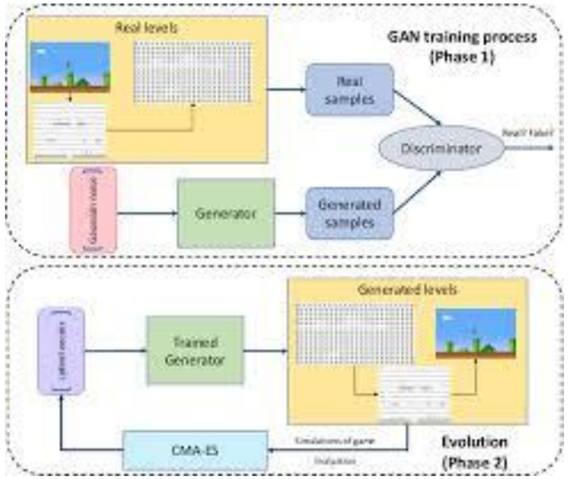
Sequential Segment-based Level Generation and Blending using Variational Autoencoders

Anurag Sarkar and **Seth Cooper**

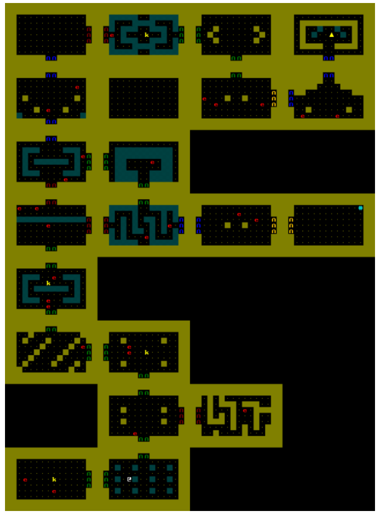
Northeastern University

Motivation

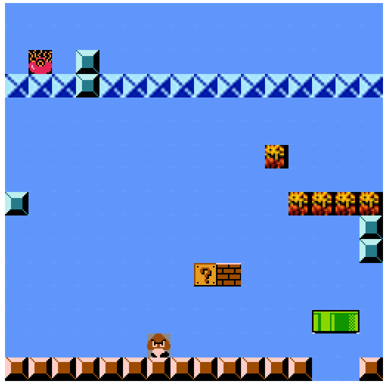
- Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have been used for generating platformer levels and dungeons via sampling, interpolation and evolution



Volz et al., 2017



Gutierrez and Schrum, 2020



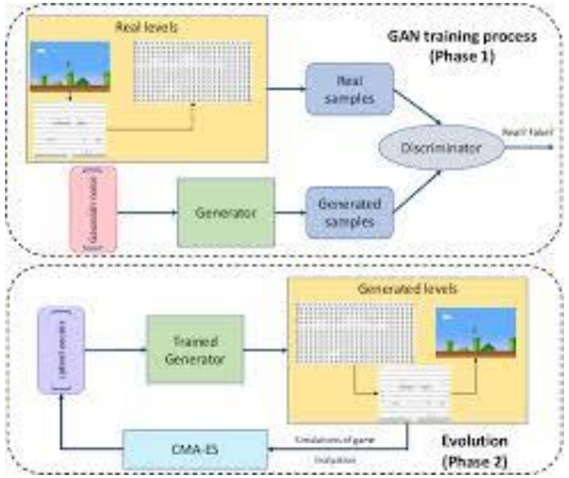
Sarkar, Yang and Cooper, 2019

Motivation

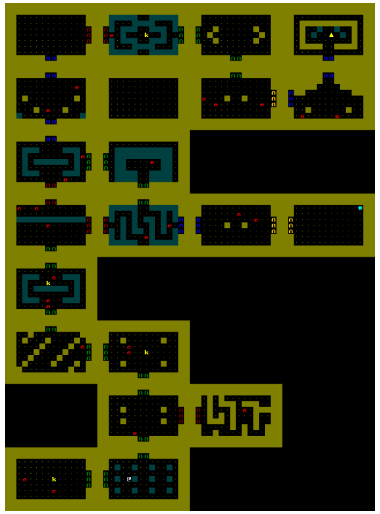
- Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have been used for generating platformer levels and dungeons via sampling, interpolation and evolution

- Work with fixed-size inputs and outputs

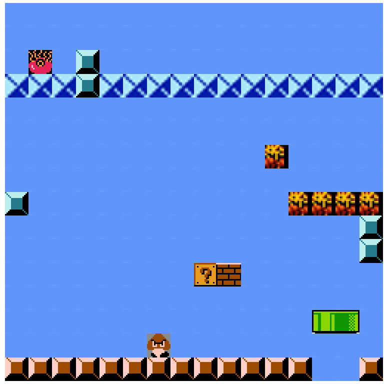
  --- necessitates generation by segment rather than by level



Volz et al., 2017



Gutierrez and Schrum, 2020



Sarkar, Yang and Cooper, 2019

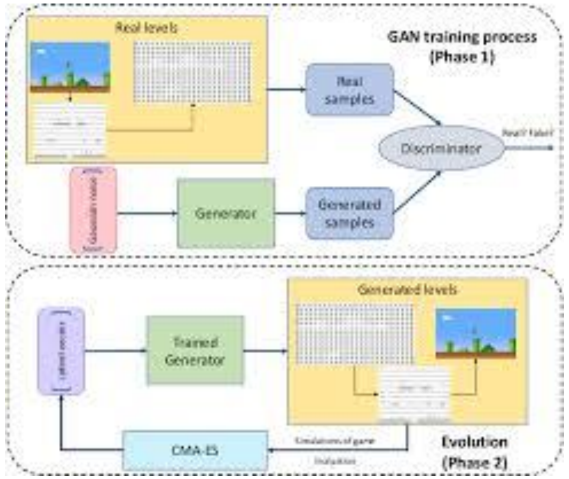
Motivation

- Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have been used for generating platformer levels and dungeons via sampling, interpolation and evolution

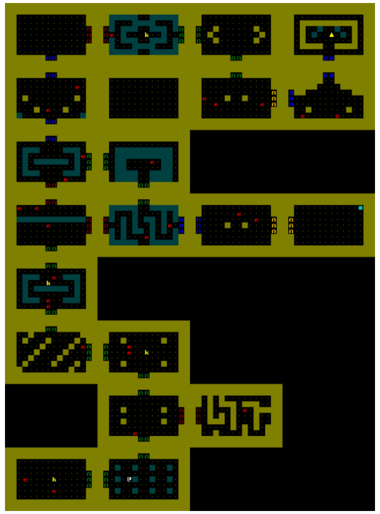
- Work with fixed-size inputs and outputs

  --- necessitates generation by segment rather than by level

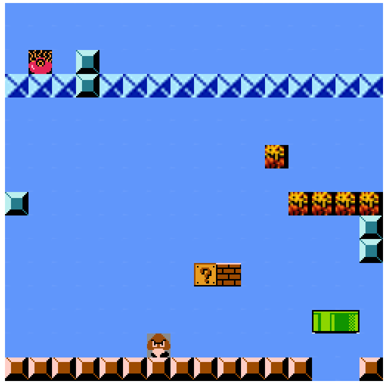
- Existing methods combine independently generated segments to form whole levels



Volz et al., 2017



Gutierrez and Schrum, 2020



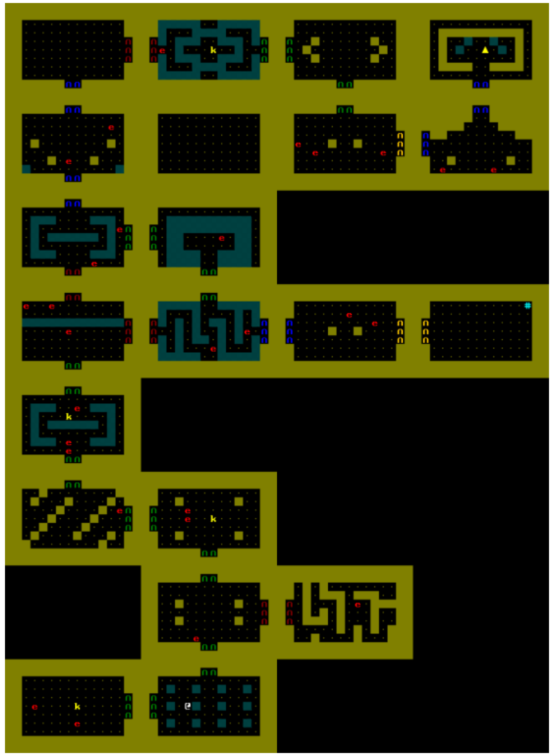
Sarkar, Yang and Cooper, 2019

Motivation

- Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have been used for generating platformer levels and dungeons via sampling, interpolation and evolution

- Work with fixed-size inputs and outputs
  --- necessitates generation by segment rather than by level

- Existing methods combine independently generated segments to form whole levels
  - Dungeons with discrete rooms



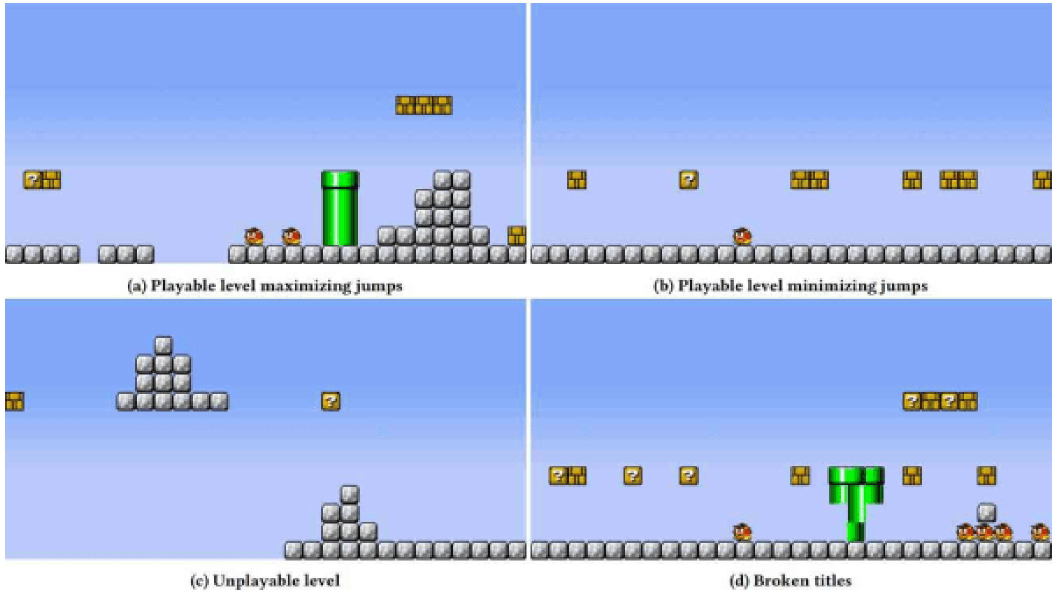
Gutierrez and Schrum, 2020

Motivation

- Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have been used for generating platformer levels and dungeons via sampling, interpolation and evolution

- Work with fixed-size inputs and outputs
  --- necessitates generation by segment rather than by level

- Existing methods combine independently generated segments to form whole levels
  - Dungeons with discrete rooms
  - Okay but not ideal for Mario



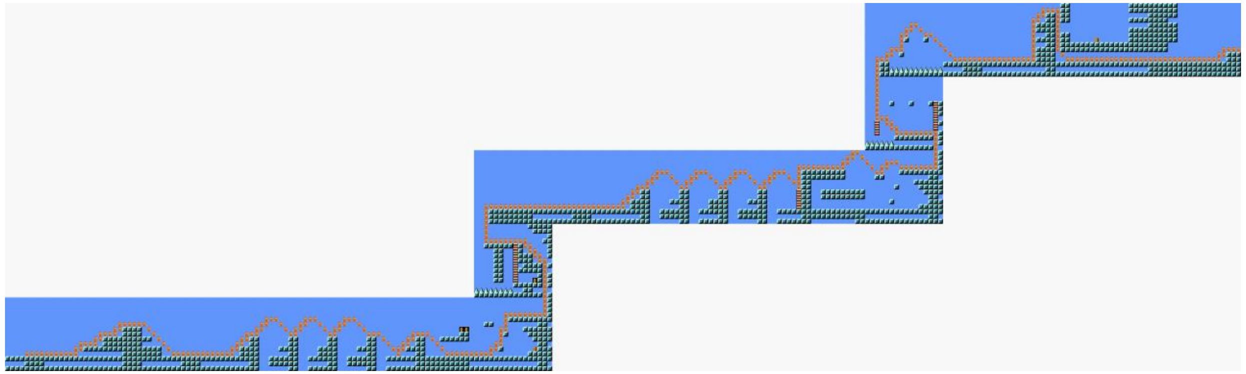
Volz et al., 2017

Motivation

- Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have been used for generating platformer levels and dungeons via sampling, interpolation and evolution

- Work with fixed-size inputs and outputs
  --- necessitates generation by segment rather than by level

- Existing methods combine independently generated segments to form whole levels
  - Dungeons with discrete rooms
  - Okay but not ideal for Mario
  - Multi-directional platformers like Mega Man



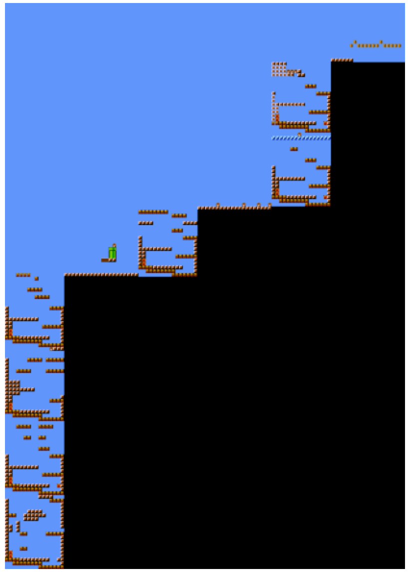
Mega Man level, source: VGLC

Motivation

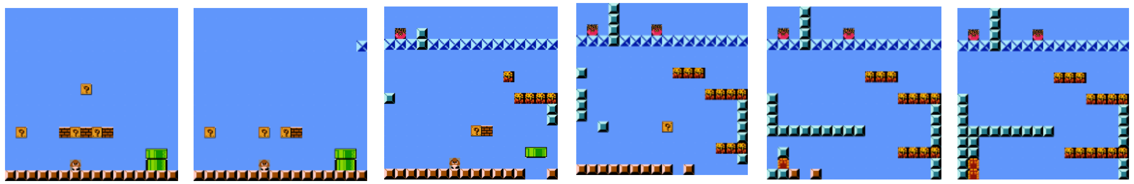
- Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have been used for generating platformer levels and dungeons via sampling, interpolation and evolution

- Work with fixed-size inputs and outputs
  --- necessitates generation by segment rather than by level

- Existing methods combine independently generated segments to form whole levels
  - Dungeons with discrete rooms
  - Okay but not ideal for Mario
  - Multi-directional platformers like Mega Man
  - Game blending restricted to segments and not whole levels

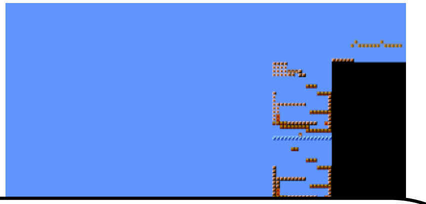


Sarkar and Cooper, 2018



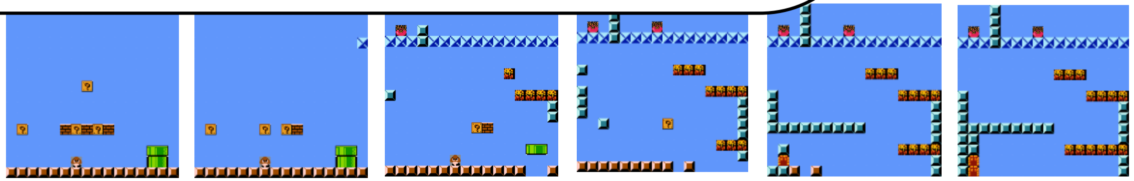
Sarkar, Yang and Cooper, 2019

Motivation



- GANs and VAEs have been used for generating platformer levels and dungeons via sampling, interpola
- Work w
  --- nec
  than b
- Existing
  generat
  - Dunge
  - Okay but not ideal for Mario
  - Multi-directional platformers like Mega Man
  - Game blending restricted to segments and not whole levels

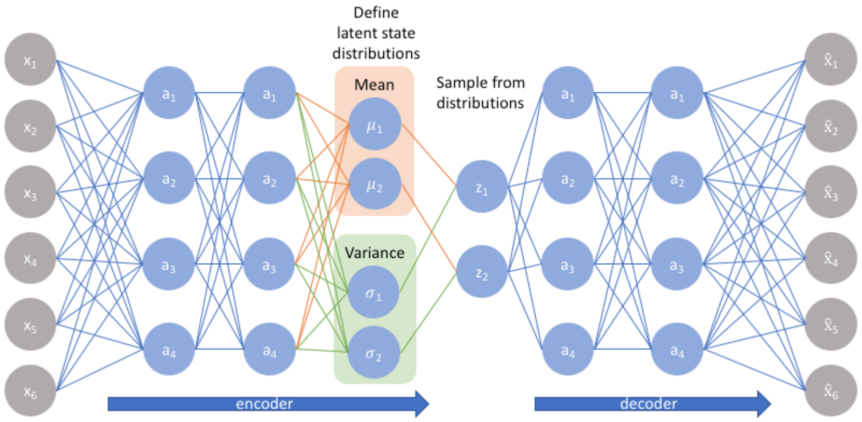
Generate and blend whole platformer levels progressing in multiple directions while still using latent variable models and their fixed-size inputs/outputs



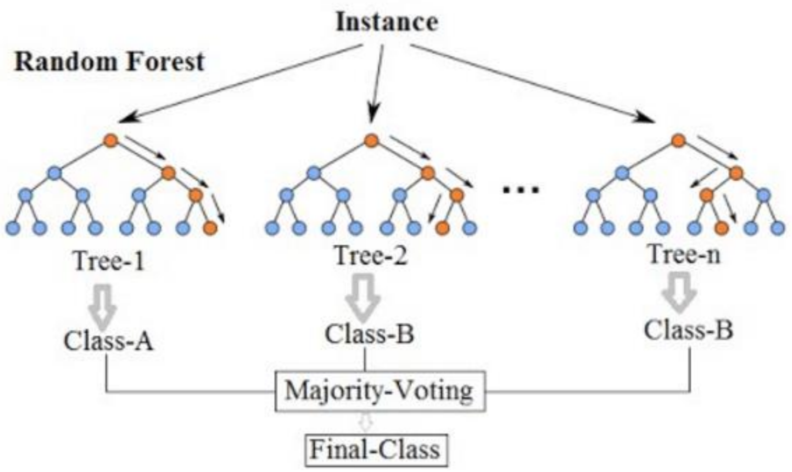
Sarkar, Yang and Cooper, 2019

Solution

- Two-step solution:
  - Modify VAE to learn encoding of next segment rather than current segment
  - Train a classifier to predict where next segment should be placed



VAE (modified), source: jeremyjordan.me



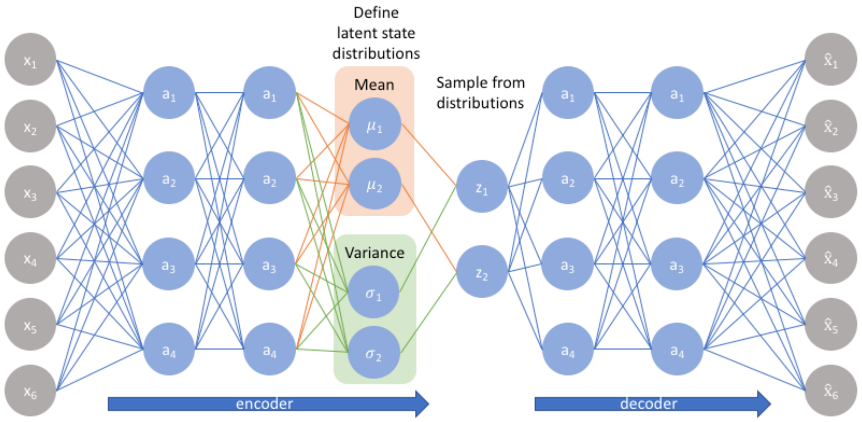
Random forest classifier

Source: https://community.tibco.com/wiki/random-forest-template-tibco-spotfire

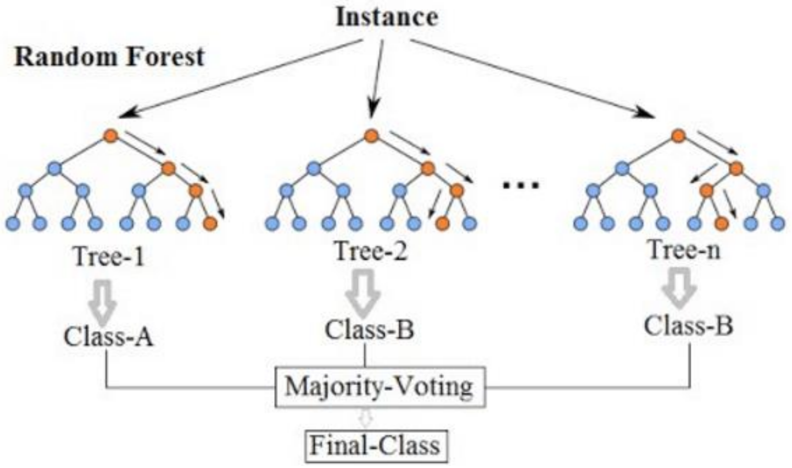
Solution

- Two-step solution:
  - Modify VAE to learn encoding of next segment rather than current segment
  - Train a classifier to predict where next segment should be placed

- Hybrid PCGML model which enables:
  - Generating arbitrarily long levels via iterative encoding-decoding of segments
  - Generating levels that can progress in multiple directions
  - Generating blended levels rather than segments



VAE (modified), source: jeremyjordan.me



Random forest classifier

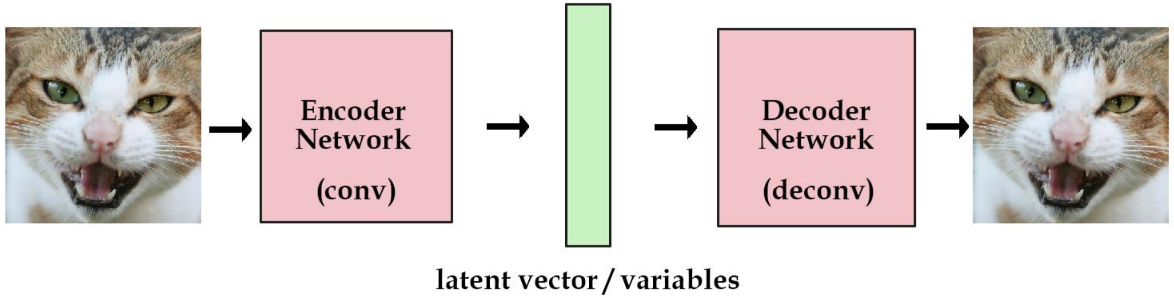
Source: https://community.tibco.com/wiki/random-forest-template-tibco-spotfire

Sequential Segment Generation

- Generative models based on VAEs trained on *Super Mario Bros* (SMB), *Kid Icarus* (KI), *Mega Man* (MM) and blended SMB-KI domain (implementation details in paper)

Sequential Segment Generation

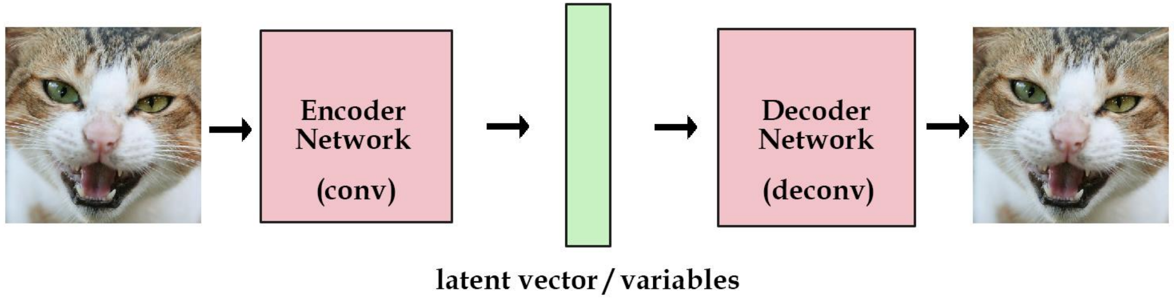
- Generative models based on VAEs trained on *Super Mario Bros* (SMB), *Kid Icarus* (KI), *Mega Man* (MM) and blended SMB-KI domain (implementation details in paper)
- Autoencoders are neural nets that learn lower-dimensional data representations
  - Encoder → input data to latent space
  - Decoder → latent space to reconstructed data



Vanilla Autoencoder

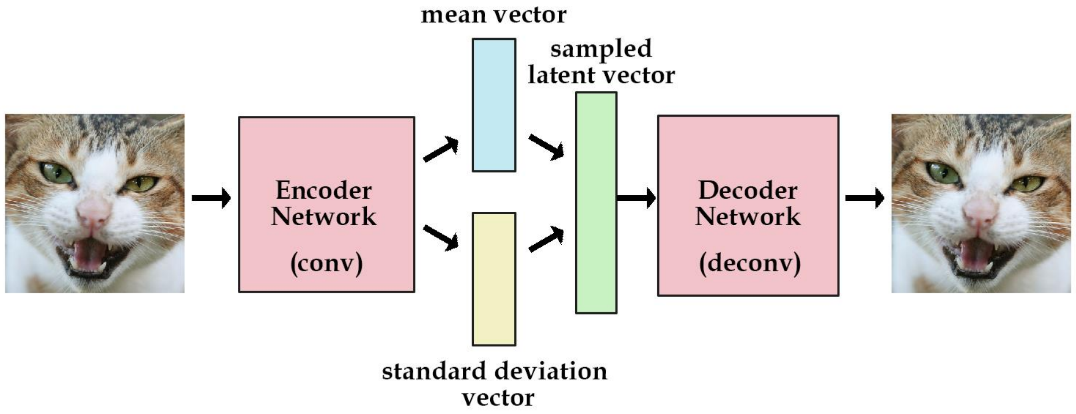
Sequential Segment Generation

- Generative models based on VAEs trained on *Super Mario Bros* (SMB), *Kid Icarus* (KI), *Mega Man* (MM) and blended SMB-KI domain (implementation details in paper)
- Autoencoders are neural nets that learn lower-dimensional data representations
  - Encoder → input data to latent space
  - Decoder → latent space to reconstructed data



Vanilla Autoencoder

- VAEs make latent space model a probability distribution (e.g. Gaussian)
  - Allows learning continuous latent spaces
  - Enables generative abilities similar to those of GANs (sampling, interpolation)



Variational Autoencoder

Source: kvfrans.com

Sequential Segment Generation

- VAE Loss function
  - Reconstruction error

    ---error between input segment and reconstruction of input segment
  - KL Divergence (between latent distribution and known prior)

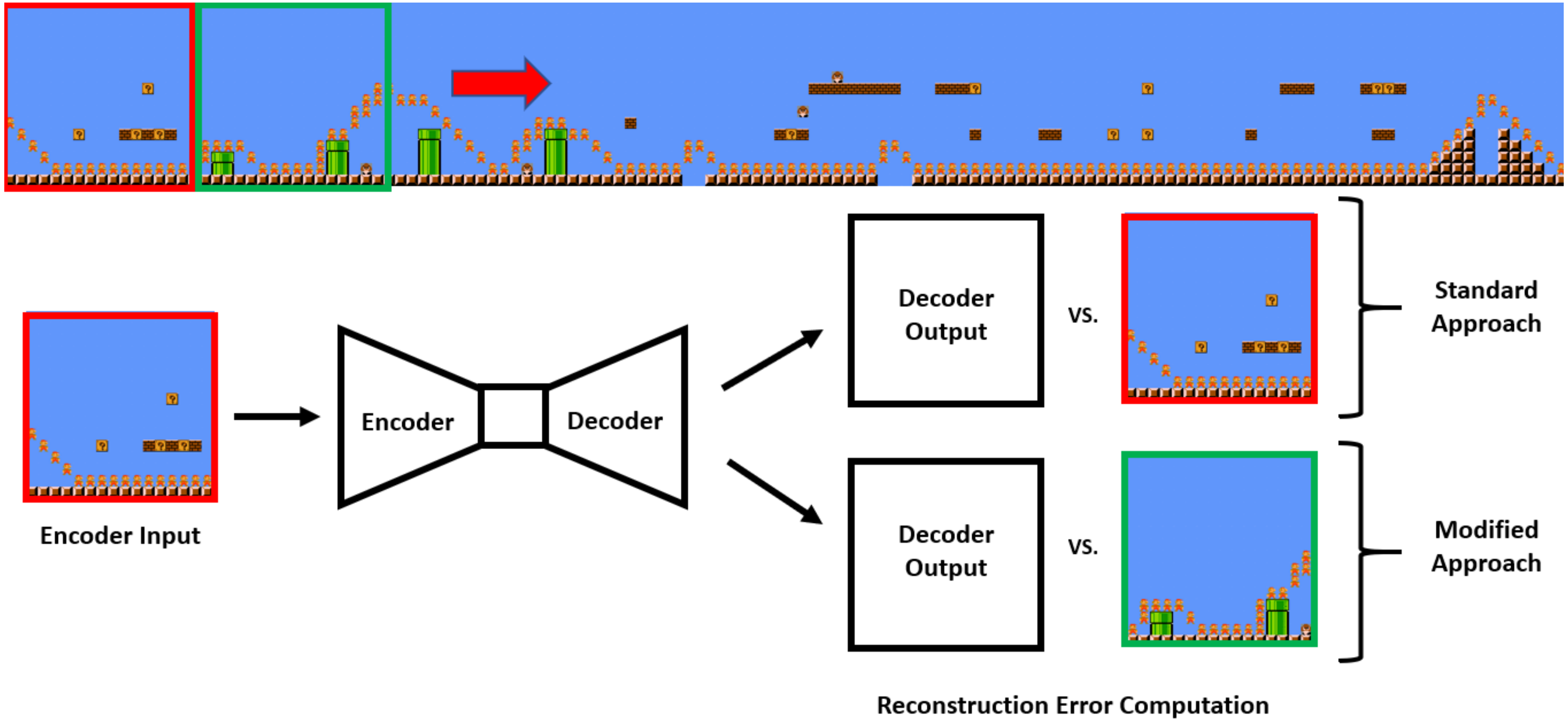
    ---forces latent space to model a continuous, informative distribution

Sequential Segment Generation

- VAE Loss function
  - Reconstruction error (modified)

    *---error between input segment and reconstruction of **next** input segment*

    ---technically, no longer 'auto'-encoding, but enables our approach



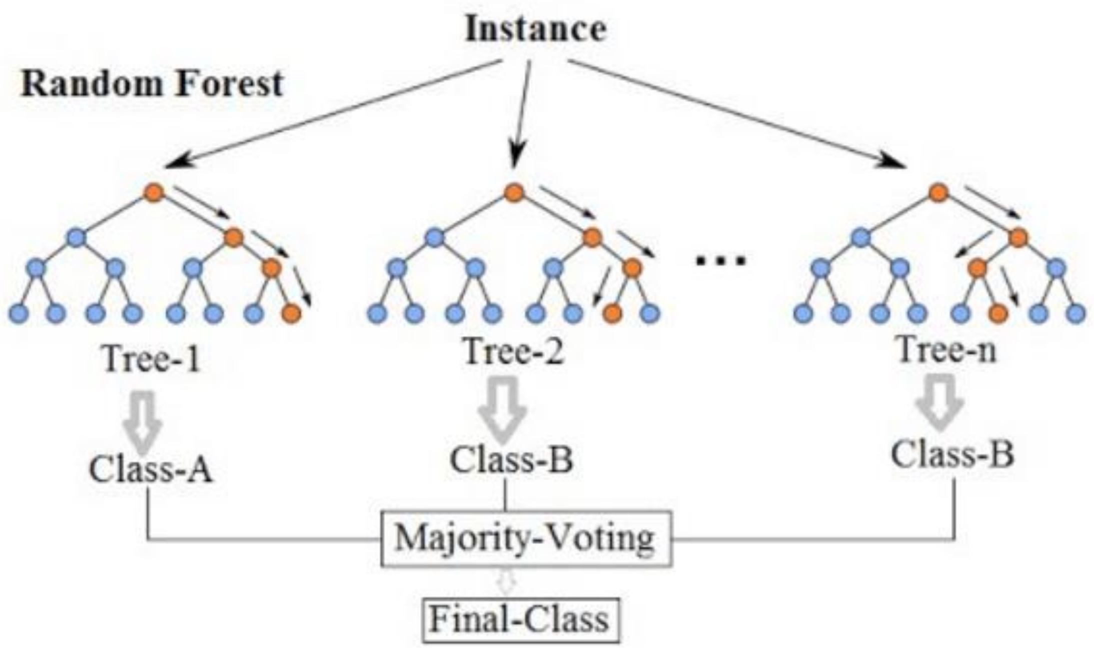
Sequential Segment Generation

Algorithm 1 GenerateLevel(init_segment, n)

```
Initialize level to init_segment
num_segments = 1
segment = init_segment
while num_segments ≤ n do
    z ← Encoder(segment)
    segment ← Decoder(z)
    Add segment to level
    num_segments += 1
end while
return level
```

Placement Classification

- To generate levels that can dynamically progress in any direction, need to determine where/how to place generated segments
- Directional classifier
  - Random forest classifier trained on segments from SMB, KI, MM and SMB-KI domain, labeled with direction of next segment in levels

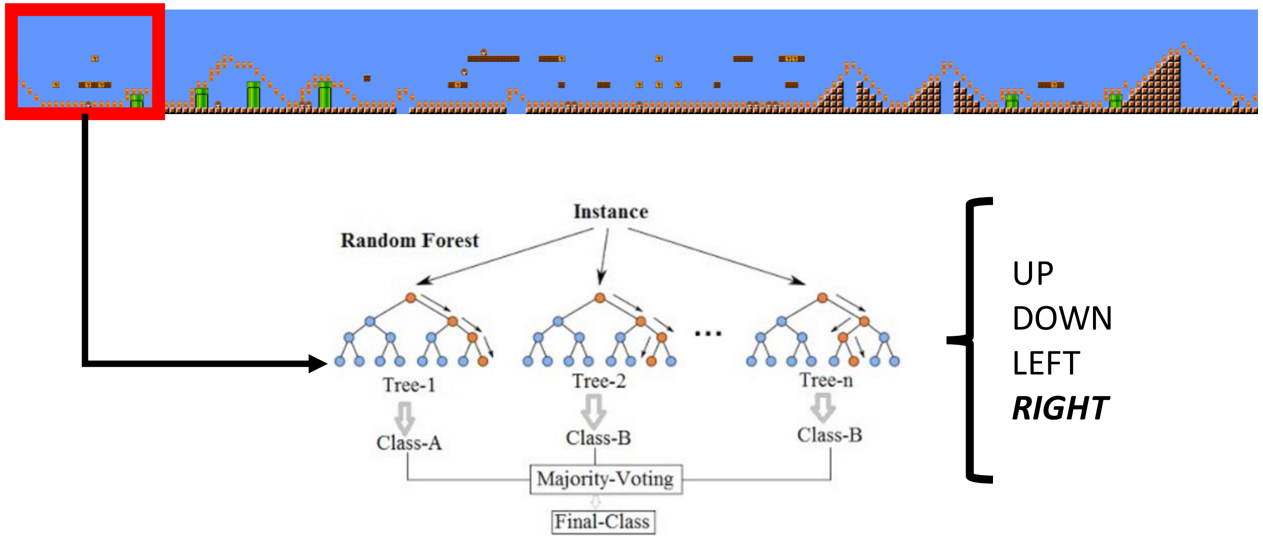


Source: https://community.tibco.com/wiki/random-forest-template-tibco-spotfire

Placement Classification

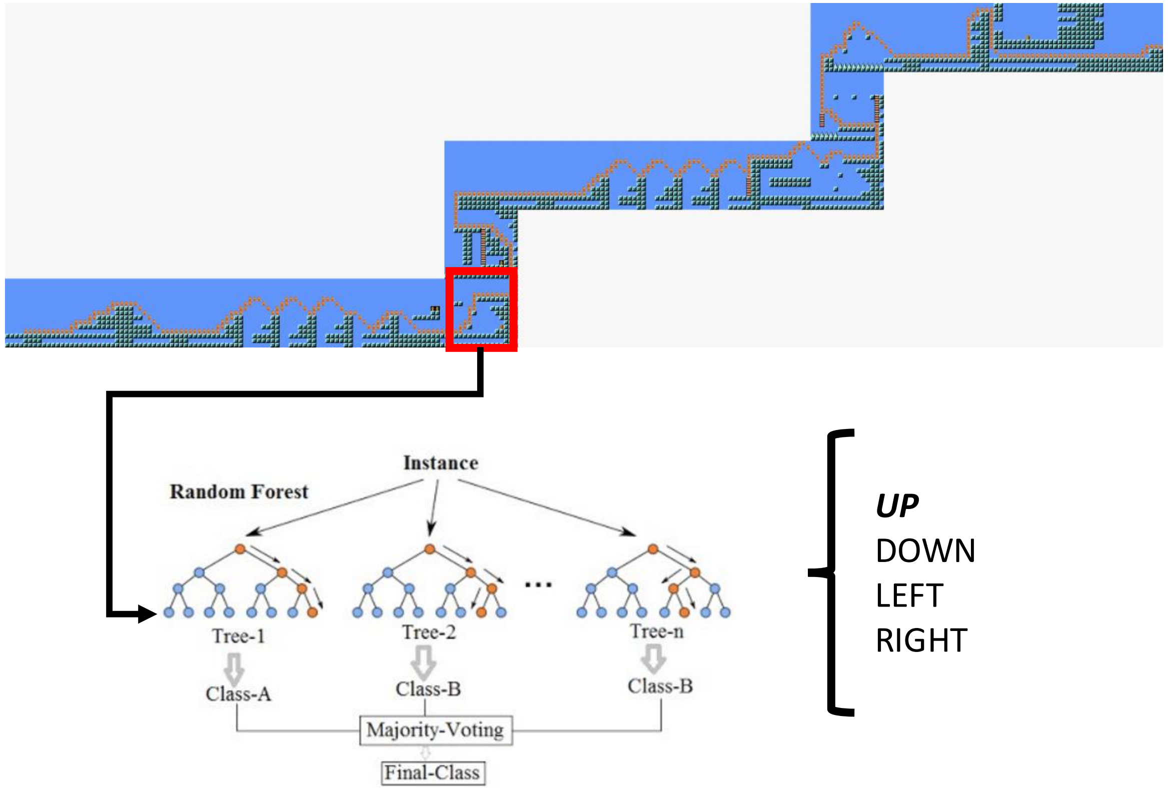
- Input: same segments as before, Label: next direction
  - SMB – right only, KI – up only, MM and SMB+KI – both
  - 70%-30% train-test split
  - 100% accuracy for SMB, KI, SMB-KI, 98.73% for MM
  - Post-processing after prediction (details in the paper)

Super Mario Bros.



UP
DOWN
LEFT
RIGHT

Mega Man

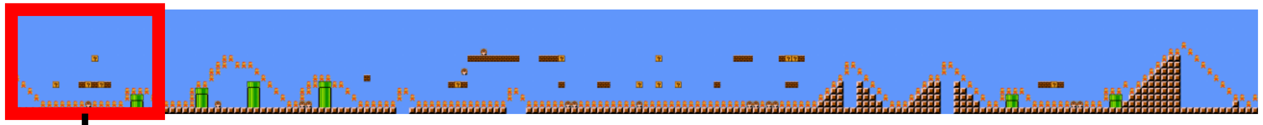


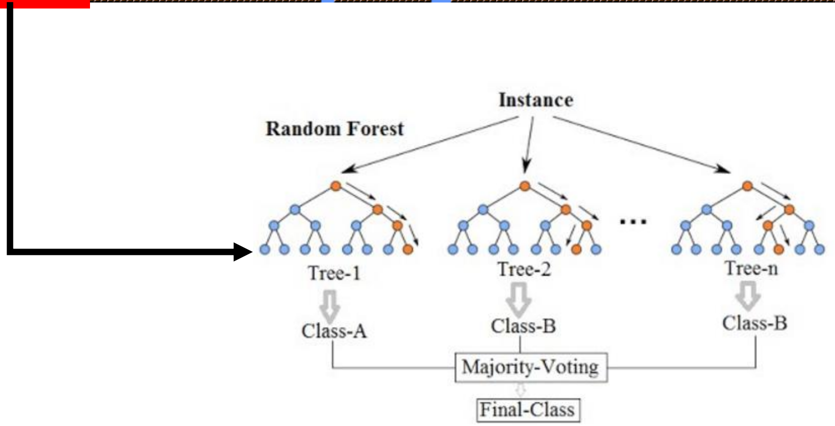
UP
DOWN
LEFT
RIGHT

Placement Classification

- Input: same segments as before, Label: next direction
  - SMB – right only, KI – up only, MM and SMB+KI – both
  - 70%-30% train-test split
  - 100% accuracy for SMB, KI, SMB-KI, 98.73% for MM
  - Post-processing after prediction (details in the paper)

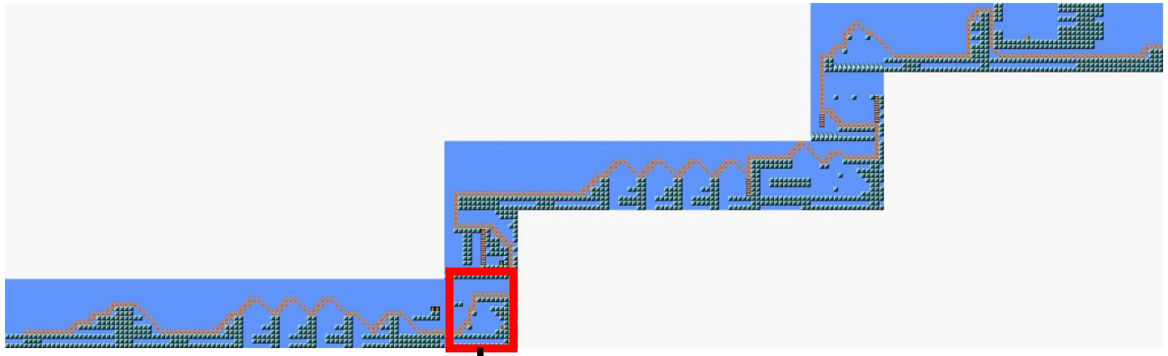
Super Mario Bros.

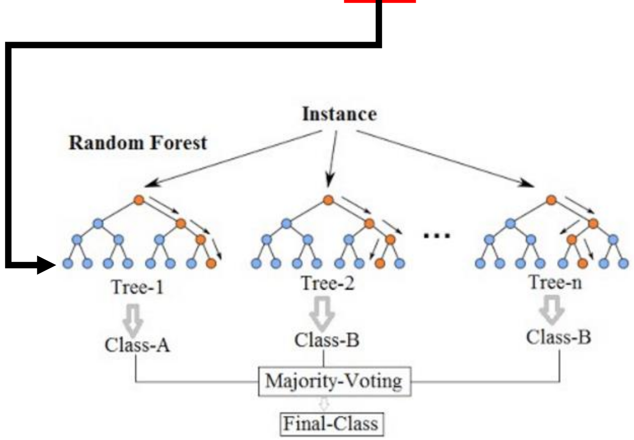




UP
DOWN
LEFT
RIGHT

Mega Man





UP
DOWN
LEFT
RIGHT

Placement Classification

Algorithm 2 GenerateLevelWithDirs(init_segment, n)

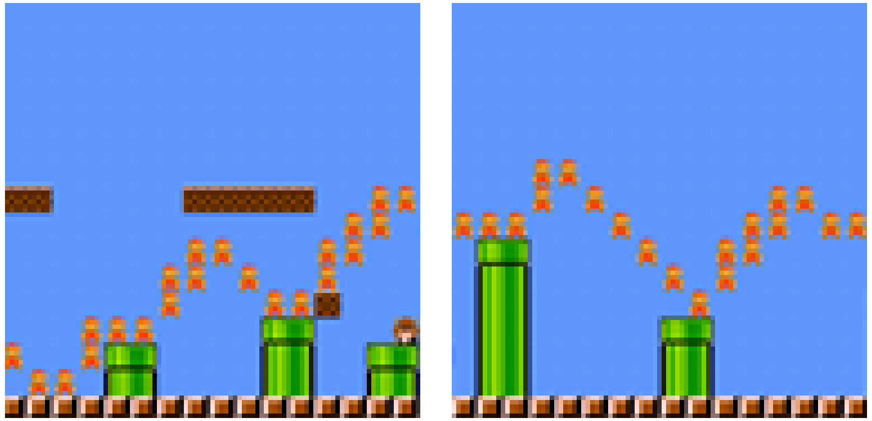
$level \leftarrow$ GenerateLevel(init_segment, n)
$level\_with\_dirs \leftarrow \emptyset$
for $segment$ in $level$ **do**
  $dir \leftarrow Classifier(segment)$
  Add $(segment, dir)$ to $level\_with\_dirs$
end for
return $level\_with\_dirs$

Evaluation

- Three-part evaluation
  - Continuous nature of generated levels
  - Properties of generated blended levels
  - Quality of arbitrarily long generated levels

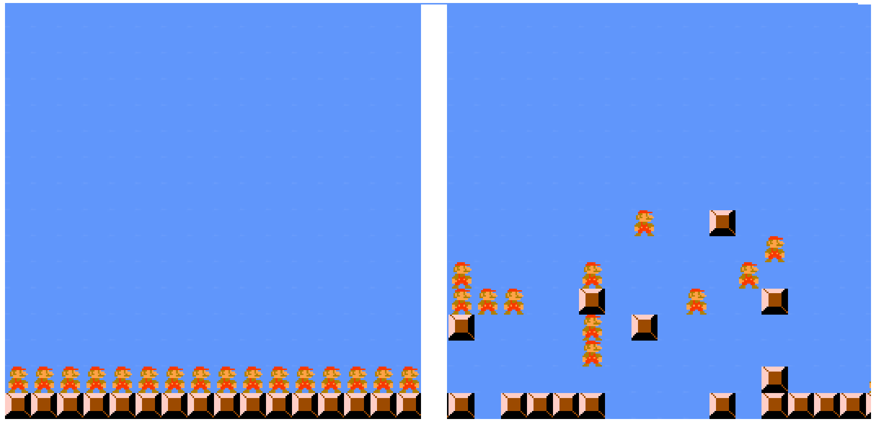
Discontinuity

- To test continuous flow of progression, introduced *Discontinuity* metric
  - Absolute distance between path tiles along the adjoining edge of two successive segments
  - Lower values → higher continuity between successive segments
  - Range from 0 (high continuity) to 16 (low continuity)



E.g. Discontinuity = 1

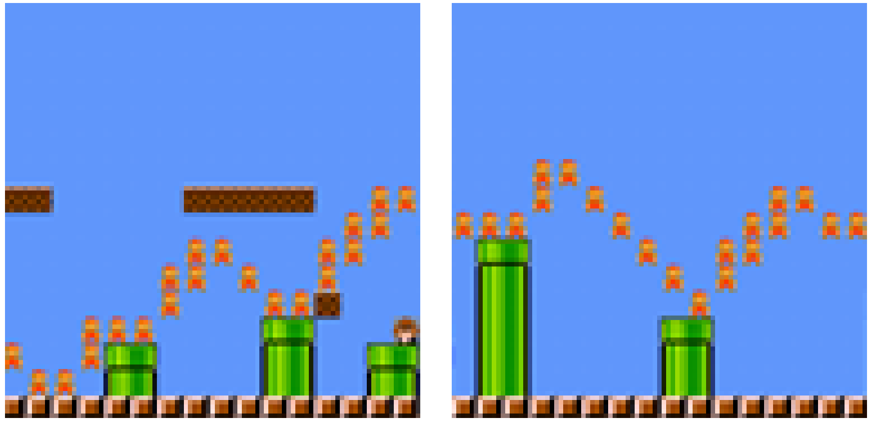
- Levels with better sense of progression would have a more continuous path through its segments i.e. low values of *Discontinuity* between successive pairs of segments



E.g. Discontinuity = 3

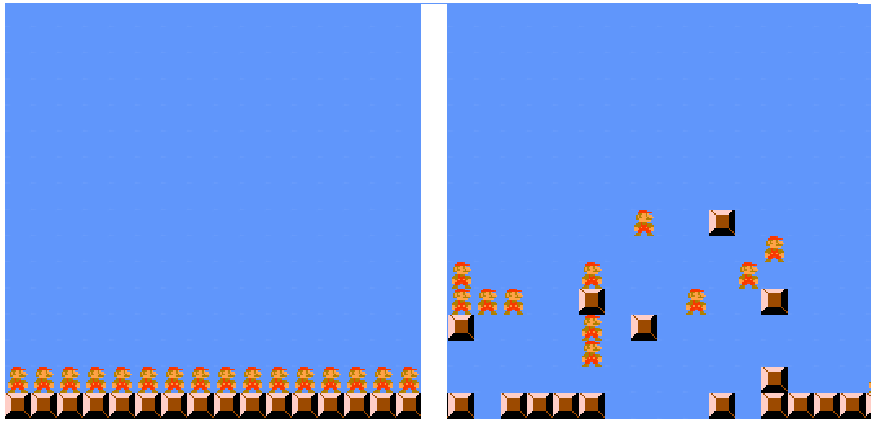
Discontinuity

- To test continuous flow of progression, introduced *Discontinuity* metric
  - Absolute distance between path tiles along the adjoining edge of two successive segments
  - Lower values → higher continuity between successive segments
  - Range from 0 (high continuity) to 16 (low continuity)



E.g. Discontinuity = 1

- Levels with better sense of progression would have a more continuous path through its segments i.e. low values of *Discontinuity* between successive pairs of segments



E.g. Discontinuity = 3

Discontinuity

- Computed average per-segment Discontinuity for 100 generated levels each for SMB, KI, MM and SMB-KI using 2 methods for generating segments:
  - *Sequential:* using our algorithm
  - *Independent:* successive segments independent of each other
  - For both, generated segments combined using classifier

- Each generated level consisted of 12 segments for SMB, KI and SMB-KI and 16 segments for MM

- Significantly lower discontinuity values using *Sequential* for all games

Game	Sequential	Independent
SMB	3.86 ± 2.28	5.91 ± 2.04
KI	3.99 ± 2.59	7.37 ± 1.99
MM	6.54 ± 2.63	11.18 ± 1.69
SMB-KI	5.4 ± 2.42	9.84 ± 1.76

Table 1: Average per-segment *Discontinuity* values along with standard deviation. A Wilcoxon Rank Sum Test showed differences to be significant with $p < .001$ in all cases.

Discontinuity

- Computed average per-segment Discontinuity for 100 generated levels each for SMB, KI, MM and SMB-KI using 2 methods for generating segments:
  - *Sequential:* using our algorithm
  - *Independent:* successive segments independent of each other
  - For both, generated segments combined using classifier

- Each generated level consisted of 12 segments for SMB, KI and SMB-KI and 16 segments for MM

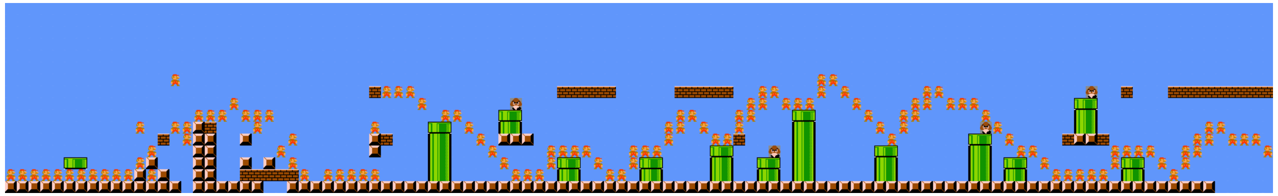
- Significantly lower discontinuity values using *Sequential* for all games

Game	Sequential	Independent
SMB	3.86 ± 2.28	5.91 ± 2.04
KI	3.99 ± 2.59	7.37 ± 1.99
MM	6.54 ± 2.63	11.18 ± 1.69
SMB-KI	5.4 ± 2.42	9.84 ± 1.76

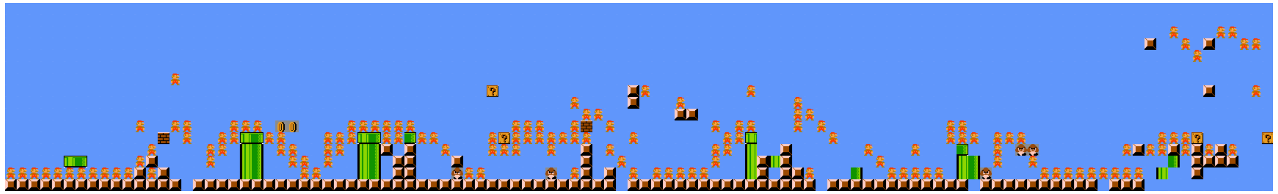
Table 1: Average per-segment *Discontinuity* values along with standard deviation. A Wilcoxon Rank Sum Test showed differences to be significant with $p < .001$ in all cases.

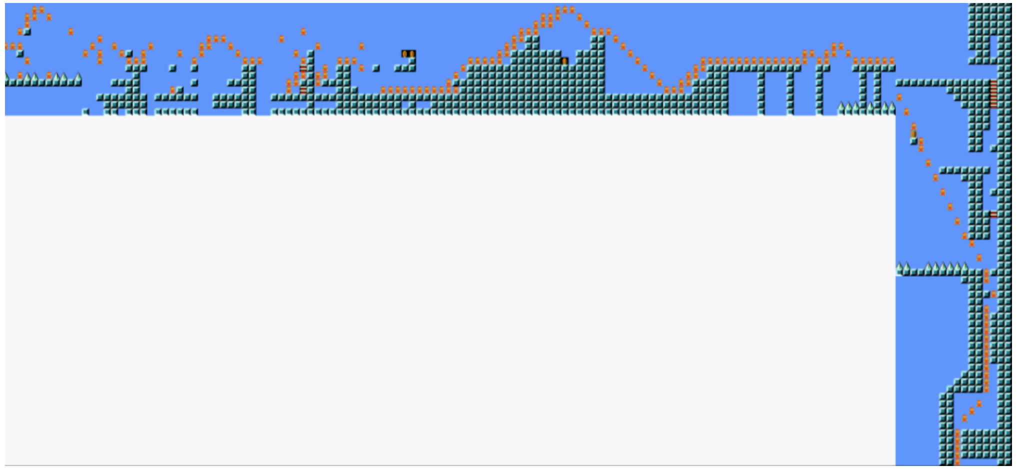
Example Levels

SMB-Sequential

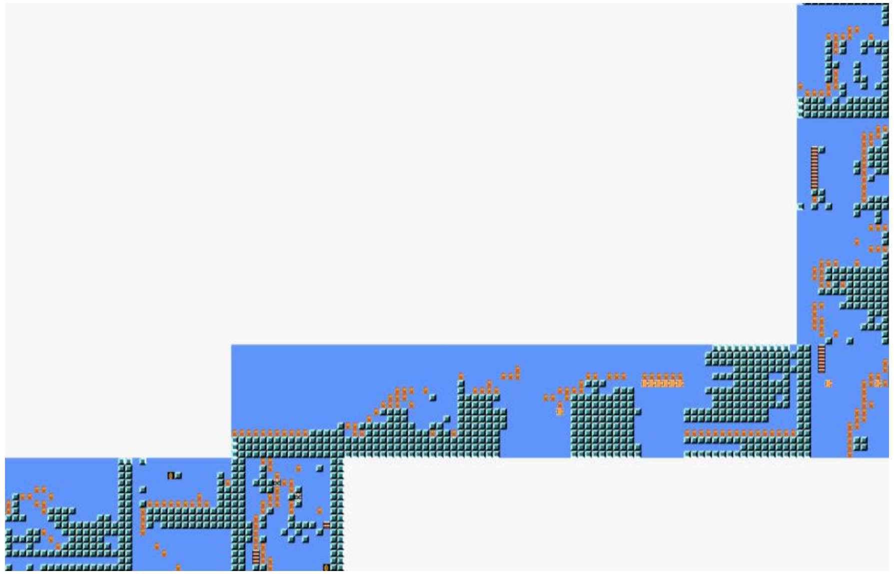


SMB-Independent





MM-Sequential



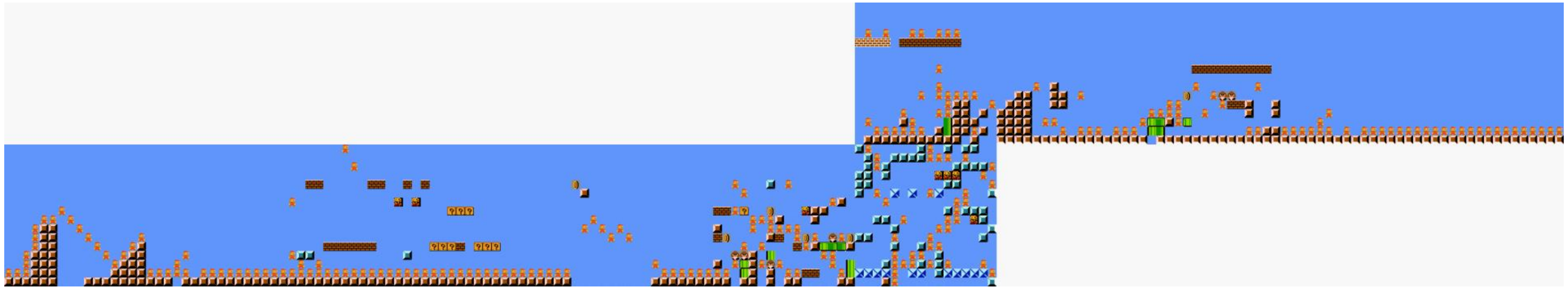
MM-Independent

KI-Sequential

KI-Independent

Example Blended Levels

Blended SMB-KI-Sequential



Blended SMB-KI-Independent

Blending

- Generated blended SMB-KI levels of 12 segments each

- 6 sets of 100 each with a different starting segment
  - Random sample from SMB-KI latent space
  - Original SMB segment
  - Original KI segment
  - 3 segments interpolated between above 2
    - SMB-25%,KI-75%
    - Both-50%
    - SMB-75%,KI-25%

Blend	SMB	KI
SMB-0	0.5	99.5
SMB-25	4	96
SMB-50	86.1	13.9
SMB-75	85	15
SMB-100	94.3	5.7
Random Blend	43.4	56.6

Table 2: Percentage of segments (out of $100x12 = 1200$**) classified as SMB-like and KI-like using the directional classifier.**

- Evaluated using directional classifier
  - Prediction: Right → Segment is more SMB-like
  - Prediction: Up → Segment is more KI-like

Blending

- Generated blended SMB-KI levels of 12 segments each

- 6 sets of 100 each with a different starting segment
  - Random sample from SMB-KI latent space
  - Original SMB segment
  - Original KI segment
  - 3 segments interpolated between above 2
    - SMB-25%,KI-75%
    - Both-50%
    - SMB-75%,KI-25%

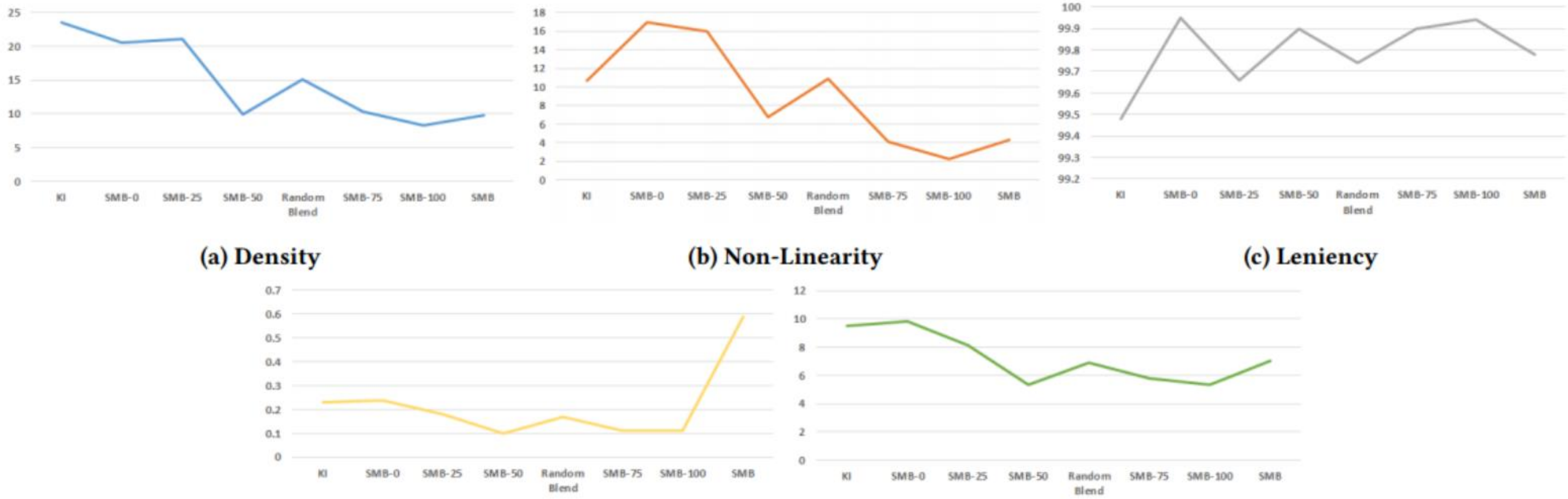
Blend	SMB	KI
SMB-0	0.5	99.5
SMB-25	4	96
SMB-50	86.1	13.9
SMB-75	85	15
SMB-100	94.3	5.7
Random Blend	43.4	56.6

Table 2: Percentage of segments (out of $100x12 = 1200$) classified as SMB-like and KI-like using the directional classifier.

- Evaluated using directional classifier
  - Prediction: Right → Segment is more SMB-like
  - Prediction: Up → Segment is more KI-like

Blending

- Compared generated blended levels with original SMB and KI levels using tile metrics
  - *Density (proportion of solid tiles)*
  - *Non-Linearity (unevenness of segment topology)*
  - *Leniency (proxy for difficulty)*
  - *Interestingness (proportion of decorative/collectible items)*
  - *Path-Prop (proportion of path tiles)*



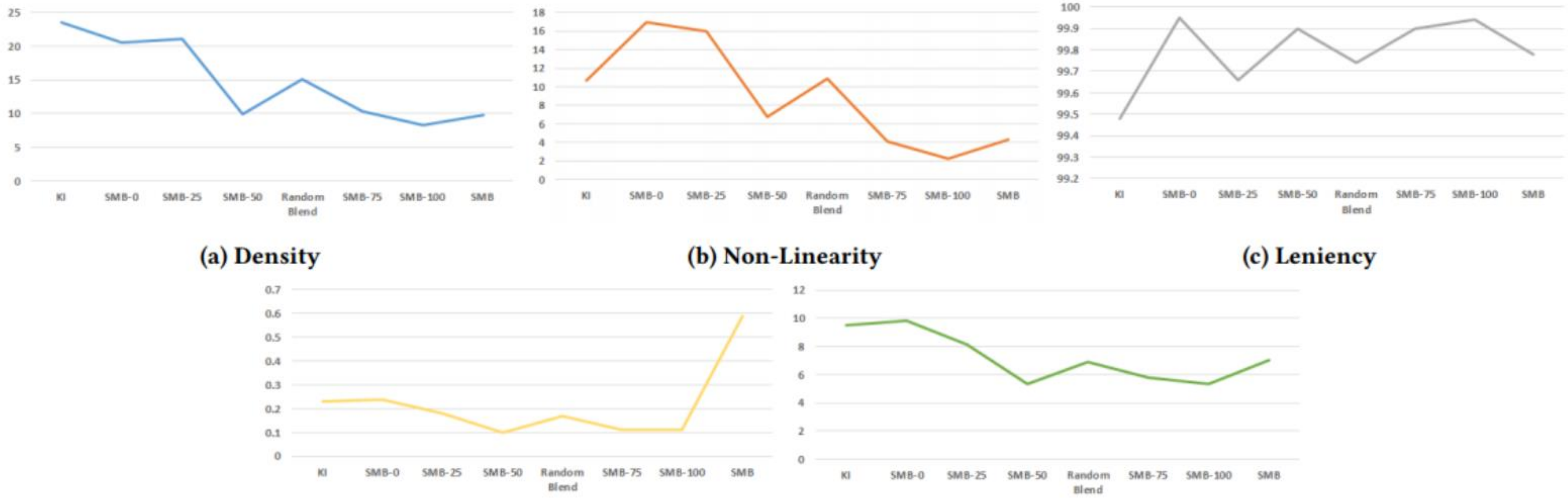
(a) Density **(b) Non-Linearity** **(c) Leniency**

(d) Interestingness **(e) Path-Prop**

Figure 2: Per-segment tile metrics for original SMB and KI levels along with different types of blends.

Blending

- Compared generated blended levels with original SMB and KI levels using tile metrics
  - *Density (proportion of solid tiles)*
  - *Non-Linearity (unevenness of segment topology)*
  - *Leniency (proxy for difficulty)*
  - *Interestingness (proportion of decorative/collectible items)*
  - *Path-Prop (proportion of path tiles)*



(a) Density (b) Non-Linearity (c) Leniency

(d) Interestingness (e) Path-Prop

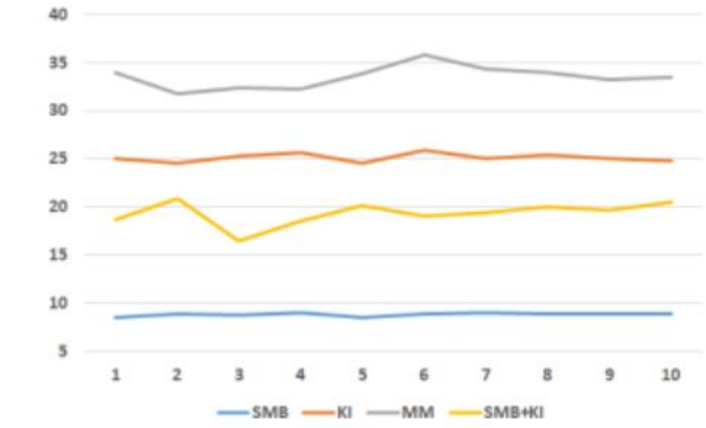
Figure 2: Per-segment tile metrics for original SMB and KI levels along with different types of blends.

Progression

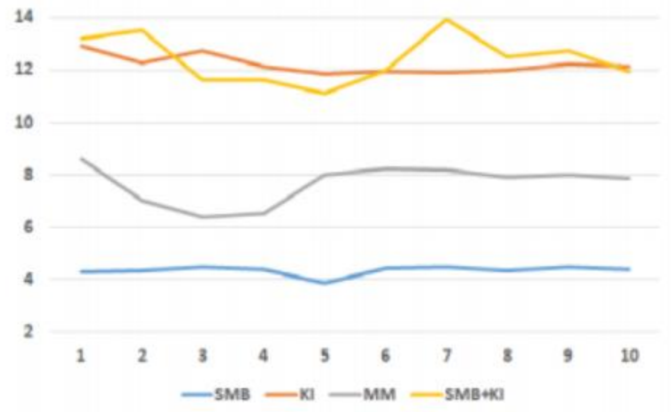
- Generate arbitrarily long levels without deteriorating quality

- Generated 100 levels of 120 segments each for SMB, KI and SMB-KI and 160 segments each for MM (approx. 10x size of average actual levels)
  - Computed average per-segment Discontinuity and tile-based metrics for each of the 10 subsections of each level
  - That is, track if/how these values change as more segments are generated conditioned on the previous ones

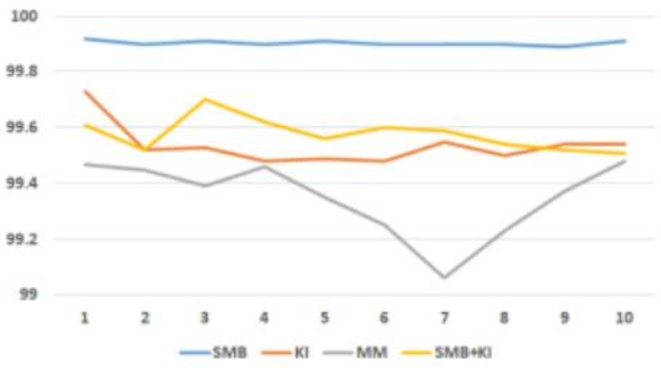
Progression



(a) Density

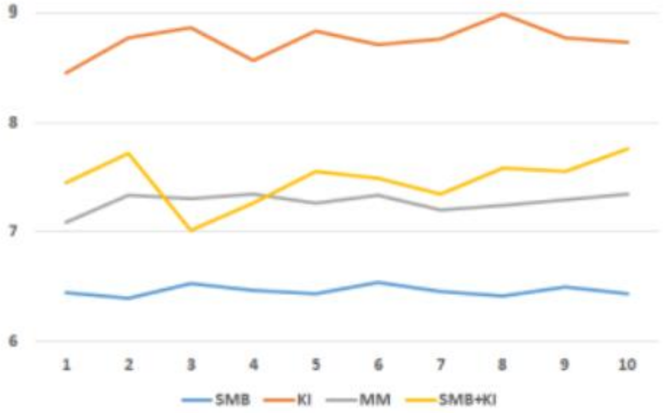


(b) Non-Linearity

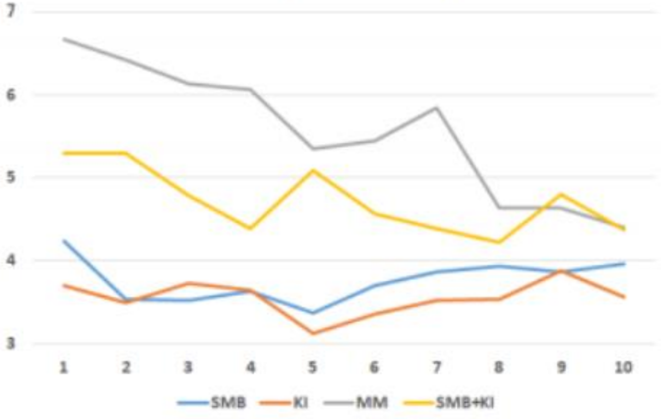


(c) Leniency

(d) Interestingness



(e) Path-Prop



(f) Discontinuity

Figure 3: Per-segment metric values plotted for each grouping of 16 segments for MM and each grouping of 12 segments for the other games. x-axis values indicate 1st such grouping, 2nd such grouping etc. y-axis indicates average metric value for the corresponding group of segments.

Progression

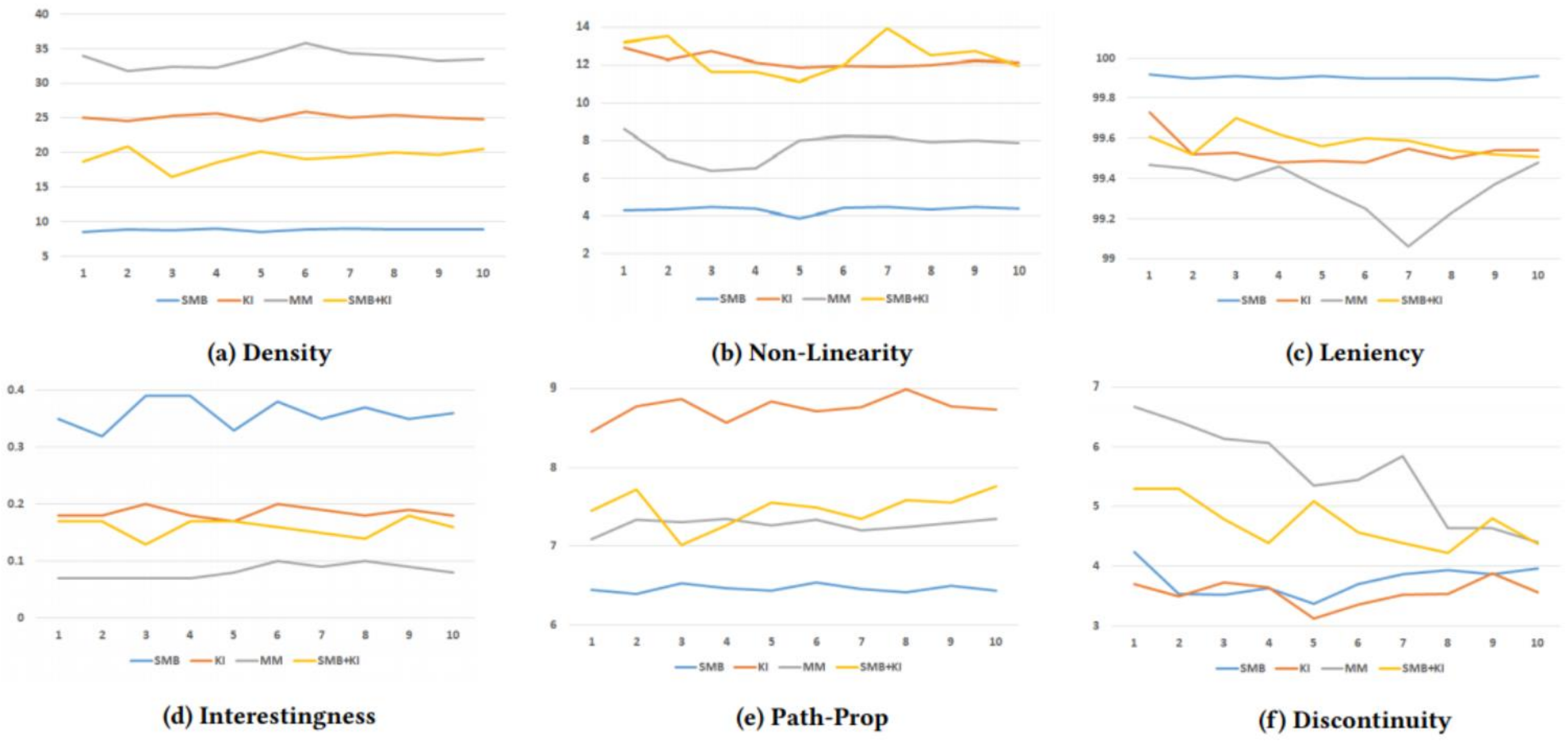


Figure 3: Per-segment metric values plotted for each grouping of 16 segments for MM and each grouping of 12 segments for the other games. x-axis values indicate 1st such grouping, 2nd such grouping etc. y-axis indicates average metric value for the corresponding group of segments.

Conclusion

- Novel PCGML approach for sequential platformer level generation and blending
- Generate arbitrarily-long coherent platformer levels
- Generate platformer levels progressing in multiple directions
- Blend levels from platformers progressing in different directions

Future Work

- Investigate other placement strategies (e.g. heuristics vs. classifier)
- Improve generation quality (particularly for Mega Man)
- Empirically test generation of left-to-right progressing levels (such as in Ninja Gaiden)
- Add controllability of generation beyond choice of initial segment

Future Work

- Investigate other placement strategies (e.g. heuristics vs. classifier)
- Improve generation quality (particularly for Mega Man)
- Empirically test generation of left-to-right progressing levels (such as in Ninja Gaiden)
- Add controllability of generation beyond choice of initial segment

Contact

Anurag Sarkar
Northeastern University
sarkar.an@northeastern.edu