SEM3: Sample Efficient Multi-Modal Model

Noah Rousell, Adithya Sriram, Julian Beaudry, Ryan Eng

Department of Computer Science
Brown University
Providence, RI
noah_rousell@brown.edu

1 Introduction

There has been extensive progress made in recent years on diverse tasks through Machine Learning (ML) algorithms and, in particular, a class of algorithms known as Artificial Neural Networks (ANNs). The Transformer architecture and the attention mechanism have led to powerful Large-Language-Models (LLMs) with the ability to perform complicated language processing tasks, such as acing standard tests on mathematics and law, or understanding and correcting code. In the field of computer vision, vision models using the technique of Stable Diffusion have the ability to generate hyper-realistic images or to create novel art.

Yet the field of Reinforcement Learning (RL) has not made a large impact on day-to-day life. We have been able to train algorithms that rival and exceed human level on tasks such as chess and Go, but this progress has not made it to embodied settings due to the need for massive amounts of data during training and the narrow skillset of these bots. In recent years, pioneering work has been done to discover what is needed to create sample-efficient RL models. In this paper, we develop a model that builds on prevous work and allows the capacity for multi-modal input. We train our model on Atari 2600 games using the raw input of video frames and audio as observations.

2 Related Work

Q-Learning consists of predicting the expected future reward of taking a particular action in a particular state. Deep Q-Networks (DQNs) use deep neural networks to predict Q-values directly from raw inputs such as the pixels of a video game, making hand-engineered features unnecessary [3]. When training a model using Q-Learning, the agent typically interacts with the environment, choosing an action based off of the current observation, and receiving a reward. The observation, action, and reward are combined into an *experience* and stored into a replay buffer that can then be sampled from in mini-batches to perform gradient descent on the network.

The typical Q-Learning loss consists of minimizing the difference between a network's predicted Q-value and a temporal-difference target.

$$L = Q(s_t, a_t) - (r_t + \gamma \max_a Q^*(s_{t+1}, a))$$

The target is constructed as the sum of the ground-truth reward from the environment and a target network's projection of future Q-values, discounted by the factor γ. The target network

is a version of the online network with "stale" parameters, often implemented as an exponential moving average of the online network. The use of a target network stabilizes training and prevents oscillation.

Since the advent of DQN, many improvements have been made to stabilize training and to make it more efficient and effective.

In 2016, Double Q-Learning was introduced to temper the optimistic Q-values produced by constructing the target by maxing over the target network's Q-values [7]. Double DQN consists of using the online network to perform action selection when constructing the target. The modified loss then becomes,

$$L = Q(s_t, a_t) - (r_t + \gamma Q^*(s_{t+1}, \arg\max_a Q(s_{t+1}, a)).$$

Dueling Networks were introduced to stabilize training [8]. Instead of implicitly approximating the value function in each output Q-value neuron, Dueling DQN splits the final layer of a Q Network into two fully-connected layers. The first consists of a single neuron and approximates the value function $V(s)$, representing the value of being in the current state s, while the second approximates the advantages $A(s, a)$, representing the advantage of taking action a over the average action. The Q-values can then be reconstructed according to the following question:

$$Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a') \right)$$

Explicitly approximating a single value that is then used to construct each of the Q-values improves training.

Distributional Q-Networks improved the DQN architecture further by approximating a distribution over q-values in place of their expected values [1]. Finally, prioritized experience was introduced, consisting of sampling experiences from the replay buffer in proportion with their corresponding RL loss such that more gradient descent steps will be performed with experiences that the model is a poor predictor for [4].

In recent years, progress has been made on sample-efficient RL algorithms that require significantly less data to train. One notably model is Self-Predictive Representations (SPR), which fills in the sparsity of the reward signal by adding in a self-supervised prediction component [5]. The SPR architecture consists of all of the DQN improvements introduced above, and also splits the Q-network into an encoder, which encodes the observation into a latent space, and the Q-learning head, which predicts Q-values from the latent space. During training, the latent space is concurrently used with a transition model to predict the latent representation of future states. SPR adds a term to the loss function consisting of the cosine similarity between the network's predictions of future latent representations and their corresponding target embeddings. The target embeddings are produced by applying a target encoder (an encoder with parameters that are an exponential moving average of those of the online encoder) to the future observations. This setup trains the encoder to encode information that is useful for predicting future states, an important problem implicit in predicting q-values.

Other work in sample-efficient RL has been involved in scaling the *replay-ratio* to greater levels [2]. The *replay-ratio* is the number of times that the agent performs a gradient update for each interaction with the environment. Higher replay-ratios seem an obvious way to improve sample efficiency, but often lead to significant overfitting to early experience. Scaled-by-Resetting Self-Predictive-Representations (SR-SPR) combats this issue by periodically resetting model parameters. Later layers are completely reset, while earlier layers (e.g. encoder, transition model) are interpolated 20%

to randomly initiated values using a technique known as *shrink-and-perturb*.

The current state-of-the-art on sample-efficient Atari 2600 play is held by the Bigger Better Faster (BBF) model introduced in 2023 [6]. The BBF model consists of the SR-SPR model with a few important changes. BBF utilizes a wider version of an ImpalaCNN (a 15-layer Resnet) for the encoder instead of the standard 3-layer CNN. They also use a stronger version of shrink-and-perturb, interpolating 50% to random values instead of 20%. They use n-step learning, in which target Q-values are constructed using n ground-truth reward values instead of only 1. The number of reward values used is known as the update horizon. They incorporate weight decay and employ a exponential annealing schedule during training for both the update-horizon n and discount factor γ.

Almost all models in the literature focus on single-modality models, yet some research has been conducted on the usefulness of incorporating audio when training agents to play Atari 2600 games. One paper published in 2018 called Cross-Modal Attentive Skill Learner (CASL) incorporated both video and audio frames to achieve superhuman scores on the Atari 2600 games Amidar and H.E.R.O. In their architecture they implemented an LSTM-based skill learner which learns to pay attention to the audio when relevant. They tested Atari games in which the audio does not necessarily convey information not included in the video but the input is more salient from the audio which allows the model to learn faster. Based on the results of this paper, we decided to explore the impact of incorporating audio by re-implementing and augmenting RL architectures proven to produce good results with video inputs.

3 Method

We recreated the BBF model as our base using the TensorFlow python library. We then constructed an audio encoder to process audio observations. The audio embedding produced by the encoder is concatenated to the spatial latent video embedding along the channel axis (Figure 1).

Our experimental setup begins with a *yaml* file to store the parameter configuration. The model initially interacts with the environment without performing gradient updates for a set number of environment steps. Experiences are stored in the replay buffer. Then the agent begins to perform $RR = 2$ gradient updates per environment step where RR is the replay ratio. Evaluation episodes are conducted every 5000 environment steps and the model's performance on train and evaluation episodes are logged in csv files. The model's parameters are saved at the beginning of each evaluation period.

We performed model runs using Oscar, Brown University's Center for Computing and Visualization's computing cluster. We utilized the Arcade-Learning-Environment (ALE) to create environments for the agent to interact in. Due to the lack of audio support in the main version of ALE, we used a fork of the codebase augmented to have audio support by the authors of Cross-Modal Attentive SKill Learner and compiled from source.

4 Results

Game	SEM3	Random
Assault	Value	222.4
Roadrunner	2700	11.5
Roadrunner w/ audio	5020	11.5
Amidar	57	5.8
Amidar w/ audio	78.4	5.8

Table 1: Best Evaluation results by 30k environment steps. Scores are the average reward of 5 episodes.

We tested our model on the Atari 2600 games

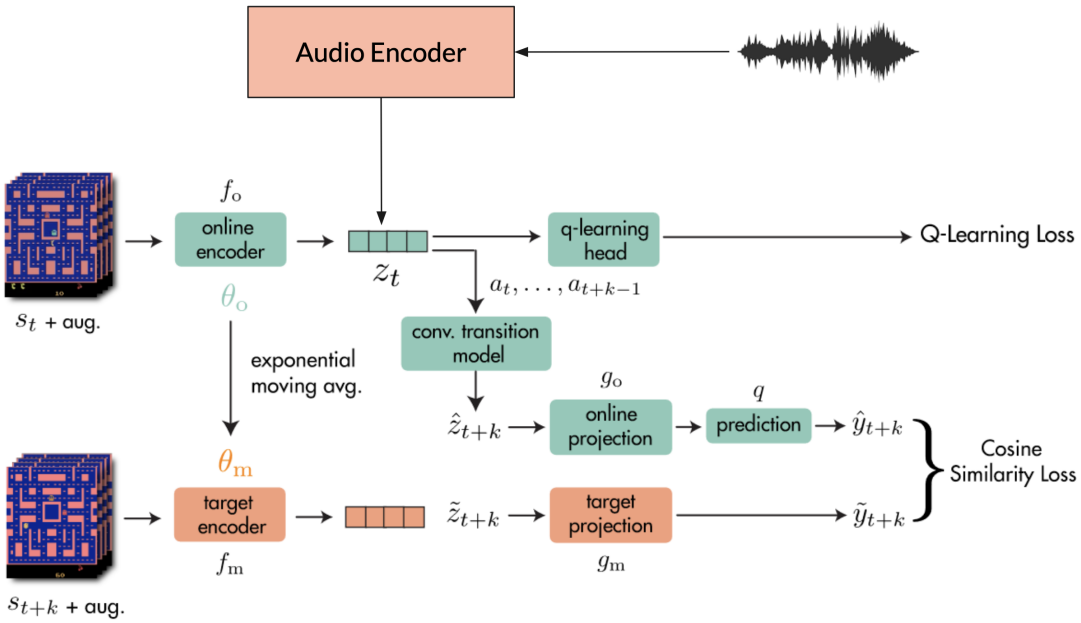

Figure 1: The Hybrid model that we develop. The audio signal is encoded using a CNN and concatenated along the channel axis of the latent embedding z.

Figure 2: Assault and Roadrunner

Assault, *Roadrunner*, and *Amidar* (Figure 2). Performances relative to random and human are given in Table 1 and videos of the model playing can be found in our github repo [1]. Due to computation restrictions, we utilized a replay ratio of 2 instead of the 8 used by BBF. We were also unable to run for the entire 100k environment steps that is traditionally used as a benchmark, so our results were achieved with between $\frac{1}{16}$ and $\frac{1}{8}$ of the gradient steps that BBF used. Even with the limited training time, we got initial results showing that the model is able to learn faster when audio is integrated (see Figure 3). Due to the promising preliminary results our model achieved, we believe that a fully-trained version of our model would be competitive with BBF and potentially beat it on certain Atari games where audio is useful.

5 Challenges

The chief obstacles we faced working on this project were setting up the ALE environment so that it supports audio and computational lim-

[1] https://github.com/nwrousell/SEM3

Figure 3: Evaluation scores for the base and audio-enabled model on the Atari 2600 game Roadrunner

itations. As audio support is not included in the main ALE branch, we had to use a fork of the repository from 2017 and compile the repository from source. Due to old dependencies and not having sudo access on Oscar this was a challenge, but we were able to compile it on Oscar! While we used Oscar and ran multiple experiments in parallel, computation was still a major constraint and we were not able to train our model to the extent that BBF or earlier models were able to.

6 Discussion and Future Work

In the future, if we gain access to greater compute resources we plan to run a wider range of experiments and search the hyperparameter space more fully. We are also interested in trying out more sophisticated integrations of audio input, such as attention mechanisms.

We are overall very satisfied with how the project turned out. We were able to achieve good results on multiple Atari 2600 games—reaching our stretch goal! One pivot we made was from attempting to train one instance of the model on multiple GPUs to running multiple experiments in parallel. If we had more time we believe we could improve the audio encoder and re-organize parts of the codebase. Our biggest takeaway from the project is that Reinforcement Learning is challenging, but rewarding!

Acknowledgements

We'd like to thank Professor Ritambhara Singh for facilitating the Deep Learning course (CSCI 2470) in which we conducted this project, as well as the deep learning day in which we presented our work. We'd also like to extend a thank you to our project mentor Raymond Dai, as well as Brown University's CCV for their helpful support responses.

7 References

References

[1] Marc G. Bellemare, Will Dabney, and Rémi Munos. A Distributional Perspective on Reinforcement Learning, July 2017.

[2] Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G. Bellemare, and Aaron Courville. Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier. In *The Eleventh International Conference on Learning Representations*, September 2022.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[4] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay, February 2016.

[5] Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron Courville, and Philip Bachman. Data-Efficient Reinforcement Learning with Self-Predictive Representations, May 2021.

[6] Max Schwarzer, Johan Obando-Ceron, Aaron Courville, Marc Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, Better, Faster: Human-level Atari with human-level efficiency, November 2023.

[7] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning, December 2015.

[8] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling Network Architectures for Deep Reinforcement Learning, April 2016.