Reinforcement Learning With Generative Adversarial Representations

Anderson, S.,[a] Carson, M.,[b] Davuluru, N.,[c] Kelly, A.,[d] Kothari, R.,[e] and Tatke, T.[f]

(Dated: June 14, 2018)

Keywords: Atari, Reinforcement Learning, Neural Networks, GANs, LSTM, Recurrent nets

I. INTRODUCTION

We explore the problem of translating the representation of an information stream to a constrained domain, attempting to preserve the information of interest as defined by the goals of an agent. Specifically, we attempted to create a system that produces midi music given the 2D pixel information from Atari's Pong, aiming for music that can be used to play the game in absence of other information. We present an end-to-end system consisting of a generative model for producing the translation, a discriminative model to constrain the output to realistic music, and a reinforcement learning agent that learns to play pong given only the generated music. This kind of system has some useful properties:

- Generalizing a way to translate goal-critical information to alternate information channels would allow both significant improvements to the usability and interactivity of interfaces for the visually impaired, and a significant reduction in the cost of developing them through automation
- Dynamic music generation that tracks information within a game could allow creating immersive musical scores for interactive systems like video games with considerably less human effort
- The specifics of our architecture may imply a potential solution to a common pitfall in training GAN architectures, where one model learns quicker than the other, due to the asymmetry of the adversarial model and the non-zero-sum objective functions within the system

To this end, we introduce PSong, a Generative Adversarial Reinforcement Learning algorithm. We start with a pong agent[4], built on a single hidden layer neural network using policy gradients as suggested by Andrej Karpathy[1]. We then ensemble the agent with an RNN in an adversarial style to generate music that encodes information relevant to the RL agent.

[a] Electronic mail: ander28@pdx.edu
[b] Electronic mail: mpc6@pdx.edu
[c] Electronic mail: davuluru@pdx.edu
[d] Electronic mail: keap@pdx.edu
[e] Electronic mail: rishab@pdx.edu
[f] Electronic mail: a34@pdx.edu

II. MOTIVATION

Machine learning is being applied in a wide range of accessibility applications, often involving variations of machine translations and other ways of changing the representation of information[8]. Our research is not in machine translation, but could be extended into accessibility for the visually impaired. We train our Pong agent using music generated from samples of its environment, learning to play the game sonically instead of visually.

More theoretically, we present the concept of an **information preserving transformation**. This has wide applications for any context where we wish to learn a concept in a particular format that is different from the most natural or efficient way of representing it. This is somewhat the opposite of machine translation in the style of Show and Tell, in that we are embedding our concept in a higher dimensional space.

Why introduce this constraint? Translating between visual and auditory data is only the surface. An information preserving transformation could be used to enforce arbitrary constraints on the format of the learner's model, such as requiring that it conform to certain ethical standards and avoid algorithmic bias. This problem is already being approached using ordinary adversarial methods[2]. But our method of censorship allows the learner to train simultaneously against the censor and the target model. This could even be applied to such tricky problems as superintelligence control, which is of great future concern[3]. Or, by requiring a representation in the form of natural language explanations, Bayesian priors, logical proofs, etc., we could force an AI to explain its reasoning, another hot button issue.

III. METHODOLOGY

A. Datasets

Deep learning is a package deal with the curse of data dimensionality. Fortunately, Psong, is an ensemble of three deep neural networks! We extend our credits to beepbox - an online sound synthesizer that embeds the midi representation in the browser URL. We also observed that most of these songs were shared as tweets. To gather our dataset, we scraped tweets that have *#beepbox* in them and further processed the URLs to pull midi information out as JSON files.

B. Preprocessing

We begin with high dimensional data in the form of pixels and their color values (210 x 160 x 3). We crop the uninformative sections of the screen image (score and frame) and downsample to a 40 x 40 image with a single color channel. We then subtract off background colors and set all remaining pixels with non-zero values (paddles and ball) to 255. This results in a sparse matrix that represents a black and white image with only informative data as non-zero values, since shades and colors are not necessary for the agent to learn. Through normalization we convert our black and white image into an array of 40 x 40 binary values. We then further reduce the dimensionality of our environment space by squishing these 1600 integers with binary values into 16-bit float representations. Thus, reducing the input space of our generator to a size of 100.

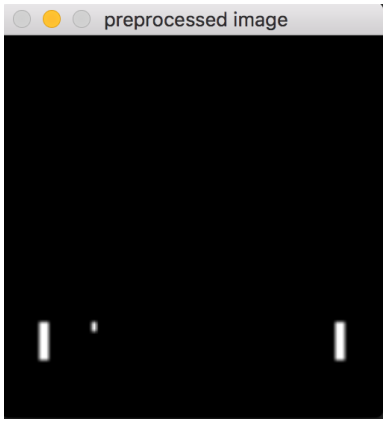

FIG. 1: A reduced version of the Pong environment from open AI with a dimensionality size of 40x40.

C. Dimensionality Reduction

The original dimensionality of an Atari game environment is 210 x 160 x 3, totaling 33,600 inputs. Through experimentation, Deepmind[5] found an optimal input for an AtariRL algorithm to be 84 x 84 or 7,056 inputs. But even with this reduction in dimensionality, their AtariRL agent took nearly 6 days of training to learn state action pairs to a high degree of confidence. By using our alternate representation of Pong pixels in the form of generated midi data, we are able to reduce the dimensionality of the RL's environment space to a low dimensional sequence of midi note values. This sequence of notes is produced from a time series of sampled pixels, encoded within them the full details and motion of all relevant game objects. Thus, once we have a generator that is capable of transforming Pong environment pixel data into sequences of midi data that fool a discriminator into believing they come from the training data's distribution, it should be fairly trivial for the RL agent to learn how to interpret its input.

D. GPU Support

Our end-to-end ensemble of two RNNs and a multilayer neural network have a high degree of computational complexity. Original experiments of our initial system showed CPU bottlenecking between backpropagations of our three neural networks. Despite the availability of multiple 1080TI GPUs, we were not able to fully utilize these resources due to the ensembling method along with our online, single batch processing. Several optimizations of the algorithm led to significantly increased throughput, but still resulted in CPU bottlenecking between GPU calculations. Attempts to parallelize the model and to construct a single end to end model that is piecewise three separate models are still in progress.

IV. ARCHITECTURE EXPLORATION

A. Generative adversarial network

Generative adversarial networks fit in perfectly to model some transformations mentioned below because of their ability to "compete" against each other by definition. The generator focuses on producing data that the discriminator tries to classify as real or fake, thereby doing a better overall job to produce entities closer to the real data distribution. Fig 2 represents a typical generative adversarial network architecture. Fig 4 gives a perspective of our ensemble model of the GAN with the RL agent.

In order for our RL agent to learn to play Pong through an alternate representation of its environment in the form of midi music, a generative model must be producing a transformation from the pixels to midi data. Since our goal is to create timeseries music, representing a timeseries of frames within the agent's environment, the generator has to create a reliable and repeatable transformation. Pixels as inputs and discounted rewards from the RL agent informing its loss function is not enough for the generator to learn to produce diverse and pleasing music. So, an adversarial model had to be introduced in the form of an RNN[6] that acts as a discriminator for the output of the generator. This discriminator acts as a gate, probabilistically allowing generated midi data through to the RL agent.

B. RNN and LSTM

RNNs are a clear model choice for generating and evaluating time series data such as music. And LSTMs are extremely useful for preventing loss of gradients and producing faster learners. Through much research[6] we

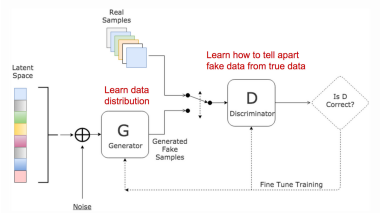

FIG. 2: A typical generative adversarial model with generator and discriminator

found[7] an optimal balance between computational complexity and model capability.

LSTMs are a level higher in terms of complexity when compared to RNNs because of the additional cell states and gating functions. RNNs are very much capable except when errors need to be backpropagated through time - that's when gradients will either blow up or shrink. Hence, the need for networks with stateful memory (LSTM).

This research led to the creation of generative and discriminative RNN models with three LSTM layers and two fully connected layers. The three LSTM layers consists of 256 cells with the first two layers having a dropout of 30%. These LSTMs are followed by a fully connected layer of 128 neurons with a 30% dropout and an output layer the size of the midi note vocabulary within the training data. While the generator has a SoftMax activation on the final layer to produce a probability distribution over the selection of notes, the discriminator has a Sigmoid activation to determine the binary choice of whether its input represents real or fake data.

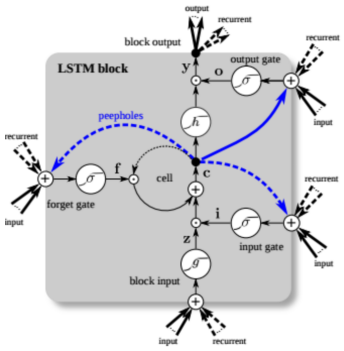

FIG. 3: An LSTM cell with different gating functions

V. RESULTS

With our constrained timeline, we narrowed our focus and worked towards a demonstrable model. Splitting our development process into three sections, generator, discriminator and RL agent, helped parallelize the process. We fairly quickly had models capable of generating music and beating a traditional, computer controlled Pong opponent. We then moved on to building an ensemble of these two models, so we had an end-to-end, online, recurrent/RL model. The RNN model sampled the game environment, generated midi representations of that environment, and fed the midi data to the RL agent that in turn predicts a move within the environment.

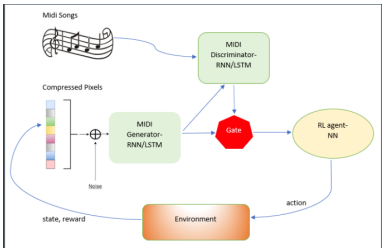

FIG. 4: PSong architecture, a Generative Adversarial Reinforcement Learning algorithm

The most challenging and computationally complex aspect of the process involved returning batched discounted rewards to the generator to inform its policy while training the ensemble in an online fashion. Fearing this would ultimately lead to a midi generation policy of a single tone to inform the agent to move up and one to inform the agent to move down, we implemented a third leg to the stool: a discriminator model to further inform the generator's policy of creating music, comparing and gating generated music against a real distribution of video game music from a training set. After days of training we have a working, end-to-end model which plays Pong through a midi format. It loses but not embarrassingly so, and shows every sign of being able to train to the same level as a pixel based agent. The midi sounds it makes cannot honestly be described as interesting music yet, but they do show some variation; it does not simply encode "up/down" as a pair of notes.

VI. FUTURE WORK

Our intermediate results have given us a much better sense of what might be feasible to explore in the future:

- We believe the more correlated the models, the more capable the model is to learn as an ensemble. The current use of discounted rewards as a loss function to the generator is a naive approach to informing the model how well the RL agent is learning from the generated music. A potential improvement would be the policy of perturbing the weights in the generator by backpropagating from the input layer of the RL agent back through the output layer of the generator. This would more closely correlate the generated music to the learning of the RL agent. Thus, simulating a single, end-to-end model instead of a piecewise ensemble.

- Currently, the model uses memory for storing elements like the genuinity of notes produced and the

observation-reward-action vector from the agent. These are implemented using standard lists in Python, which by far, could be the most prototypical use-case. In fact, Python lists are extremely slow compared to the case where these memory elements could be off-loaded to a Tensorflow backend, expediting the training process exponentially. The obvious reason for this is the model literally queries data outside of its shell and loads this data during every episode and iteration of training.

- Given the timeline to work on the project, it seemed out of scope to look into further optimizations or hyper-parameter tuning to try and yield better results. Furthermore, the complexity of the system is extremely high and improvements in computational complexity are difficult because each training session is linearly dependent on the model.

VII. CONCLUSION

We have introduced a novel neural ensemble architecture for the creation of information preserving transformations, and implemented it on the simple toy problem of singing pongs and pinging songs. Even with a very simple application domain, this is a very complex structure, and a great deal of optimization will be needed before it can be expanded into a full product. However, we have demonstrated that the fundamental concept is sound, and we are capable of preserving relevant information through the transformation.

VIII. REFERENCES

[1]Andrej Karpathy. Deep Reinforcement Learning: Pong from Pixels, 2016.
[2]Angwin, J., Larson, J., Mattu, S., and Kirchner, L. Machine Bias. *ProPublica* (5 2016).
[3]Bostrom, N. Superintelligence: Paths, dangers, strategies., 2014.
[4]Carson, M., and Davuluru, N. Atari-RL, 2018.
[5]Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. a., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Playing Atari with Deep Reinforcement Learning Volodymyr. *Nature* (2015).
[6]Mogren, O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *CoRR abs/1611.09904* (2016).
[7]Sigurur Skúli. How to Generate Music using a LSTM Neural Network in Keras, 2017.
[8]Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2015).