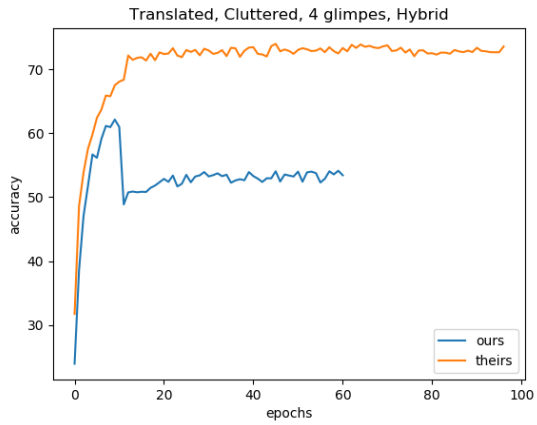

Fig. 13. Comparison of our hybrid model and original hybrid model validation accuracy on 60x60 translated and cluttered MNIST.

work, our model can compare with or exceed the performance of the original.

Figures 12 and 13 show the original model vs. our model with a hybrid REINFORCE loss in which rewards were the sum of prediction accuracy and negative normalized prediction error. In the case of pure tranlated MNIST, the hybrid model seems to slightly outperform the original, though the effect is very small. When clutter is added, our model shoots far down in performance, although we believe that this may be a problem of parameter tuning.

VI. Discussion

Our experiments show that adding prediction to models can indeed improve performance. In particular, a model trained only to predict the outcome of random future glimpses on the same image learns an embedding that contains enough information to classify the image with high accuracy. We have also shown that adding prediction loss can improve classification performance when both are trained together, at least in deterministic cases where actions are chosen without random noise and learned with both negative log likelihood loss and prediction error. However, in the stochastic domain, existing reinforcement learning algorithms outperform ours, but that may be due to a problem of parameter tuning and lack of model sophistication. Our data also suggests that using a hybrid reinforcement learning algorithm with both negative prediction error rewards and classification accuracy rewards can potentially increase performance, but further work is needed.

We surmise that the domain of MNIST digits is not particularly well-suited to this kind of prediction, due to the fact that MNIST images consist mostly of uniform black pixels, with very sharp and high-contrast borders next to the bright MNIST digits. Uniform background patches do not provide information relevant to classification, and an over-abundance of them during training may bias the model towards always predicting that it will see background. A more suitable domain is one in which there is lots of visual information everywhere in the scene suggestive of what exists nearby.

VII. Future Work

In the literature there is little work on recurrent attention models for classification, and there is much room for improvement of our method. In particular, the ability of recurrent attention models to transfer knowledge between datasets, or to perform well on natural images, is still ripe for investigation. The use of more sophisticated recurrent architectures can also be investigated. Given the recent trend of action-conditional-predictive methods in deep reinforcement learning, attention models could benefit from the latest techniques in that field. As human intelligence fundamentally relies on attention over time, we believe that such research is well-worth pursuing.

References

[1] V. Mnih, N. Heess, A. Graves, *et al.*, "Recurrent models of visual attention," in *Advances in neural information processing systems*, pp. 2204–2212, 2014.

[2] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," *arXiv preprint arXiv:1611.01779*, 2016.

[3] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *Advances in neural information processing systems*, pp. 2863–2871, 2015.

[4] S. Minut and S. Mahadevan, "A reinforcement learning model of selective visual attention," in *Proceedings of the fifth international conference on Autonomous agents*, pp. 457–464, ACM, 2001.

[5] J. Ba, R. R. Salakhutdinov, R. B. Grosse, and B. J. Frey, "Learning wake-sleep recurrent attention models," in *Advances in Neural Information Processing Systems*, pp. 2593–2601, 2015.

[6] W. Wiese and T. Metzinger, "Vanilla pp for philosophers: A primer on predictive processing," 2017.

[7] K. Friston, "The free-energy principle: a unified brain theory?," *Nature Reviews Neuroscience*, vol. 11, no. 2, p. 127, 2010.

[8] K. Friston, R. Adams, L. Perrinet, and M. Breakspear, "Perceptions as hypotheses: Saccades as experiments," *Frontiers in psychology*, vol. 3, p. 151, 2012.

[9] A. Clark, *Surfing uncertainty: Prediction, action, and the embodied mind*. Oxford University Press, 2015.

[10] W. R. Ashby, *An introduction to cybernetics*. Chapman & Hall Ltd, 1961.

[11] W. R. Ashby, "Principles of the self-organizing dynamic system," *The Journal of general psychology*, vol. 37, no. 2, pp. 125–128, 1947.

[12] R. C. Conant and W. Ross Ashby, "Every good regulator of a system must be a model of that system," *International journal of systems science*, vol. 1, no. 2, pp. 89–97, 1970.

[13] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*, pp. 52–59, Springer, 2011.

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

[15] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, pp. 2048–2057, 2015.

[16] M. Hayhoe and D. Ballard, "Eye movements in natural behavior," *Trends in cognitive sciences*, vol. 9, no. 4, pp. 188–194, 2005.

[17] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *arXiv preprint arXiv:1412.7755*, 2014.

[18] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[19] 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.