SOUTHWESTERN UNIVERSITY

Evolving Mega Man Levels Using Generative Adversarial Networks

Benjamin Capps

Introduction

Generative Adversarial Networks (GANs) are a way of training a computer to generate fake samples based on real examples. GANs are used to generate Mega Man levels that are similar to but distinct from levels in the original game. Due to the snaking pattern of levels, several GANs were used in order to handle different types of level segments, resulting in levels whose design is more human-like and better organized.

Mega Man

Mega Man is a side-scrolling game that requires the player to fight enemies and jump carefully to reach the end of each level.

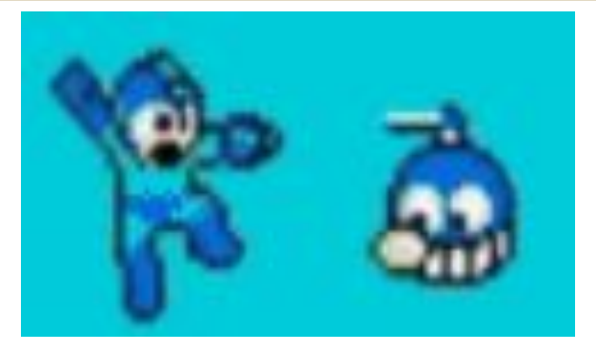


Figure 1: Mega Man Gameplay Snapshot

Generative Adversarial Networks

- Used for generating novel results with the same style as the training data.
- Previous work [1] supports the idea that GANs can generate interesting video game level.
- GANs were used to generate new levels in Mega Man

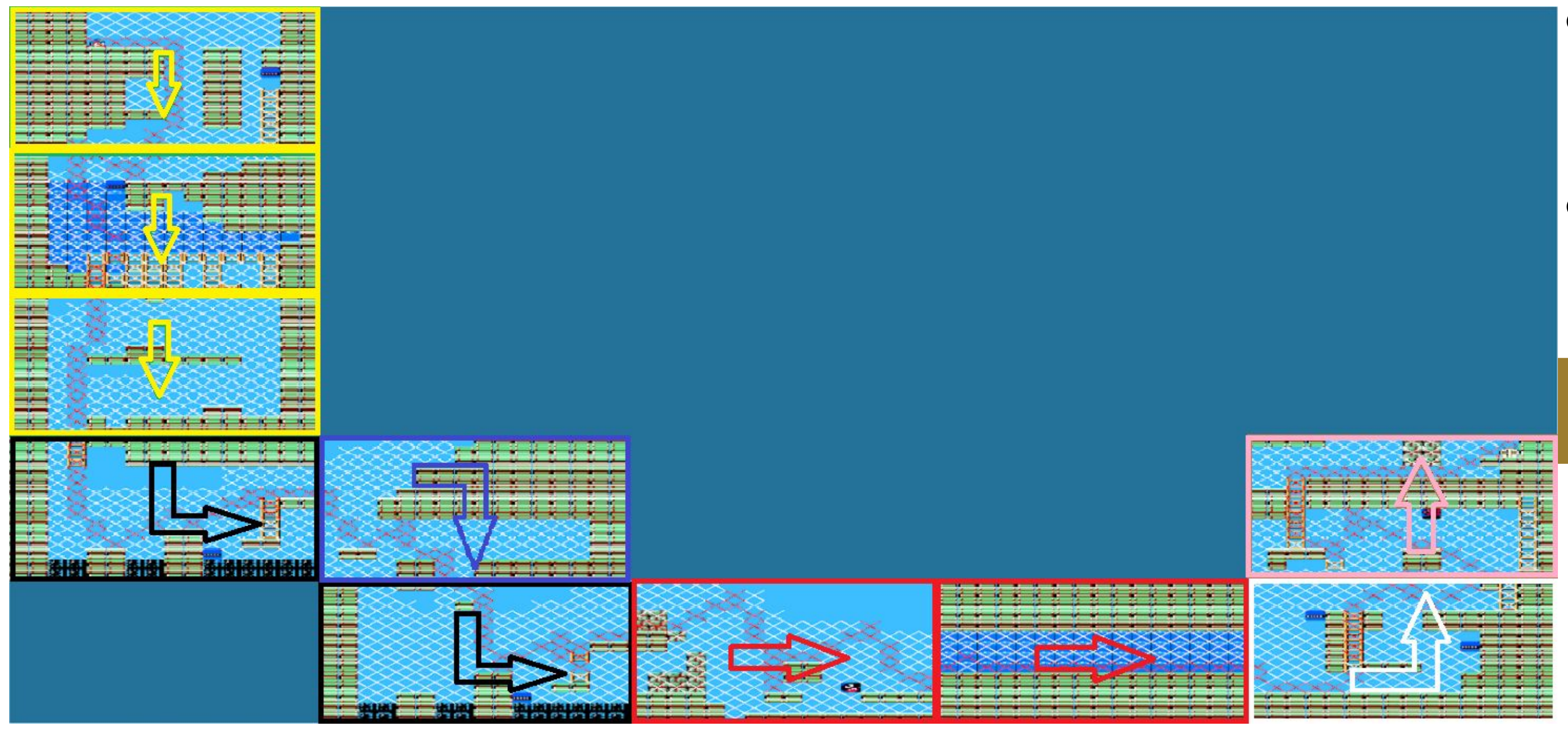
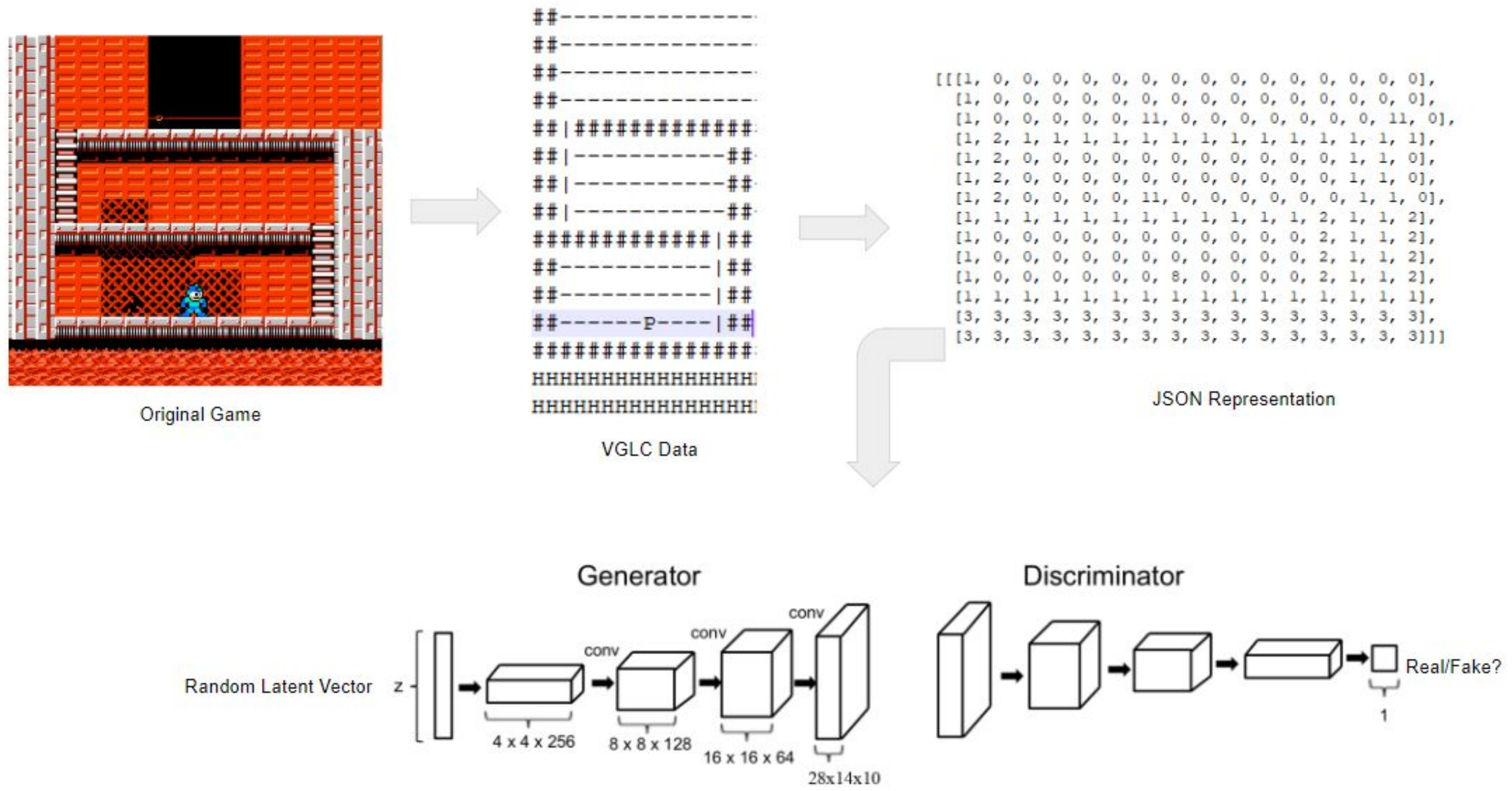


Figure 4: A level generated by the SevenGAN



```
##--------------
##--------------
##--------------
##--------------
##|#############
##|-----------##
##|-----------##
##|-----------##
#############|##
##-----------|##
##-----------|##
##------P----|##
################
HHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHH
```

```
[[[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
  [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
  [1, 0, 0, 0, 0, 0, 11, 0, 0, 0, 0, 0, 0, 0, 11, 0],
  [1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
  [1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0],
  [1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0],
  [1, 2, 0, 0, 0, 0, 11, 0, 0, 0, 0, 0, 0, 1, 1, 0],
  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2],
  [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 2],
  [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 2],
  [1, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 2, 1, 1, 2],
  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
  [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
  [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]]]
```

Figure 2: Takes in data from the original game and converts it to an integer encoding, then trains the GAN.

Evolving Mega Man Levels

- Data was taken from the VGLC [2] and used to train the GAN(s)
- Evolving numeric latent vectors and convert into levels to input into the GAN
  - The GAN converts latent vectors into level segments
- 10 segments per level
  - Segments placed vertically or horizontally adjacent to form snaking pattern.
- Multiple fitness functions for evolution:
  - Solution path length (determined using A*)
  - Connectivity (percentage of reachable locations in level)

Methods for Level Generation

- One GAN
  - All of the data in one training set
- Seven GANs
  - Each GAN is assigned one segment type (up, down, horizontal, and four corners)

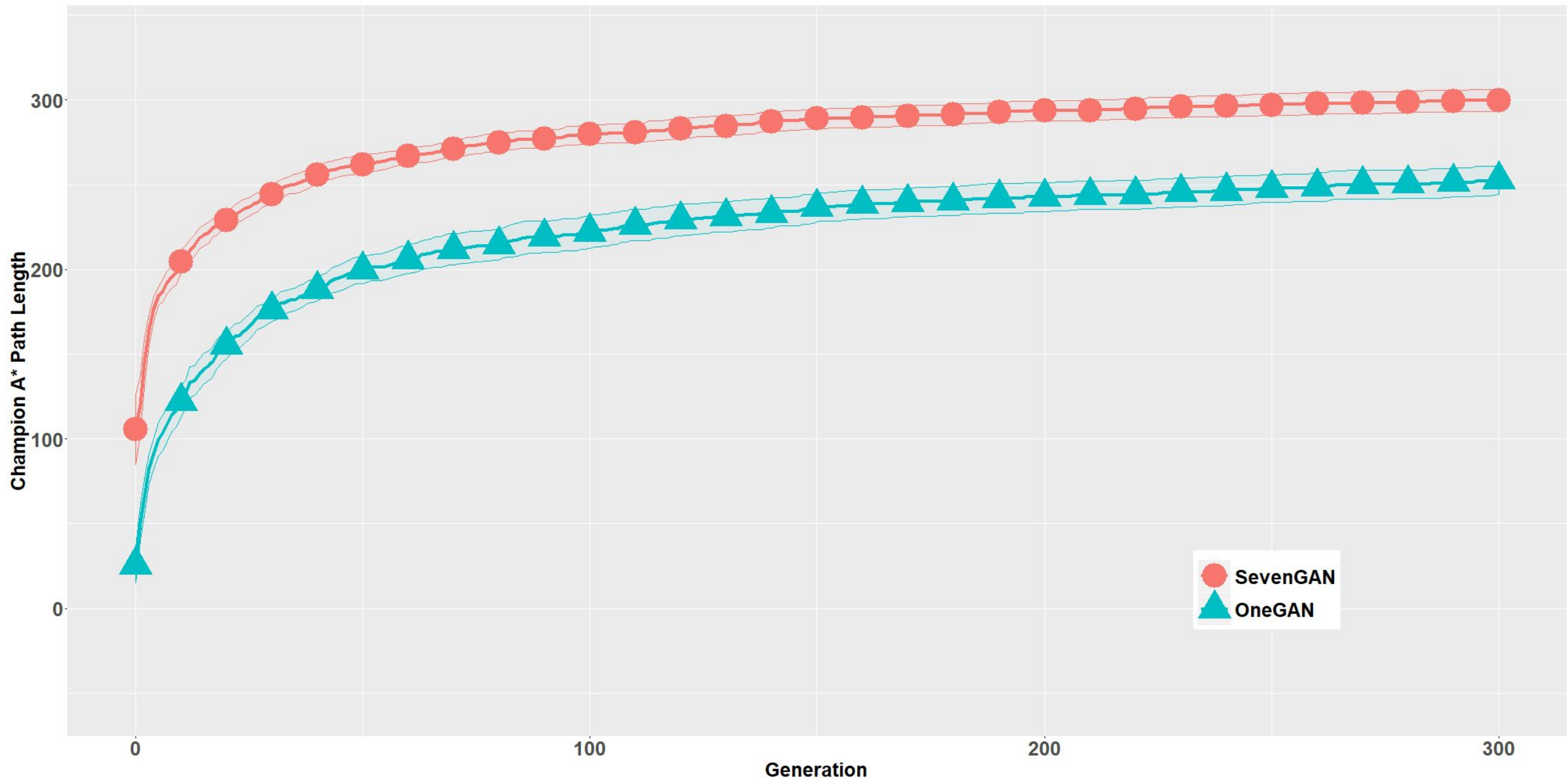


Figure 3: Average scores A* path length across levels with 10 segments. Longer A* path length means that it takes longer to complete the level.

Results

- A* path lengths are significantly longer with SevenGAN than OneGAN
  - Longer A* paths utilize more space in the levels, making them longer and harder to beat.
- SevenGAN levels look more natural
  - Each GAN has an individual set of data and does not communicate with the other GANs to make perfectly smooth transitions from one segment to another.
- OneGAN levels look more chaotic
  - This randomness is due to having all data in one GAN

References

[1] Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M. Lucas, Adam M. Smith, and Sebastian Risi (2018). Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network, Proceedings of the Genetic and Evolutionary Computation

[2] Summerville, A., Sam Snodgrass, M. Mateas and S. Ontañón (2016). “The VGLC: The Video Game Level Corpus.” Proceedings of the 7th Workshop on Procedural Content Generation.