Learning Play Styles on Super Mario Bros: Generating Levels by Augmenting Human Data via Imitation Learning

Michael Guevarra[*], Justin Stevens[*], Shuwei Wang[*]

Abstract

AAAI creates proceedings, working notes, and technical reports directly from electronic source furnished by the authors. To ensure that all papers in the publication have a uniform appearance, authors must adhere to the following instructions.

Introduction

Procedural Content Generation (PCG) has been used a lot in recent years to generate content for games. One popular game is Mario, which has a plethora of papers (Dahlskog and Togelius 2014; Summerville and Mateas 2016) and even consists of an entire framework for applying AI to the game (Karakovskiy and Togelius 2012). One open research area in applying PCG is that of generating content that is tailored to players through personalization (Yannakakis and Togelius 2015) (Summerville et al. 2016). In personalization, a model of the players is learned based on how the player engages with the game (Yannakakis et al. 2013).

One limitation of player modeling is it's very cumbersome to have users automatically play through levels and give lots of examples of their desired playstyle. However, many PCG approaches that use machine learning are very data hungry and require lots of data to successfully train. Therefore, it'd be ideal if we could augment the data that a player provides by using imitation learning to imitate a user's playstyle to provide more data to a PCG system (Hussein et al. 2017). In doing so, we'd be able to have a PCG system that can more easily adapt to a wide variety of playstyles and also use more powerful PCG systems that are data-hungry such as transformers (Vaswani et al. 2017).

In our work, we offer several novel contributions to generating personalized content for players:

- Integrating the Mario-AI framework with OpenAI gym to extract player paths and metadata from a human or agent's playthrough of a Mario level.
- Experiments demonstrating that imitation learning agents can generate similar player paths to a given user capturing a distinct play style.
- Applying transformers to generate Mario levels.
- An evaluation framework for evaluating whether generated levels in a game match a player's playstyle.

[*]These authors contributed equally.

Related Work

Shway to take the lead on writing

- NATURE paper using RL to solve atari games

... we just copy the CNN + frame stacking architecture

- maybe a survey on different Learning from Demonstration (LfD) techniques (BC, GAIL, AIRL)

- imitation learning in video games (link that matthew sent us in his first critique)

Methodology

Everyone to work on the parts they contributed to

Mario-AI Framework as a Gym Environment

- converting Mario-AI framework into a gym environment

... explain benefits (custom levels, a* agents, fwd model + game model, game stats, privileged information like invisible tiles etc.)

... implementation details

- state representation

... [figure] to visualize 11x11 window around mario + frame stacking (nature paper)

... [figure or table] to show mapping of tile number to grayscale

- alternative design choices / experiments / justifications

... fwd model vs game model (fwd better on ordinary RL but worse when doing LfD)

... Mario-centred vs screen centred

... 11x11 window vs other window sizes

... alternative CNN architectures (1-hot encoding, changes to layers/kernels) ... Frame stacking/skipping (skipping good in ordinary RL but very bad in LfD, tried to apply rule-based stuff on skipped actions but did not work well, too much data loss) ... random starts (location)

... sticky actions (stochasticity) *** Explain how mario is very deterministic (pros and cons, maybe allude to this in discussion instead), tried random actions but diluted the source playstyle data

... fixed horizons (blit screen with a pixel vs keep last obs to show good/bad of dying/completing early)

Imitation Learning

- Trajectory collection process (can handle user input or agent input), + preprocessing details
- *** At the same time we could throw in implementation details of path collection that happens at this same step
- stable baselines 3
- BC, GAIL, AIRL
- Q: should subsectioning the levels go here or in evaluation?

The Decoder-only Transformer Level Generator

Experiments

Not sure if we want this as its own section or a subsection inside either methodology or evaluation but for sure need a dedicated area to outline our various experiments, we can label each experiment so we can refer to it easier in the rest of the paper

- Experiment 1 evaluating generator percent completable levels (random vs biased prompts)
- Experiment 2: evaluating generator plagiarism (completionist vs speedrunner)
- Experiment 3: comparison(s) between completionist and speedrunner outputs
- Experiment 4: BC vs GAIL vs AIRL
- Experiment 5: BC - testing agent on new levels (new subsections not levels)
- Experiment 6: BC - path variations from 1 sample

Experiment 1

- explain

Evaluation

One of the goals of our project was to see if the generator is adapting to the individual user's playstyles.
- establish why edit distance is a good indicator and why we use it throughout everything
- Explain how we evaluate each playstyle?
... Speedrunner
... Completionist
... Enemy-stomper (lets go with percent of total enemys killed in the level + full completion)

Results

Everyone to work on the parts they contributed to

Imitation Learning

- experiment 4: BC vs GAIL vs AIRL on lvl-1 (subsections)
... EXPLAIN the setup and why (3 different types of models, help break problem down, do not affect input to transformer negatively, can also act as a proxy of being 3 different levels, Edit distance for consistency)
... [figure] Can still present a training curve of the different algorithms if we want
... [figure] of paths similar to in final presentation

- experiment 5: BC with 5-samples (or 1 only depending on results), matrix running agents on level subsections not trained on (i.e. matrix that shows lvl-11 agent on lvl-11, lvl-12, lvl-13, etc.) – glimpse at agent running on 'new' level not seen without us biasing by creating the new level ourselves
... [table] showing above results in a matrix

- experiment 6: BC agent with 1 sample, providing more than 1 sample on the same and on new level. Maybe test if adding sticky action can improve variation (stdev) but retain goals
... Although we do not fully integrate path to our generator, we are exploring how we can provide variations of path data that still retain underlying playstyle!
- [figure] visualizing more than 1 path on a level to show variation
- [table] next to it with tabular results (avg edit distance +/- stdev)

Discussion

Primarily Michael
- Summarize out takeaways and restate them in the context of a class deliverable. Mainly talk about our limited time and what we were able to accomplish vs what we wanted to accomplish
- most 'discussion' likely happens in result section
- other noteworthy 'cool' stuff we noticed that arent a takeaway

Future Work

- should spend lots of time here explaining what our current progress could lead towards, and the prospective benefits/novelty of future work

- tuning of RL algorithms (ability of AIRL to generalize more, recover the learned reward function,)
- revisit 1-hot encoding state space + other implementation optimizations

Conclusion

Primarily Shway

Group Member Breakdown

References

Dahlskog, S.; and Togelius, J. 2014. A multi-level level generator. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*, 1–8. IEEE.

Hussein, A.; Gaber, M. M.; Elyan, E.; and Jayne, C. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2): 1–35.

Karakovskiy, S.; and Togelius, J. 2012. The mario ai benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1): 55–67.

Summerville, A.; Guzdial, M.; Mateas, M.; and Riedl, M. O. 2016. Learning player tailored content from observation: Platformer level generation from video traces using lstms. In *Twelfth artificial intelligence and interactive digital entertainment conference*.

Summerville, A.; and Mateas, M. 2016. Super mario as a string: Platformer level generation via lstms. In *Proceedings of the 1st International Joint Conference of DiGRA and FDG*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yannakakis, G. N.; Spronck, P.; Loiacono, D.; and André, E. 2013. Player modeling.

Yannakakis, G. N.; and Togelius, J. 2015. Experience-driven procedural content generation. In *Proceedings of the International Conference on Affective Computing and Intelligent Interaction (ACII)*, 519–525. IEEE.