MarioGPT: Open-Ended Text2Level Generation through Large Language Models

Viet Nhat Nguyen (21520378)

Thi Lien Le (21522282)

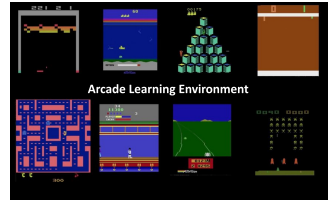
July 6, 2023

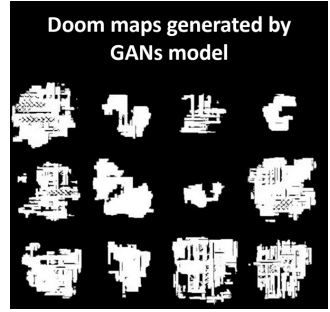
UNIVERSITY OF INFORMATION TECHNOLOGY

Table of Contents

PCG and Machine Learning

1 Introduction

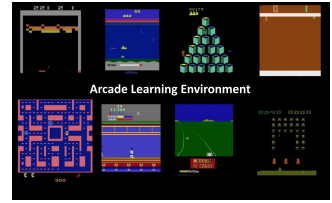


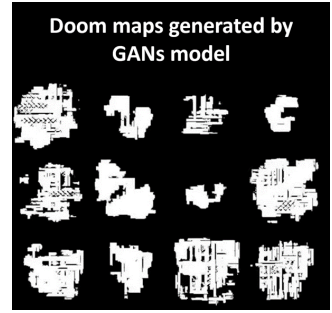


PCG and Machine Learning

1 Introduction

- PCG for evaluating generalization capabilities of trained agents.
  - Offer more diverse and challenging scenarios.
  - A better way to test the adaption and generalization of trained agents.

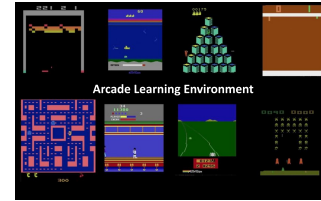


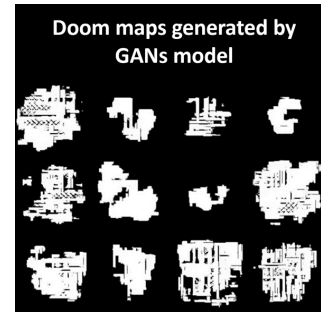


PCG and Machine Learning

- PCG for evaluating generalization capabilities of trained agents.
  - Offer more diverse and challenging scenarios.
  - A better way to test the adaption and generalization of trained agents.

- Incorporation of machine learning-based approaches in PCG systems.
  - GANs for generating levels (Doom, Super Mario Bros)





Challenges

- **Costly** searching in the latent space of neural networks.
- Desire to **directly condition** a generator for creating levels with specific properties, ideally in natural language. (**prompt**)

Contributions

- Introduction of MarioGPT, a text-to-level model generating Mario levels based on natural language prompts.

Contributions

- Introduction of MarioGPT, a text-to-level model generating Mario levels based on natural language prompts.
- Combining MarioGPT with novelty search for producing diverse levels.

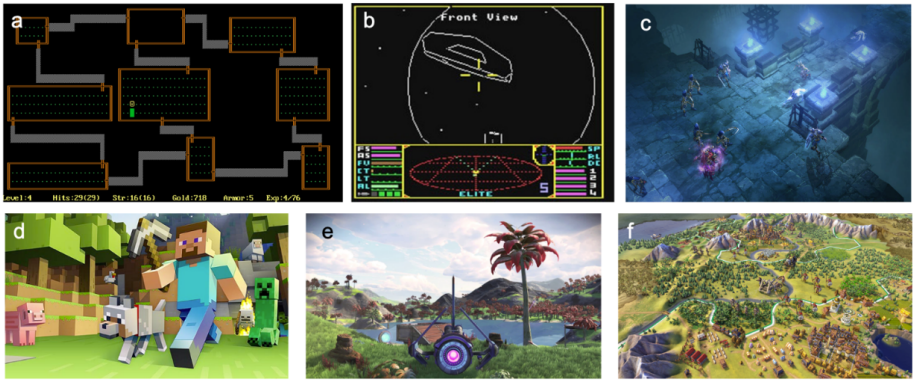
Table of Contents

2 Background and Related Work

Procedural Content Generation (PCG)

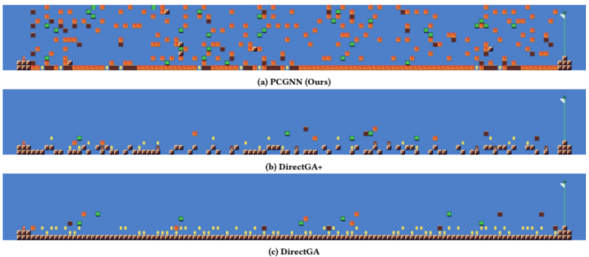
- Procedural Content Generation (PCG) refers to techniques that can automatically create game content (e.g. levels, maps, or characters).
  - Increasing the replayability
  - Reducing production costs

Procedural Content Generation (PCG)

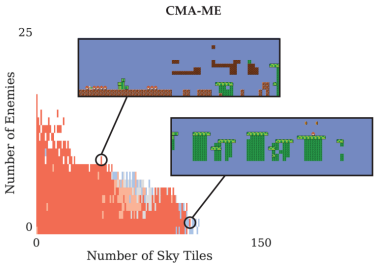


a) Rouge **b**) Elite **c**) Diablo III **d**) Minecraft **e**) No Man's Sky **f**) Civilisation VI

Neural Network-based Level Generation

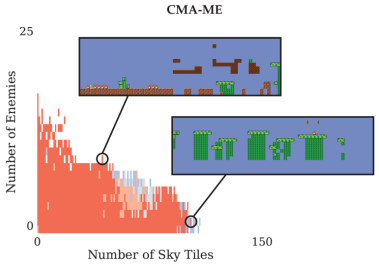


Using neural networks (Beukman et al.[1])



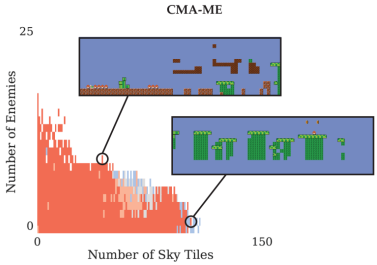
Searching in the latent space of GANs (Matthew et al.[2])

- Guided sampling of the latent space could result in a diverse set of levels.
- The abilities to control characteristics in generated levels.



Searching in the latent space of GANs (Matthew et al.[2])

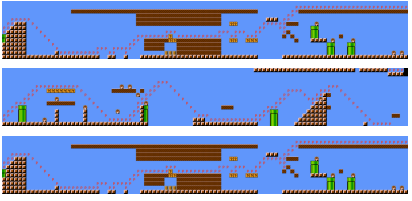
- Guided sampling of the latent space could result in a diverse set of levels.
- The abilities to control characteristics in generated levels.
- *Using A* agent to measure whether the generated levels were playable.*



Searching in the latent space of GANs (Matthew et al.[2])

Open-Endedness Paradigm

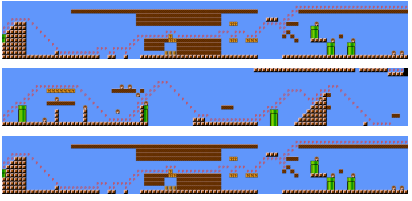
- The open-endedness paradigm focuses on algorithms that can produce infinite innovation.



Diversity but not-playable cases

Open-Endedness Paradigm

- The open-endedness paradigm focuses on algorithms that can produce infinite innovation.
- *Must balance the hard task of generating content with **diversity** as wll as **playability***

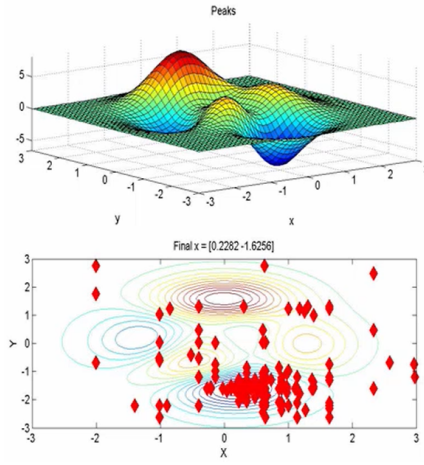


Diversity but not-playable cases

Genetic Algorithms

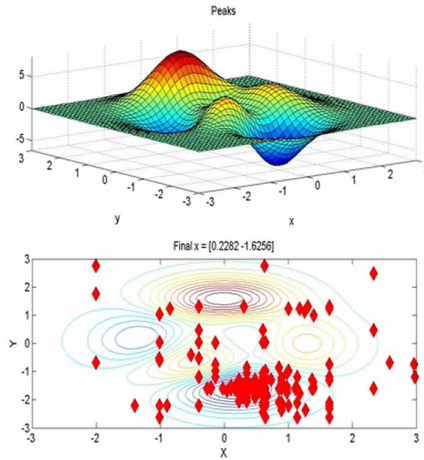
2 Background and Related Work

- Uses concepts from *evolutionary biology*



Genetic Algorithms

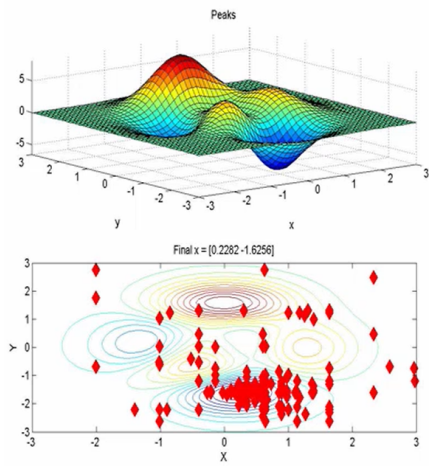
- Uses concepts from *evolutionary biology*
- Start with an initial generation of candidate solutions that are tested against the objective function



Genetic Algorithms

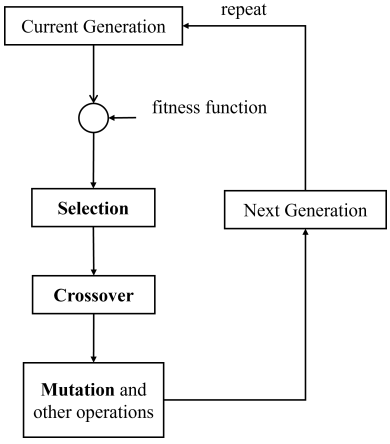
2 Background and Related Work

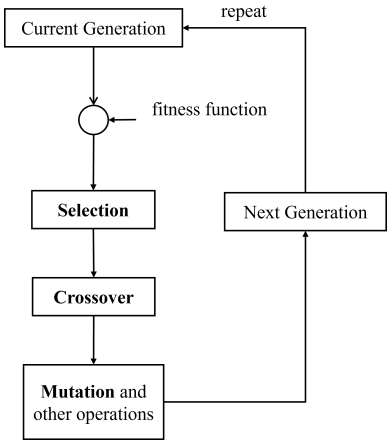
- Uses concepts from *evolutionary biology*
- Start with an initial generation of candidate solutions that are tested against the objective function
- Subsequent generations evolve from the 1st through *selection*, *crossover* and *mutation*



Genetic Algorithms

2 Background and Related Work

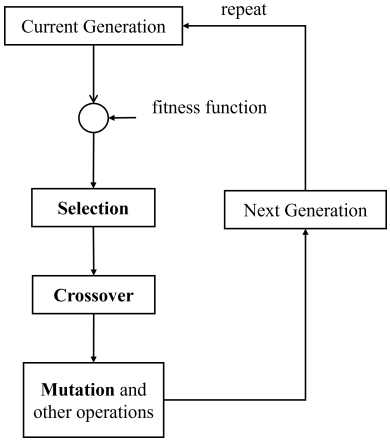




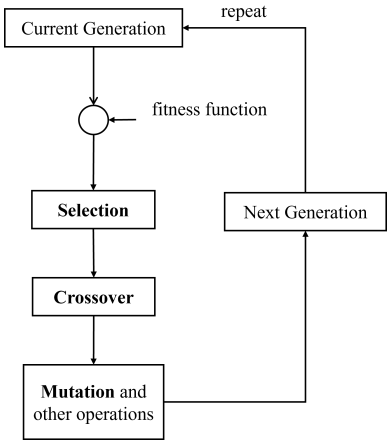
- First generation is arbitrarily, or $x = random()$

Genetic Algorithms

2 Background and Related Work



- First generation is arbitrarily, or $x = random()$
- Evaluate their *fitness*

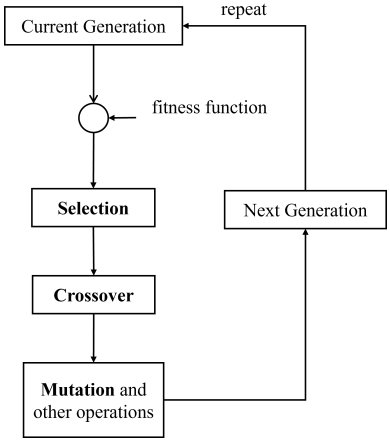


- First generation is arbitrarily, or $x = random()$
- Evaluate their *fitness*

$$fitness(x) = f(x)$$

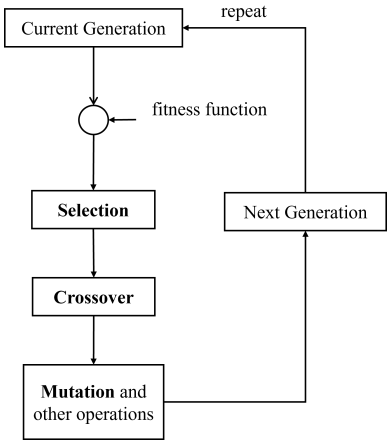
Genetic Algorithms

2 Background and Related Work

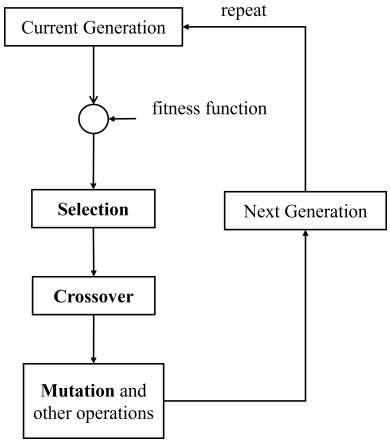


- First generation is arbitrarily, or $x = random()$
- Evaluate their *fitness*
- *Select* members of the population:

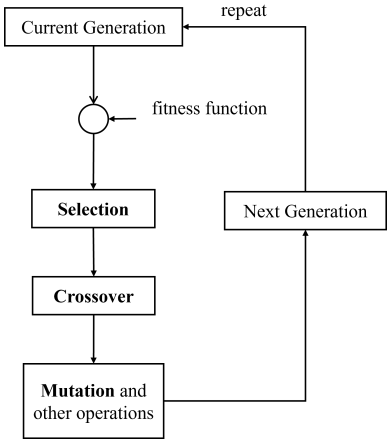
Genetic Algorithms



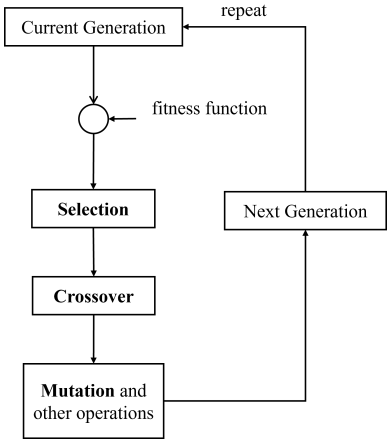
- First generation is arbitrarily, or $x = random()$
- Evaluate their *fitness*
- *Select* members of the population:
  - Deterministic



- First generation is arbitrarily, or $x = random()$
- Evaluate their *fitness*
- *Select* members of the population:
  - Deterministic: Elitism
    $\text{argmax}_x fitness(x)$

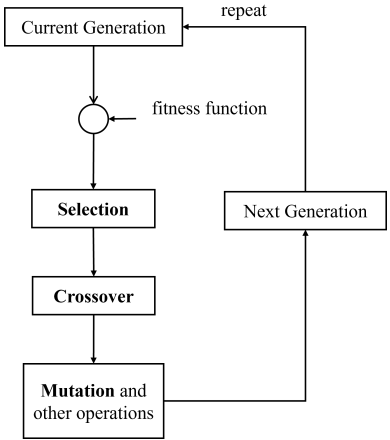


- First generation is arbitrarily, or $x = random()$
- Evaluate their *fitness*
- *Select* members of the population:
  - Deterministic
  - Stochastic



- First generation is arbitrarily, or $x = random()$
- Evaluate their *fitness*
- *Select* members of the population:
  - Deterministic
  - Stochastic: Roulette wheel selection

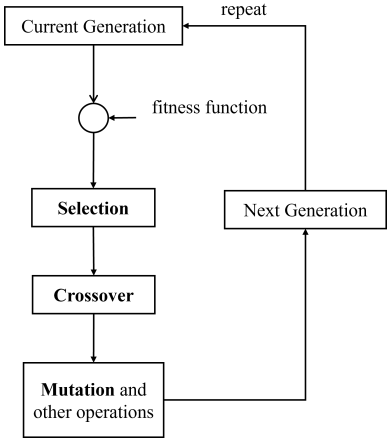
$$P(x) = \frac{fitness(x)}{\sum fitness(x)}$$



- First generation is arbitrarily, or $x = random()$
- Evaluate their *fitness*
- *Select* members of the population:
  - Deterministic
  - Stochastic

Genetic Algorithms

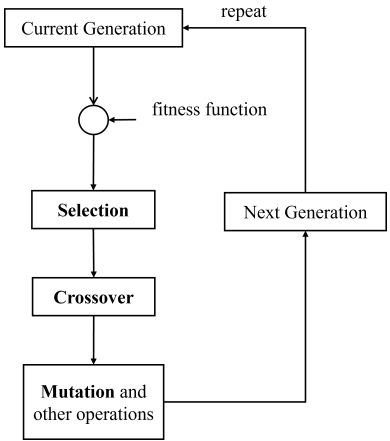
2 Background and Related Work



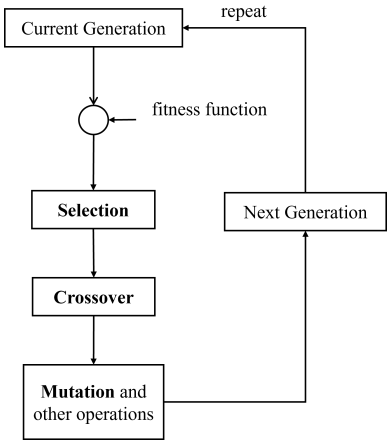
- First generation is arbitrarily, or $x = random()$
- Evaluate their *fitness*
- *Select* members of the population:
  - Novelty Selection

Genetic Algorithms

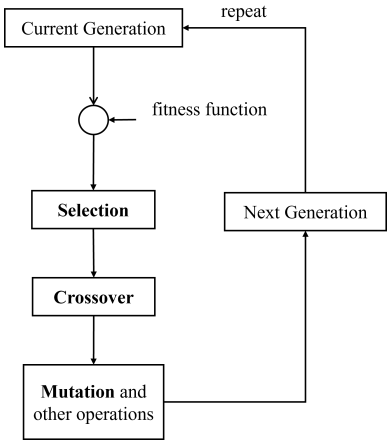
2 Background and Related Work



- First generation is arbitrarily, or $x = random()$
- ~~Evaluate their *fitness*~~
- *Select* members of the population:
  - Novelty Selection

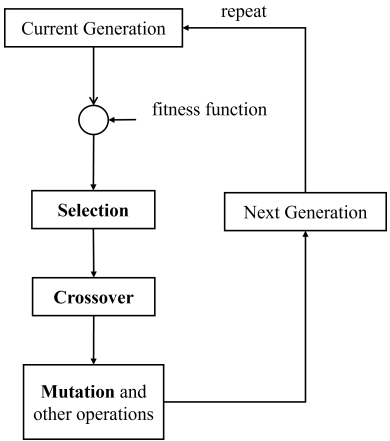


- First generation is arbitrarily, or $x = random()$
- ~~Evaluate their *fitness*~~
- *Select* members of the population:
  - Novelty Selection
- Implement *crossover* operation on the reproduced chromosomes



- First generation is arbitrarily, or $x = random()$
- ~~Evaluate their *fitness*~~
- *Select* members of the population:
  - Novelty Selection
- Implement *crossover* operation on the reproduced chromosomes

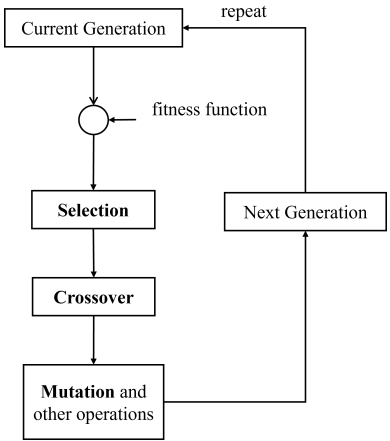
$$Crossover(x_1, x_2) = y$$



- First generation is arbitrarily, or $x = random()$
- ~~Evaluate their *fitness*~~
- *Select* members of the population:
  - Novelty Selection
- Implement *crossover* operation on the reproduced chromosomes

$$Crossover(x_1, x_2) = y$$

- Execute *mutation* operation with low probability

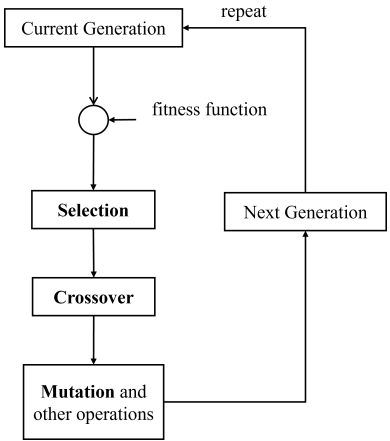


- First generation is arbitrarily, or $x = random()$
- ~~Evaluate their *fitness*~~
- *Select* members of the population:
  - Novelty Selection
- Implement *crossover* operation on the reproduced chromosomes

$$Crossover(x_1, x_2) = y$$

- Execute *mutation* operation with low probability

$$Mutation(y) = y', \text{ with } p_m$$



- First generation is arbitrarily, or $x = random()$
- ~~Evaluate their *fitness*~~
- *Select* members of the population:
  - Novelty Selection
- Implement *crossover* operation on the reproduced chromosomes

  $$Crossover(x_1, x_2) = y$$
- Execute *mutation* operation with low probability

  $$Mutation(y) = y', \text{ with } p_m$$
- *Allow the integration of multiple objectives*

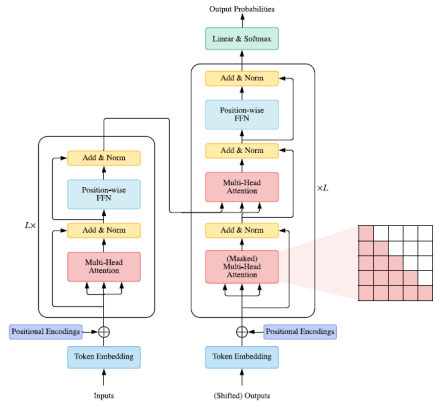
Sequence Modelling and Transformer

- Classic approaches to sequence modelling are RNNs (Rumelhart et al.[4]) and LSTM networks (Hochreiter et al. [3])
  - fading memory
  - limited scalability
- Transformers (Vaswani et al.[5]) can address both challenges, can be parallelized and achieve good scalability!

Sequence Modelling and Transformers

2 Background and Related Work

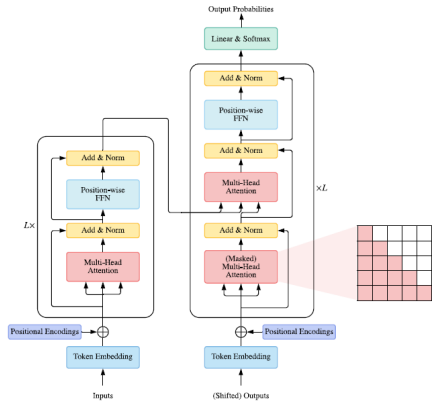
- "Attention is all you need" (Vaswani et al., 2017)
- **Encoder**:
- **Decoder**:



Sequence Modelling and Transformers

2 Background and Related Work

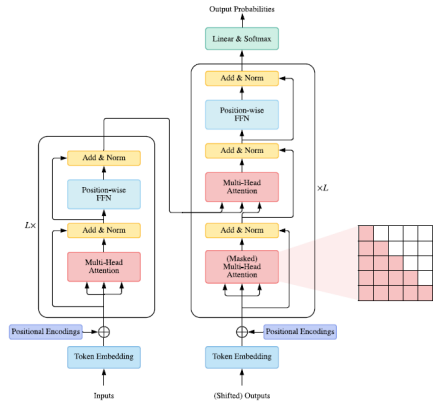
- "Attention is all you need" (Vaswani et al., 2017)
- **Encoder**:
  - N identical layers (Multi-Head Attention and FFN)
  - Residual connections
- **Decoder**:



Sequence Modelling and Transformers

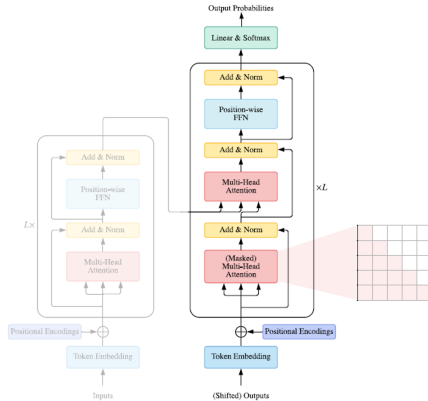
2 Background and Related Work

- ”Attention is all you need” (Vaswani et al., 2017)
- **Encoder**:
  - N identical layers (Multi-Head Attention and FFN)
  - Residual connections
- **Decoder**:
  - N identical layers (Multi-Head Attention, FFN and Masked Multi-Head Attention)
  - Residual connections

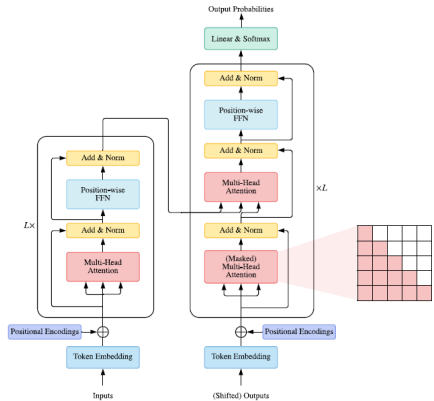


Sequence Modelling and Transformers

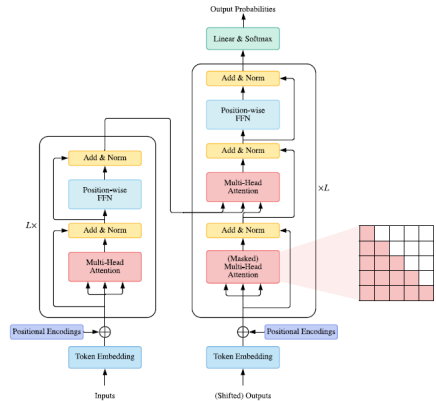
- ”Attention is all you need” (Vaswani et al., 2017)
- **Encoder**:
  - N identical layers (Multi-Head Attention and FFN)
  - Residual connections
- **Decoder**:
  - N identical layers (Multi-Head Attention, FFN and Masked Multi-Head Attention)
  - Residual connections
- *GPT consists of a stack of decoder layers.*



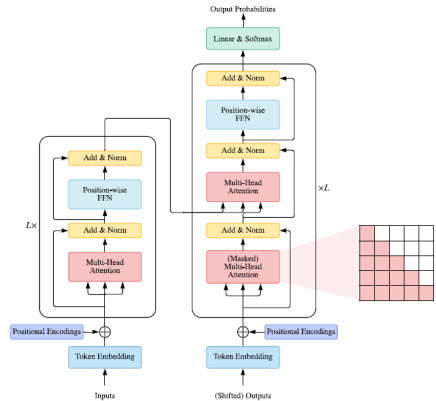
- Enable LLMs to learn from massive datasets



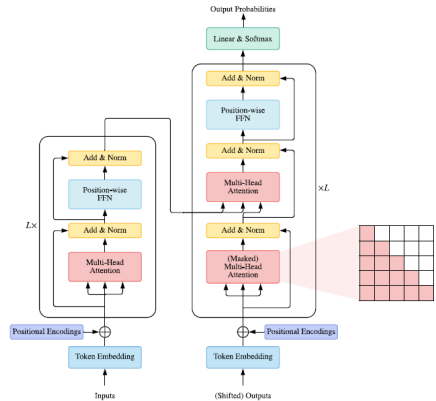
- Enable LLMs to learn from massive datasets
  - Using pre-trained model weights as a weight initialization for new tasks.



- Enable LLMs to learn from massive datasets
  - Using pre-trained model weights as a weight initialization for new tasks.
- *Evolution through Large Models* (Lehman et al., 2022)



- Enable LLMs to learn from massive datasets
  - Using pre-trained model weights as a weight initialization for new tasks.
- *Evolution through Large Models* (Lehman et al., 2022)
  - LLM diff model
  - used as a ”mutation operator”



Sequence Modelling and Transformers

2 Background and Related Work

- Enable LLMs to learn from massive datasets
  - Using pre-trained model weights as a weight initialization for new tasks.
- *Evolution through Large Models* (Lehman et al., 2022)
  - LLM diff model
  - used as a ”mutation operator”
- Produce incredibly diverse mutations, vary increasingly over the course of the GA!

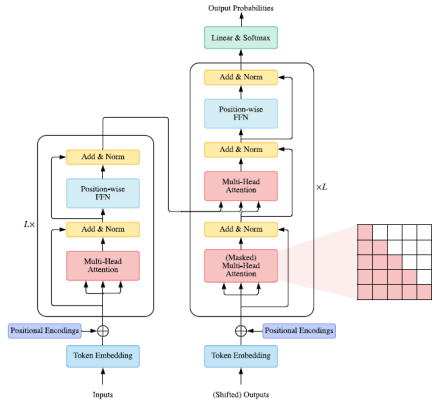


Table of Contents

3 Open-Ended Level Generation through LLMs

Level Representation

- Use the levels provided in the Video Game Level Corpus (Summerville et al., 2016)

Level Representation

- Use the levels provided in the Video Game Level Corpus (Summerville et al., 2016)
- Utilize a set of path-annotated levels, taken from Super Mario Bros. and Super Mario Bros.: The Lost Levels (in total 37 levels)

Level Representation

- Use the levels provided in the Video Game Level Corpus (Summerville et al., 2016)
- Utilize a set of path-annotated levels, taken from Super Mario Bros. and Super Mario Bros.: The Lost Levels (in total 37 levels)
  - Stitch selected levels together, make one giant level!

Level Representation

- Use the levels provided in the Video Game Level Corpus (Summerville et al., 2016)
- Utilize a set of path-annotated levels, taken from Super Mario Bros. and Super Mario Bros.: The Lost Levels (in total 37 levels)
  - Stitch selected levels together, make one giant level!
- Tokenize string representation into discrete values using Byte Pair Encoding.

Level Representation

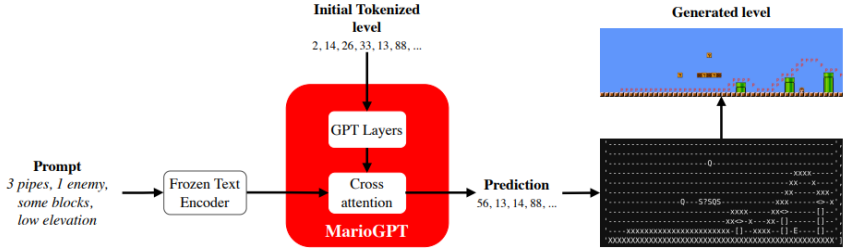
3 Open-Ended Level Generation through LLMs

Tile Type	Symbol	Visualization
Empty	-	
Unbreakable	X	
Breakable	S	
Question Block	? / Q	
Coin	o	
Enemy	E	
Left pipe top	<	
Right pipe top	>	
Left pipe lower	[	
Right pipe lower	]	
Cannon Top	B	
Cannon Body	b	
Path	x	

Unique Mario tiles (Sudhakaran et al., 2023)

MarioGPT Model

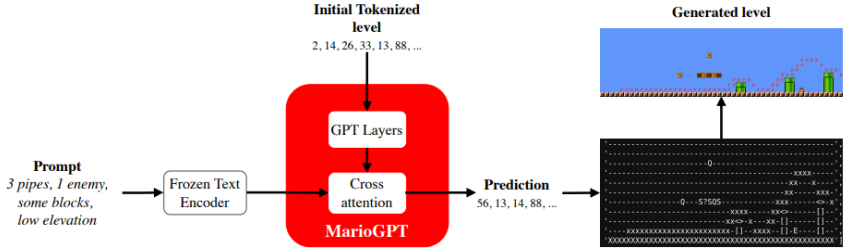
3 Open-Ended Level Generation through LLMs



- **DistilGPT2** (Li et al., 2021) is MarioGPT's backbone.

MarioGPT Model

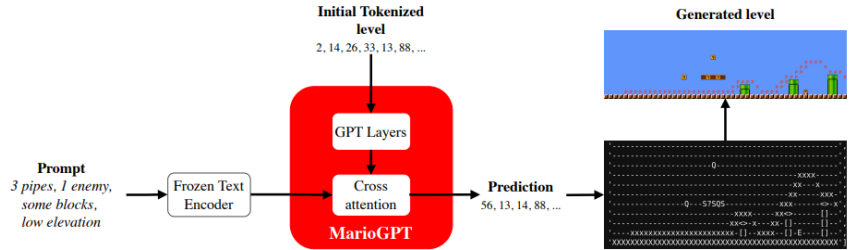
3 Open-Ended Level Generation through LLMs



- **DistilGPT2** (Li et al., 2021) is MarioGPT's backbone.
- **Cross attention** layer takes prompt information into account.

MarioGPT Model

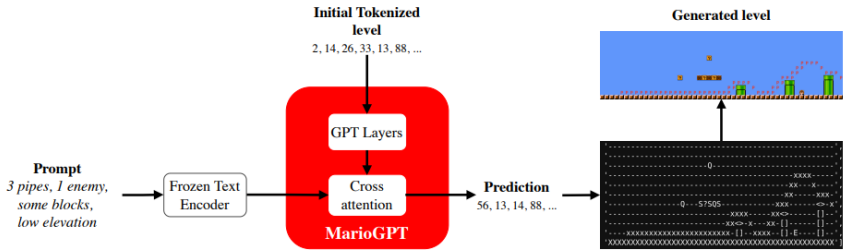
3 Open-Ended Level Generation through LLMs



- **DistilGPT2** (Li et al., 2021) is MarioGPT's backbone.
- **Cross attention** layer takes prompt information into account.
- 96M parameters of DistilGPT2 (86M) and Cross attention weights (10M).

MarioGPT Model

3 Open-Ended Level Generation through LLMs



- Prompts are encoded through BART (Lewis et al., 2019).
- Prompt's feature vector is combined with the actual level sequence in the cross attention layers.

MarioGPT Model

3 Open-Ended Level Generation through LLMs

- Prompts:
  - { *no*, *little*, *some*, *many*, *[0-1000]* } *pipes*
  - { *no*, *little*, *some*, *many*, *[0-1000]* } *enemies*
  - { *little*, *some*, *many*, *[0-1000]* } *blocks*
  - { *low*, *high* } *elevation*

MarioGPT Model

3 Open-Ended Level Generation through LLMs

- Prompts:
  - { *no*, *little*, *some*, *many*, *[0-1000]* } *pipes*
  - { *no*, *little*, *some*, *many*, *[0-1000]* } *enemies*
  - { *little*, *some*, *many*, *[0-1000]* } *blocks*
  - { *low*, *high* } *elevation*
- Example:
  - *"no pipes, many enemies, low elevation"*
  - *"many pipes, many enemies, many blocks"*

MarioGPT Model

3 Open-Ended Level Generation through LLMs

tile	no	little	some	many
pipes	0	1	2	5
enemies	0	1	3	7
blocks	0	50	75	176

Prompt Quantiles and corresponding counts within a 50 column window

MarioGPT Model

3 Open-Ended Level Generation through LLMs

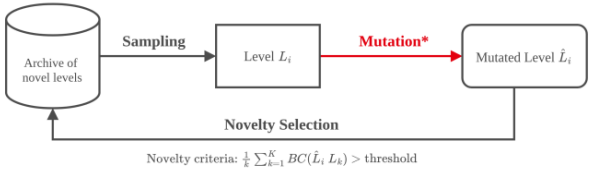


Table of Contents

4 Experiments and Results

Table of Contents

MarioGPT: Open-Ended Text2Level Generation through Large Language Models

Thank you for listening!

Any questions?