LEARNING FROM MUSICAL FEEDBACK WITH SONIC THE HEDGEHOG

Faraaz Nadeem
Massachusetts Institute of Technology
faraaz@mit.edu

ABSTRACT

Most videogame reinforcement learning (RL) research only deals with the video component of games, even though humans typically play while experiencing both audio and video. In this paper, we aim to bridge this gap in research, and present two main contributions. First, we provide methods for extracting, processing, visualizing, and hearing gameplay audio alongside video. Then, we show that in *Sonic The Hedgehog*, agents provided with both audio and video can outperform agents with access to only video by 6.6% on a joint training task, and 20.4% on a zero-shot transfer task. We conclude that game audio informs useful decision making, and that audio features are more easily transferable to unseen test levels than video features.

1. INTRODUCTION

Although classification and decision making are well studied areas of machine learning (ML), most research and practical application of these areas has involved the use of video or text based data, as opposed to audio. Of the audio research that exists, only an even smaller subset has analyzed environmental audio or music as a method of providing feedback.

This gap in research needs to be addressed since environmental and feedback-related audio is critical to achieving human performance on a number of tasks, especially when visual or textual clues are not sufficient. Self driving cars cannot achieve full automation without being able to react and respond to emergency vehicle sirens in the distance, or nearby car horns. Emergency responders listen for fire alarms and calls for help to locate the emergency when arriving at a scene. The clicking feedback produced by a mouse, keyboard, or button, confirms to the user that the hardware or software has received the input. [1].

In this work, we use *Sonic The Hedgehog* games for the SEGA Genesis as an environment to understand how RL agents can incorporate both audio and video observations, to make decisions with immediate consequence, feedback, and long term goal. [1]

2. BACKGROUND

2.1 Sonic The Hedgehog

Sonic The Hedgehog is a 2D side scrolling platforming game. The main idea of the game is to navigate Sonic through vertical loops, over bottomless pits, off of springs, while collecting rings. A central game mechanic is that Sonic can build great speed if his movement is not interrupted by stopping or bumping into obstacles, and this can then be used to launch him into the air off of ramps, or through an array of enemies. Each game consists of individually themed Zones, and each Zone has 1 to 3 Acts. Going forward, we will refer to each Act as a level.

Rings are placed in groups of 3 or more along the levels. They give points, and act like a shield. If Sonic is hit by an enemy or hazard without any rings, he loses a life. But if Sonic has at least 1 ring, then hitting an enemy or hazard knocks Sonic backwards and he forfeits up to 20 of his rings, without losing a life.

Many levels contain sections which are underwater. Sonic can survive approximately 30 seconds underwater until he needs to find an air bubble or jump out of the water to avoid drowning.

2.2 Related Work

Gotta Learn Fast [2] is a transfer learning benchmark with Gym Retro [3] on the Sonic games for the SEGA Genesis. It shows that pretraining on 47 train levels before fine tuning on 11 test levels achieves the best test result when compared to several other models. They do not perform any audio-related experiments.

Kim et al. [4] modify the Atari Learning Environment [5] to support audio queries, and demonstrate that latent audio/video features increase performance on H.E.R.O and Amidar on the Atari 2600, and transfer knowledge to accelerate learning in a door puzzle game.

Kaplan et al. [6] show that the additional modality of natural language suggestions can be used to improve performance on the Atari 2600 games. Ngiam et al. [7] present a series of tasks for multimodal learning and demonstrate cross modality feature learning, where better features for one modality (e.g., video) can be learned if mul-

[1] Link to Github repository with code and gameplay video examples provided here https://github.com/faraazn/meng

tiple modalities (e.g., audio and video) are present at feature learning time. Poria et al. [8] show that a multimodal system fusing audio, visual, and textual clues outperforms previous state of the art systems by 20% on a YouTube sentiment dataset.

Henkel et al. [9] show that RL can be useful in the music domain by applying it to the score following task. Various works have used neural network approaches to acoustic scene classification [10–12] and sound event detection [13–15], achieving state of the art results.

3. RELEVANT MUSIC THEORY

3.1 Western Associations in Music

In Western music theory and culture, there are clear associations of consonance, dissonance, and rhythmic patterns with certain emotions. The major mode is slightly more consonant, and therefore associated with "happy, merry, graceful, and playful", while the minor mode is slightly more dissonant, and associated with "sad, dreamy, and sentimental". Firm rhythms are perceived as "vigorous and dignified", while flowing rhythms are "happy, graceful, dreamy, and tender". Combining these ideas, complex dissonant harmonies are "exciting, agitating, vigorous, and inclined towards sadness", and simple consonant harmonies are "happy, graceful, serene, and lyrical" [16].

Studies have shown that these associations can largely be attributed to cultural conditioning, rather than universal human behavior. Residents of a village in the Amazon rainforest with no exposure to Western music showed no preference between consonant and dissonant sounds [17]. We can trace back Western preference for consonance at least to the early 18th century, when the name "diabolus in musica", or "the Devil in music" was attributed to the augmented fourth, the most dissonant interval [18].

3.2 Application to Videogame Sound Effects

Videogame music and sound effects are designed to leverage our implicit biases as a means of meeting expectation and lowering the barrier to entry for learning a new game. We can better understand this by applying it to the Sonic games.

It is good for Sonic to collect rings, since they give some protection before losing a life, award more points, and collecting enough of them result in an extra life. In line with our understanding of Western preference for simple consonance, the ring sound is a major triad, implying a sense of positive value for the act of acquiring a ring. It is comprised of the notes E5, G5, and C6, which form specifically a first inversion C major triad (Figure 1).

When Sonic loses rings after being hit by an enemy or hazard, this is bad because he is prone to losing a life with an additional hit. In line with our understanding of Western music theory, the corresponding sound is aggressively dissonant, implying a sense of negative value for the act of losing rings. This sound is created by rapidly alternating notes A6 and G6, which form a major second interval.

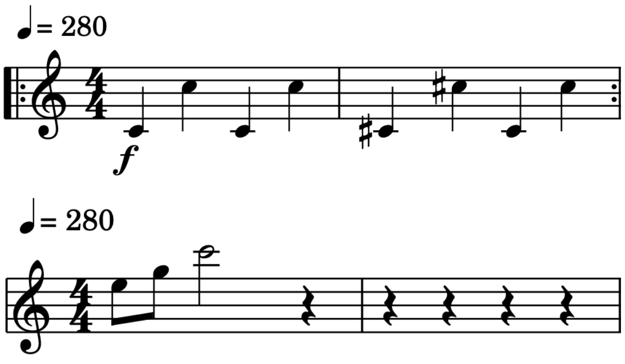

Figure 1. Drowning Theme and Ring Sound. Top: the drowning theme consists of dissonant jumps between octave intervals. Bottom: the three consonant notes of the ring acquiring sound.

When Sonic has spent too much time underwater without air, and is close to drowning, the drowning theme begins to play. It alternates between octave intervals on C, and a minor second interval jump to octave intervals on C# (Figure 1). The jumps lie on a dissonant interval, and the music gets increasingly loud and fast as Sonic gets closer to drowning, giving a strong sense of impending doom.

In general, consonance bias is an effective tool for conveying positive or negative value of player actions or setting emotional expectation with audio, without needing to spoon-feed explicit instructions or visual cues.

4. EXPERIMENTAL SETUP

4.1 Reinforcement Learning

Reinforcement learning [19, 20] is an increasingly popular field of machine learning research. It is a semi supervised approach to teaching an agent complex sequential decision making in an environment with potentially sparsely labeled data or delayed reward.

More concretely, we define an agent that operates under a policy π with parameters θ, and produces actions given an observation, $\pi_\theta : o \to A$. At time t, the agent takes an action $a \in A$ while in state $s \in S$, transitions to state $s' \sim \mathcal{T}(s, a, s')$, and receives observation $o' \sim \mathcal{O}(o', s')$ and reward $r = \mathcal{R}(s, a) \in R$.

In the Sonic games, we can imagine a scenario where Sonic is being approached by an enemy. The game state s tells us the magnitude and direction of the enemy's velocity. An observation o of this state might be a single frame of video which tells us the enemy is close to Sonic. When the player takes an action a to control Sonic, performing a spin attack may lead to a transition (s, a, s') to a new state s' where the enemy is defeated. This would result in a new visual observation o' of the defeated enemy, and a reward r of 100 points.

The goal of an RL agent is to maximize the total expected reward. To optimize our agent parameters θ and achieve the best performance, we use Proximal Policy Optimization (PPO) [21]. PPO is a policy gradient algorithm that achieves state of the art results on a number of RL benchmarks. It builds on former advancements in policy gradient research [22–24], while also aiming to be fast and easy to implement. The details of its implementation are outside of the scope of this work; we used an open source implementation [25].

4.2 Task Outlines

To quantify the effect of audio on videogame RL, we set up two different tasks.

1. Joint PPO Training. We train a single agent on 47 of the 58 levels, which we call the training levels. The agent trains for 30 million time steps, and we self-evaluate the agent on the training levels every 5 million steps.

2. Zero-shot Transfer. We take the jointly trained agent from the previous experiment and evaluate it on the 11 test set levels, without any fine tuning. If the agent learned general techniques for level progression in task 1, it should be able to perform better than a random agent, or one following a simple policy. For each test level, there is at least one training level from the same Zone.

4.3 Open AI Gym Retro

We used the Open AI Gym Retro framework to set up the two experiments outlined above. Its API allows us to interface with games by inputting player actions, and receiving the updated game state, observations, and reward in return. There are thousands of games available to use with Gym Retro, including the original Sonic series for the SEGA Genesis.

However, the Gym Retro (version 0.8.0) package does not readily expose an API for interacting with the videogame emulator's audio. To aid others facing this issue, we have open sourced our codebase, in which we were able to expose the audio features for training our RL models, and create various tools for visualization and playback. This includes tools for both online and offline playback, audio visualization, and neural network feature heat mapping for a given runthrough of a level.

4.4 Environment Setup

For all of our experiments, we use mostly the same base environment setup as described by Nichol et al. [2]. This includes usage of save states, episode boundaries, stochastic frame skips, action space reduction, and reward function. At a high level, this setup overrides the in-game point system to instead reward Sonic for making new progress to the right of the screen, and reduces the number of / speed with which buttons can be pressed, among other things. Due to computational constraints, we set the horizon of experience generation to 512 instead of 8192, number of workers per level to 3 instead of 4, and total train steps to 30 million instead of 400 million. The convolutional neural network (CNN) architecture we use to process audio and video observations is given by Figure 2.

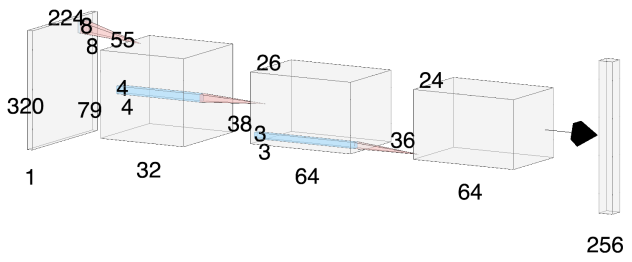

Figure 2. CNN architecture for the observation encoder. The input observation in this example is a grayscale frame of video, and the same structure is used to encode audio.

4.5 Agent Variants

We create 5 different agent variants to be able to differentiate the effects of incorporating audio.

Agent 1: conceptually the same as the one used in the Gotta Learn Fast [2] benchmark, albeit with fewer trainable parameters and shorter training time due to computational constraints.

Agent 2: we convert video inputs to grayscale and introduce a random start of 16-45 random actions (about 1-3 seconds) which we expect will result in improvements to generalization.

Agent 3: takes an additional video frame from the previous time step as input, and therefore also has double the model size as Agent 2. We hypothesize that Agent 3 may be able to learn temporal features, and therefore achieve better results than Agent 2.

Agent 4: only keeps the current grayscale video frame, but adds audio features. We construct these audio features by concatenating the past 16 frames, without frame skipping. Since the Gym Retro sample rate is 44100 Hz, this means we get 735 samples of audio per frame and 47040 samples per observation (about 1 second long). We process the 1D sequence of samples into a 2D mel spectrogram with 256 mels, window size 256, and hop length 128, for a resulting size of 367 by 256 pixels. We construct a new encoder for this audio with the same CNN architecture as video, so the overall model size is comparable to Agent 3.

Agent 5: the same as Agent 4 except we create a 2D mel spectrogram with 256 mels, window size 1024, and hop length 512, for a resulting size of 92 by 256 pixels. This spectrogram is 4 times smaller, so the overall model size is comparable to Agents 1 and 2.

To normalize the spectrogram values, we first convert from the power to decibel scale, setting the max decibel range to 80, and then dividing by 80 to put the range of values between 0 and 1. Separate CNNs are used to process the video and audio observations, each with hidden size 256, and concatenated together into a final hidden state size 512.

4.6 Baselines

4.6.1 "Hold Right" Baseline

By definition of the reward function defined for us [2], Sonic accumulates reward as he makes net progress to-

Agent	Video Frames	Rand. Start	RGB	Audio	Num Params
1	1	no	yes	no	14m
2	1	yes	no	no	14m
3	2	yes	no	no	28m
4	1	yes	no	yes	31m
5	1	yes	no	yes	18m

Table 1. Agent Variants.

wards the right. We can therefore define a simple yet effective policy that requires no machine learning: always make Sonic move right. We evaluate this policy and set it as a lower bound for the performance we expect our trained agent to achieve.

4.6.2 Human Baseline

There is an existing human baseline for the Sonic Genesis games. In this baseline, 4 test subjects practiced on the 47 Sonic training levels for 2 hours, before playing each of the 11 test levels over the course of an hour.

5. RESULTS

5.1 Overview

Table 2 shows us the most important result of this work, which is that audio+video agents outperform video-only agents on the joint training task, and achieve higher scores on the zero-shot transfer task. Agent 5, which is provided with the current frame of video and past 1 second of audio, outperforms Agent 3, which is provided with the current and previous frames of video, no audio, and 55% larger model size, by 6.6% on the joint task, and 20.4% on the zero-shot task. This result supports our hypothesis that Sonic game audio informs sequential decision making, and extracted audio features are more easily transferable to unseen test levels than video features.

The "Final Avg" section contains the primary scores used to evaluate overall agent performance. Final refers to the fact that these scores were computed by evaluating the final saved checkpoints of each agent, rather than averaging over the entire training run.

We add a "Best Ckpts" table, which aggregates the top per-level mean scores across all corresponding agent checkpoints. We saved checkpoints every 5 million train steps. The goal of adding this table is to demonstrate each agent's best effort for each level.

Scores for each agent are presented as the mean $\pm$ the first standard deviation taken along the *individual level averages*. The gameplay figures are overlayed with Score CAM heat maps [26, 27], and important parts of these figures are annotated with circles or arrows. Each agent was trained and evaluated with three different random seeds.

5.2 Video Agent Analysis

Agent 1 is only able to outperform the "hold right" policy by a small margin. We found that over the course of training, it tended to overfit to specific levels every few million train steps.

Agent	Joint Final Avg	Zero-Shot Final Avg
1	1344.6 ± 206.4	435.6 ± 130.0
2	1479.5 ± 911.3	578.5 ± 409.7
3	1905.0 ± 286.9	678.5 ± 238.3
4	1893.7 ± 225.3	817.3 ± 320.2
5	$\mathbf{2031.1 \pm 501.9}$	$\mathbf{936.6 \pm 220.8}$

Agent	Joint Best Ckpts	Zero-Shot Best Ckpts
1	1780.2 ± 1708.5	755.1 ± 1028.1
2	2998.6 ± 2010.63	1526.8 ± 1257.4
3	3013.0 ± 2186.2	1686.9 ± 1155.1
4	$\mathbf{3041.4 \pm 2227.9}$	$\mathbf{1779.2 \pm 1403.1}$
5	2901.5 ± 2011.1	1756.9 ± 1298.3

Baseline	Joint Final Avg	Zero-Shot Final Avg
Hold Right	1099.1 ± 1092.8	321.9 ± 277.5
Human [2]	–	7438.2
N. et al. [2]	5083.6	~ 1000

Table 2. Performance summary of all 5 agent variants and 3 baselines over both tasks. Note that N. et al. trained with significantly more computational power.

Agent 2 uses grayscaling instead of RGB, and adds a random start between 1 and 3 seconds to the Agent 1 setup. Adding the random start makes it much more difficult for Agent 2 to try to memorize a high reward path through each level. This substantially improves scores on the joint and zero-shot tasks, because increased starting state diversity forces Agent 2 to learn more general features for level progression. If the agent learns that jumping over spikes is good in one level (Figure 3), the the lower layers of the CNN that were used to recognize the spikes will help to transfer that knowledge to other levels.

Agent 3 builds upon Agent 2 by doubling the hidden layer size to 512 and increasing the number of video frame observations from 1 to 2. By including the video frame from the previous time step, Agent 3 now has the ability to learn temporal meta-variables, such as relative velocity of objects on screen. Agent 3 does actually learn to use this information to determine, for example, when Sonic does not have enough momentum to go up a steep ramp, and must backtrack in order to build the momentum needed to try again (Figure 4). It improves over the Agent 2 joint train and zero-shot scores by 29% and 17%, and also provides lower variance results.

5.3 Audio+Video Agent Analysis

Agents 4 and 5 replace the second video frame with a spectrogram representing the last 1 second of in-game audio.

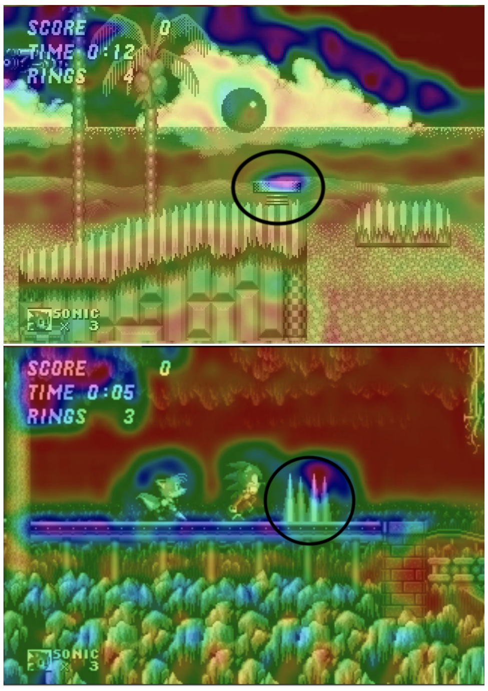

Figure 3. Agent 2 detects various objects during training. Top: the agent jumps on a spring to launch Sonic over the pit. Bottom: the agent recognizes that Sonic needs to jump over the spikes.

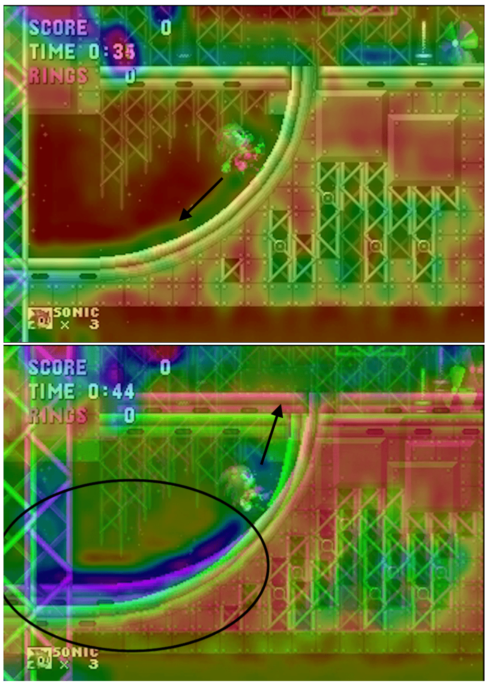

Figure 4. Agent 3 uses temporal information to judge its progression up the ramp. Top: the agent decides it does not have enough momentum, and begins to backtrack to the left. Bottom: the Score CAM weighting shows the agent understands that the screen shifting upwards means that Sonic has enough momentum to make it up the ramp.

5.3.1 Relative Receptive Field Comparison

Agent 4 constructs a spectrogram with window size 256 and hop length 128, while Agent 5 uses window size 1024 and hop length 512. This leads to spectrogram sizes 368 by 256 pixels and 92 by 256 pixels, respectively. Both convey the same amount of information, but result in audio CNNs with different numbers of trainable parameters (Agent 4 has about 14 million more than Agent 5) and different relative receptive field sizes (Agent 4 has a 4 times smaller relative receptive field). In other words, Agent 4 learns a larger number of smaller audio details, and Agent 5 learns a smaller number of larger ones.

In general, we found that Agent 5 was better able to learn high level audio features. Agent 4 was more prone to noise and overfitting because it could only recognize smaller parts of overall sound effects.

5.3.2 Important Audio Features

Rings are small, keep rotating, and turn into even smaller stars when grabbed by Sonic, so the act of acquiring them is not recognized by the agent's visual CNN component. The corresponding sound is easy to see in a spectrogram, so it is readily learned by the audio CNN component. The same is true for when Sonic loses his rings.

An agent will respond to the ring acquiring sound by increasing its likelihood of taking the "move right" action, since rings are generally located along paths that lead to forward progress in the level. Figure 5 shows an example of Agent 5 doing this on a zero-shot run of Hill Top Zone Act 2. Ironically, losing rings also produces the effect of increased "move right" likelihood, despite having a dissonant sound and negative connotation for humans, because Sonic is granted temporary invincibility after losing his rings. The agent uses this as an opportunity to get past the danger that was originally in the way.

Visual indicators that Sonic is about to drown are small see-through bubbles with numbers that count down and do not appear every frame. These cues are easily missed by the visual CNN, and video-only agents do not act on them. However, our audio+video agents learn to pick up on the drowning theme and quickly locate an oxygen-restoring air bubble or exit the water entirely to avoid losing a life (Figure 6).

Before Sonic gets to the drowning stage, there are pings that play every 5 seconds Sonic is underwater. In some levels where prolonged underwater travel is not always required, such as Angel Island Zone Act 2, Agent 5 exits the water after hearing the first one of these pings.

Surprisingly, a significant number of sounds highlighted by the Score CAM algorithm come from the background musical score, which is not supposed to be feedback-related. We hypothesize that these learned features are mainly an artifact of the audio CNN incorrectly attributing accumulated reward to the part of the score that happened to be playing at the time. Typically these features do not seem to have a significant effect on agent actions, but they can be coincidentally reinforced when the same parts of the music repeat.

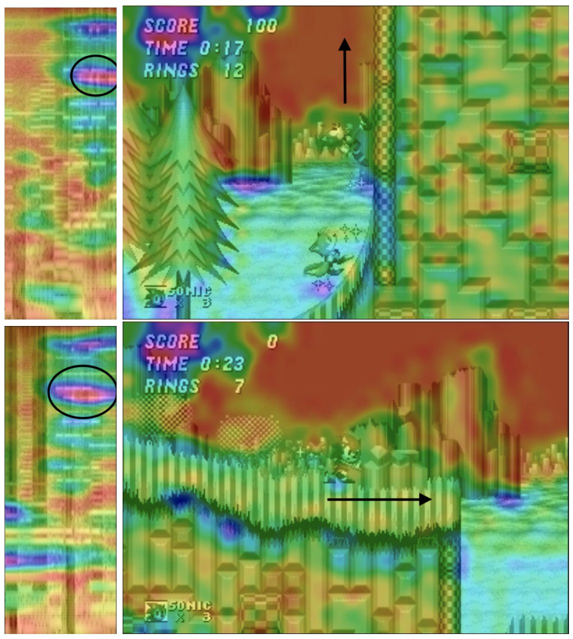

Figure 5. In Hill Top Zone Act 2, during a zero-shot run, Agent 5 looks at ring sound as motivation to move right. Note that the left column shows spectrogram heat maps from the last second of audio. Top: the agent acquires rings while holding right and continues to run up a steep cliff. Bottom: the agent continues to run into the rock blocking its path until the ring sound is freed from its memory.

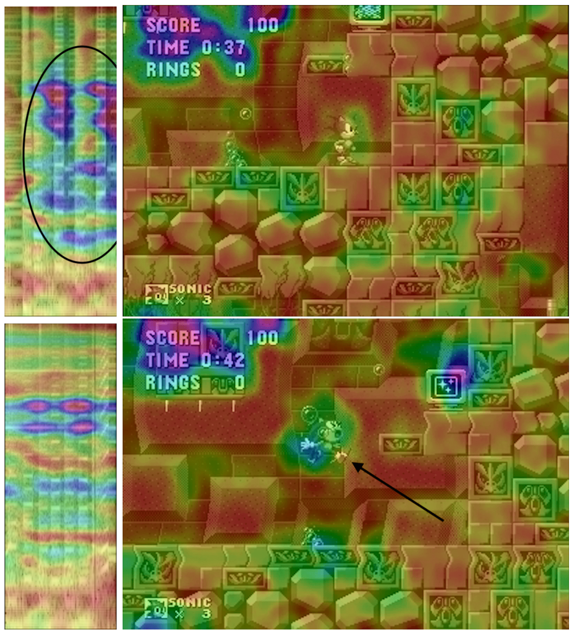

Figure 6. Agent 5 avoids drowning. Top: the agent hears the beginning of the drowning theme. Bottom: the agent navigates Sonic to the bubble stream on the left, and waits for a bubble to appear. The model highlights Sonic as he breathes in the air bubble, confirming the danger has passed.

5.3.3 Unimportant Audio Features

There are a few sound effects that we expected to be more important for our training tasks. We thought that the jump sound effect would be an important temporal indicator of Sonic's jump trajectory, i.e. determining if he is rising or falling. We also expected the sound effect produced by defeating a enemy to be important for learning to defeat enemies before getting hit and losing a life.

We conclude that these features were not learned because in Sonic 2 and Sonic 3&K, a side character follows Sonic around and produces many of the same sound effects while autonomously interacting with his local environment. This means that the jump sound or defeating a enemy sound is often not attributed to Sonic, and therefore loses its predictive power.

Our agents also did not learn to differentiate between consonant and dissonant sounds, or along any other axis of Western classical music theory explained earlier. This was expected, since the sounds in the Sonic games are not plentiful or diverse enough to be able to learn beyond simple classification. These theories should play a greater role if/when a large unsupervised audio model is trained on massive amounts of Western audio data and fine tuned on videogame RL tasks, similar to how BERT [28] and GPT-3 [29] have transformed the field of NLP.

5.3.4 Learned Video Features

None of this detailed audio analysis is to say that our audio agents have weaker visual components. In fact, Agent 5 is capable of navigating multiple multi-step visual challenges with what appears to be limited overfitting. This suggests that the audio component can be purely supplemental to the video component, although as mentioned previously, the agent may confuse itself in compounding ways when it chooses to assign reward responsibility to the wrong component.

6. CONCLUSION

The diverse nature of our learned audio features supports some of our original hypotheses, namely that environmental audio can reinforce existing visual ideas, and call attention to cues that are missed by the visual component, or simply non visual. We can expect that the supplementary role of our learned audio features, combined with their natural transfer ability, would extend beyond our work on *Sonic The Hedgehog*.

As video games become increasingly realistic, we expect that agents with the ability to process both audio and video will be the ones to achieve state of the art results on RL benchmarks. For future work, it would be interesting to see how audio features transfer across different games (say, Sonic and Mario), and how state of the art Atari models such as Agent 57 [30] might perform when given access to audio and games with richer audio/video components.

Overall, we hope that this work provides an insightful application to multi-modal videogame reinforcement learning, and motivates further such research in this field.

7. ACKNOWLEDGEMENTS

I would like to express my gratitude for the contributions of Eran Egozy, Professor of the Practice in Music Technology at MIT. He was an invaluable advisor and mentor, and should be considered as a secondary author for this work.

8. REFERENCES

[1] B. Langkjær, “Making fictions sound real-on film sound, perceptual realism and genre,” *MedieKultur: Journal of media and communication research*, vol. 26, no. 48, pp. 13–p, 2009.

[2] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman, “Gotta learn fast: A new benchmark for generalization in rl,” *arXiv preprint arXiv:1804.03720*, 2018.

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.

[4] D.-K. Kim, S. Omidshafiei, J. Pazis, and J. P. How, “Crossmodal attentive skill learner: learning in atari and beyond with audio–video inputs,” *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 1, p. 16, 2020.

[5] 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*, vol. 47, pp. 253–279, 2013.

[6] R. Kaplan, C. Sauer, and A. Sosa, “Beating atari with natural language guided reinforcement learning,” *arXiv preprint arXiv:1704.05539*, 2017.

[7] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, “Multimodal deep learning,” in *ICML*, 2011.

[8] S. Poria, E. Cambria, N. Howard, G.-B. Huang, and A. Hussain, “Fusing audio, visual and textual clues for sentiment analysis from multimodal content,” *Neurocomputing*, vol. 174, pp. 50–59, 2016.

[9] F. Henkel, S. Balke, M. Dorfer, and G. Widmer, “Score following as a multi-modal reinforcement learning problem,” *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, 2019.

[10] D. Battaglino, L. Lepauloux, N. Evans, F. Mougins, and F. Biot, “Acoustic scene classification using convolutional neural networks,” *IEEE AASP Challenge on Detec*, 2016.

[11] S. H. Bae, I. Choi, and N. S. Kim, “Acoustic scene classification using parallel combination of lstm and cnn,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016, pp. 11–15.

[12] J. Salamon and J. P. Bello, “Deep convolutional neural networks and data augmentation for environmental sound classification,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[13] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, “Polyphonic sound event detection using multi label deep neural networks,” in *2015 international joint conference on neural networks (IJCNN)*. IEEE, 2015, pp. 1–7.

[14] G. Parascandolo, H. Huttunen, and T. Virtanen, “Recurrent neural networks for polyphonic sound event detection in real life recordings,” in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 6440–6444.

[15] E. Cakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional recurrent neural networks for polyphonic sound event detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[16] K. Hevner, “Experimental studies of the elements of expression in music,” *The American Journal of Psychology*, vol. 48, no. 2, pp. 246–268, 1936.

[17] Sabbi Lall, “Universal musical harmony,” "http://news.mit.edu/2020/universal-musical-harmony-0701", 2020, [Online; accessed 14-August-2020].

[18] W. Apel, *The Harvard dictionary of music*. Harvard University Press, 2003.

[19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[20] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[22] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*, 2016, pp. 1928–1937.

[24] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” 2014.

[25] I. Kostrikov, “Pytorch implementations of reinforcement learning algorithms,” https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

[26] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, "Score-cam: Score-weighted visual explanations for convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 24–25.

[27] F. Nadeem, "Multi-modal reinforcement learning with videogame audio to learn sonic features," Master's thesis, Massachusetts Institute of Technology, 2020.

[28] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[29] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[30] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell, "Agent57: Outperforming the atari human benchmark," *arXiv preprint arXiv:2003.13350*, 2020.