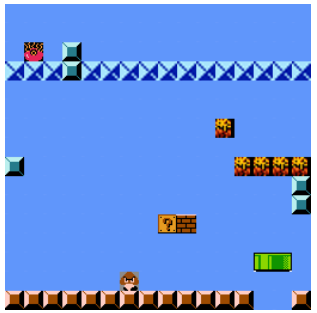
Dungeon and Platformer Level Blending and Generation using Conditional VAEs

Anurag Sarkar and **Seth Cooper**

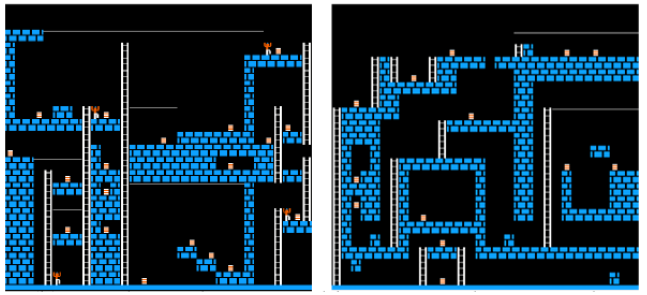
Northeastern University

Motivation

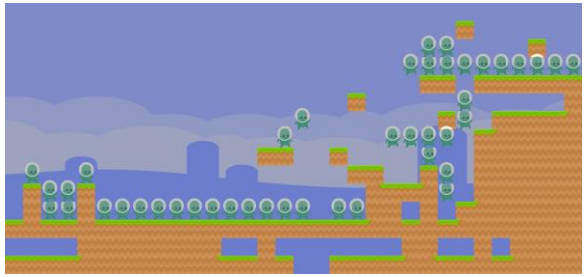
- Recent works have used variational autoencoders (VAEs) for generating and blending levels for several games



Sarkar, Yang and Cooper, 2019



Thakkar et al., 2019

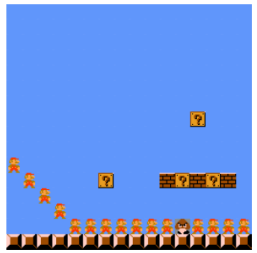


Sarkar et al., 2020

Motivation

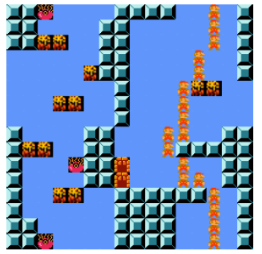
- Recent works have used variational autoencoders (VAEs) for generating and blending levels for several games

- Additional controllability via conditional VAEs (CVAEs)
  - Train on labeled data
  - Generation conditioned on input labels



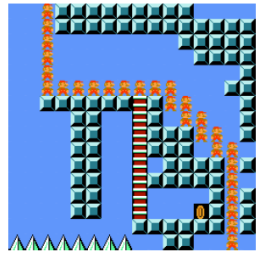
SMB - ⟨10011⟩

< Enemy, Pipe, Coin, Breakable, ?-Mark >



KI - ⟨1101⟩

< Hazard, Door, Moving, Stationary >



MM - ⟨10101⟩

< Hazard, Door, Ladder, Platform, Collectible >

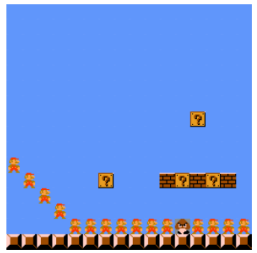
Sarkar, Yang and Cooper, 2020

Motivation

- Recent works have used variational autoencoders (VAEs) for generating and blending levels for several games

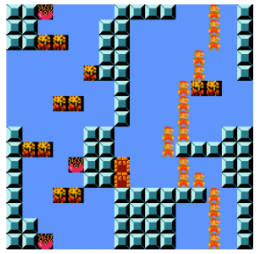
- Additional controllability via conditional VAEs (CVAEs)
  - Train on labeled data
  - Generation conditioned on input labels

- Drawbacks
  - Fixed size inputs/outputs necessitates working with segments
  - Prior blending work limited to platformers



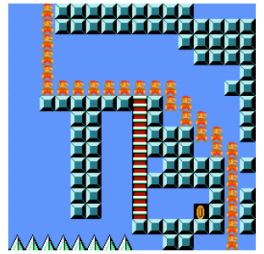
SMB - $\langle 10011 \rangle$

< Enemy, Pipe, Coin, Breakable, ?-Mark >



KI - $\langle 1101 \rangle$

< Hazard, Door, Moving, Stationary >



MM - $\langle 10101 \rangle$

< Hazard, Door, Ladder, Platform, Collectible >

Sarkar, Yang and Cooper, 2020

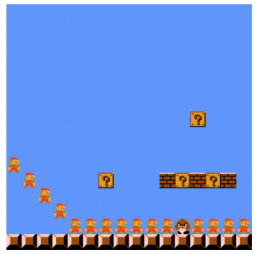
Motivation

- Recent works have used variational autoencoders (VAEs) for generating and blending levels for several games

- Additional controllability via conditional VAEs (CVAEs)
  - Train on labeled data
  - Generation conditioned on input labels

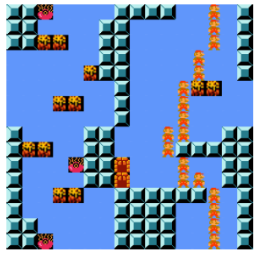
- Drawbacks
  - Fixed size inputs/outputs necessitates working with segments
  - Prior blending work limited to platformers

- IDEA: use conditioning labels of CVAE to control progression rather than content
  - Generate whole platformer and dungeon levels
  - Blend between platformers and dungeons



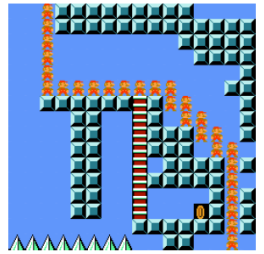
SMB - $\langle 10011 \rangle$

< Enemy, Pipe, Coin, Breakable, ?-Mark >



KI - $\langle 1101 \rangle$

< Hazard, Door, Moving, Stationary >



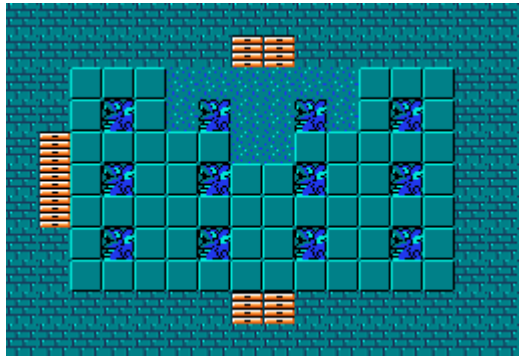
MM - $\langle 10101 \rangle$

< Hazard, Door, Ladder, Platform, Collectible >

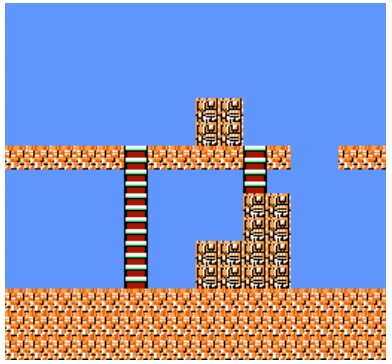
Sarkar, Yang and Cooper, 2020

Approach

- Trained CVAEs on VGLC levels of Zelda, Mega Man, Metroid, Lode Runner and various blends
  --- four versions of each model with latent dimensions of 4, 8, 16 and 32

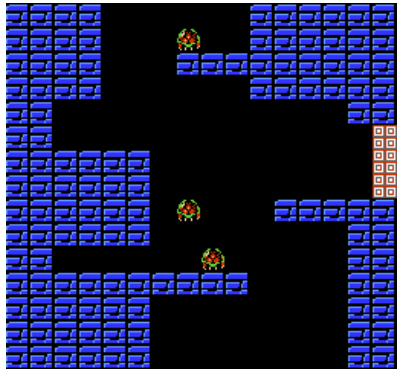


Zelda

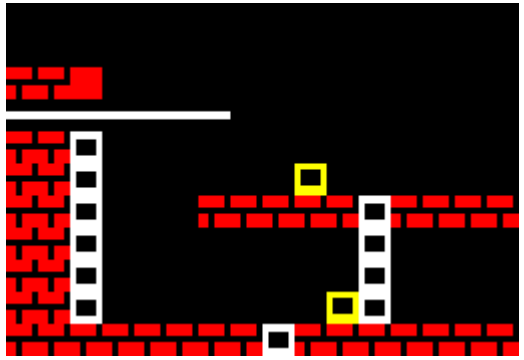


Mega Man (MM)

- Labels indicating progression
  - Platformer segments: direction(s) that are open
  - Dungeon rooms: direction(s) with doors
  - Enable generating rooms and segments with orientations so that they can be connected to form whole levels



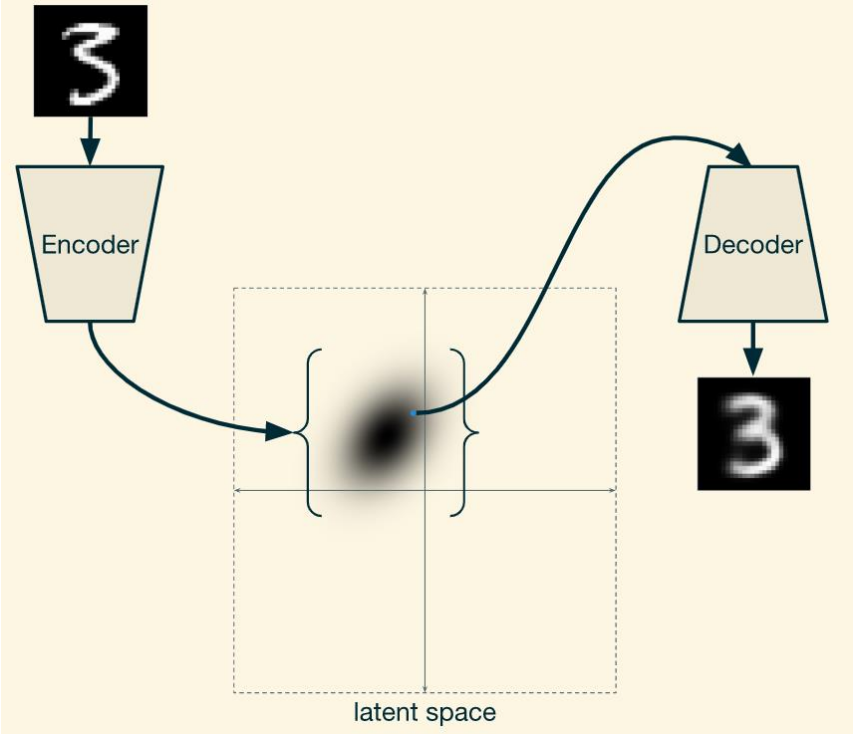
Metroid (Met)

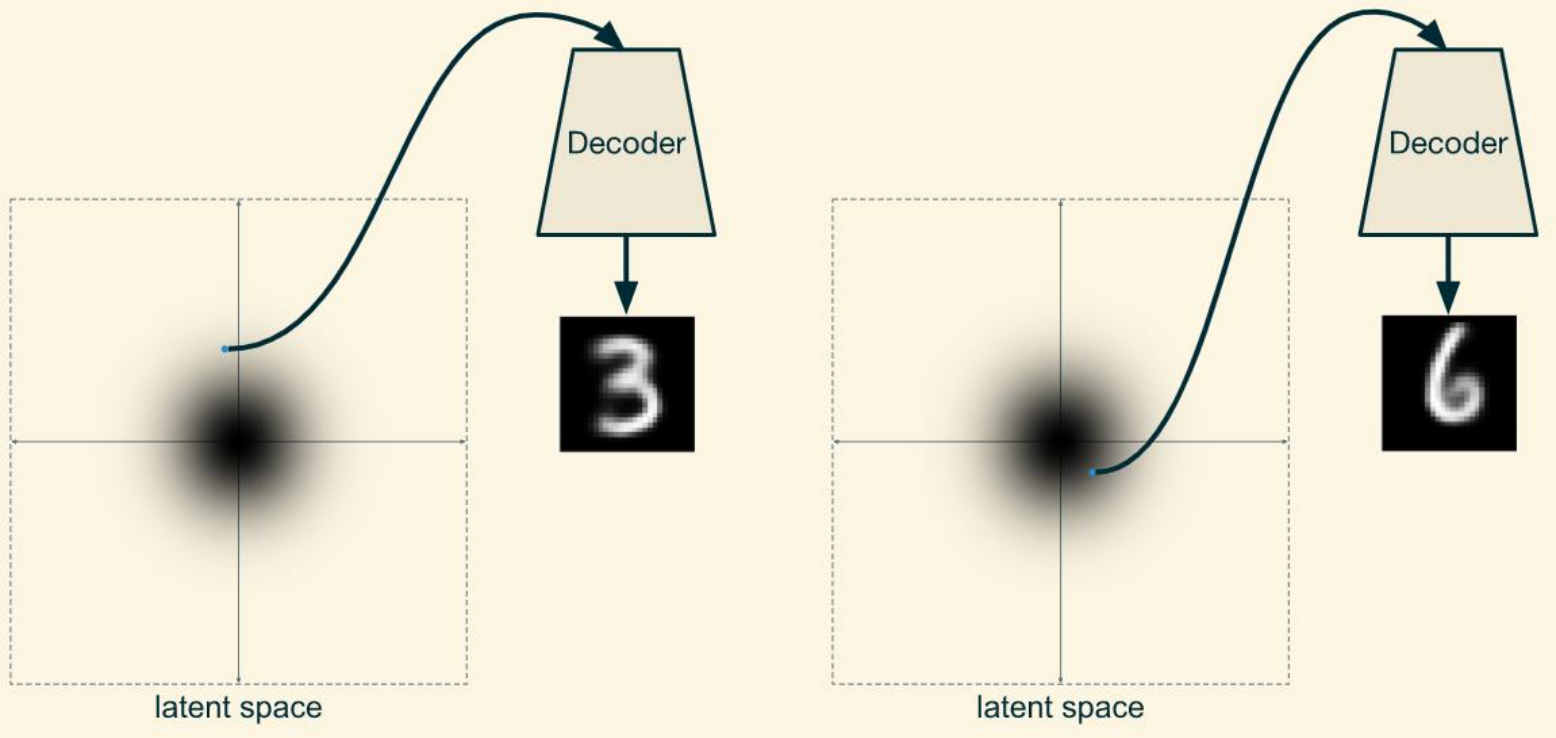


Lode Runner (LR)

Variational Autoencoder (VAE)

- Autoencoders are neural nets that learn lower-dimensional data representations
  - Encoder → input data to latent space
  - Decoder → latent space to reconstructed data
- VAEs make latent space model a probability distribution (e.g. Gaussian)
  - Allows learning continuous latent spaces
  - Enables generative abilities similar to those of GANs (via sampling and interpolation)

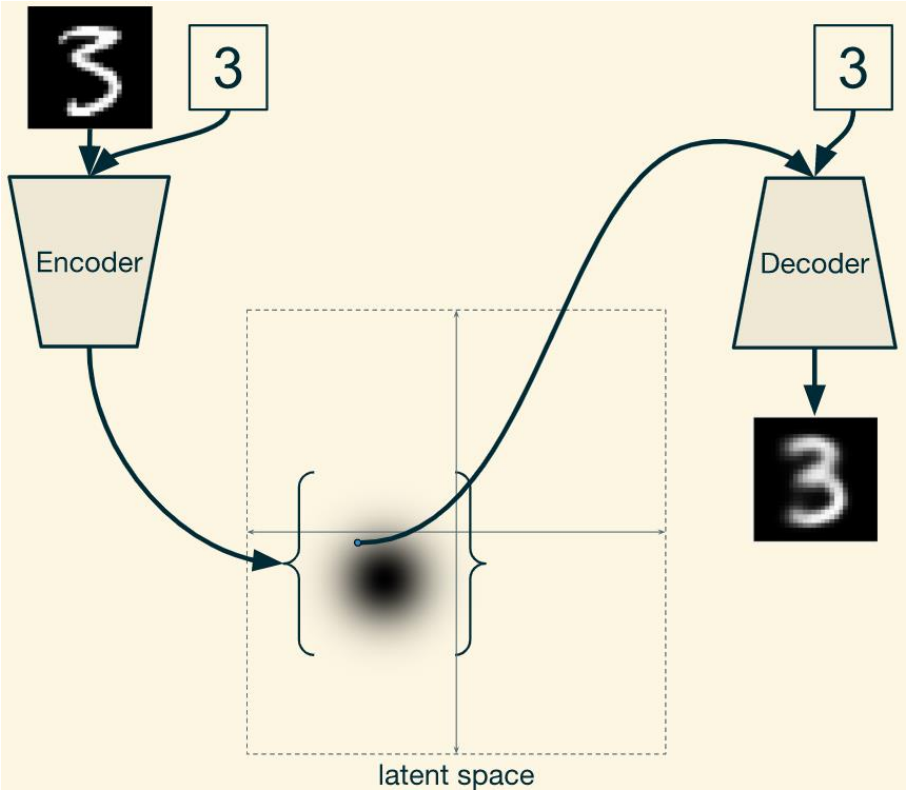


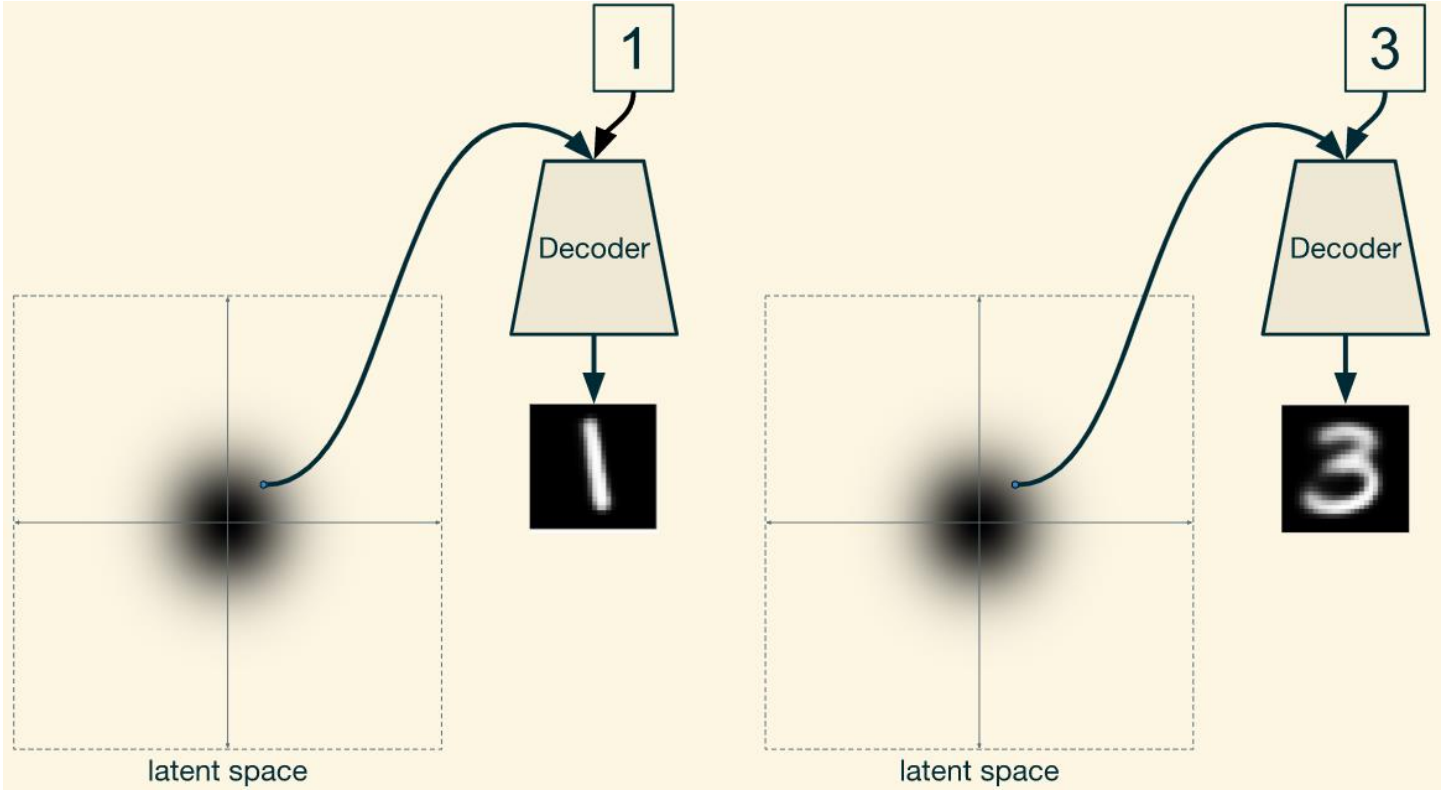


source: jdykeman.github.io/ml/2016/12/21/cvae.html

Conditional VAE (CVAE)

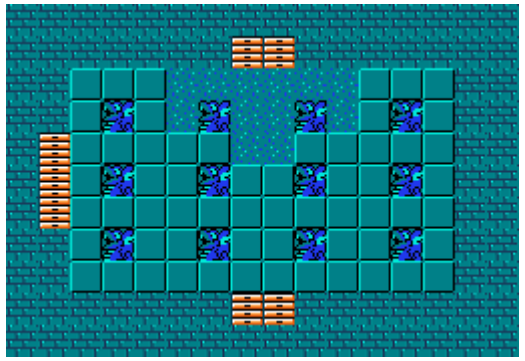
- CVAEs associate input data with labels during training
- Encoder uses label to learn latent encodings of inputs
- Decoder uses same label to learn how to reconstruct input from latent encoding
- Same latent vector can produce different outputs by varying label



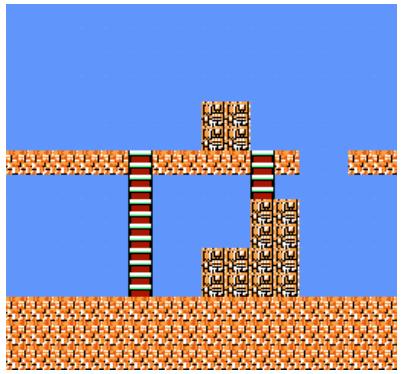


source: jdykeman.github.io/ml/2016/12/21/cvae.html

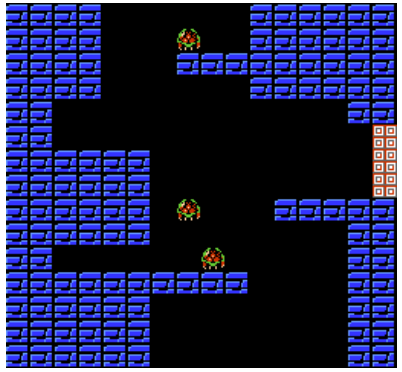
Level Representation and Conditioning



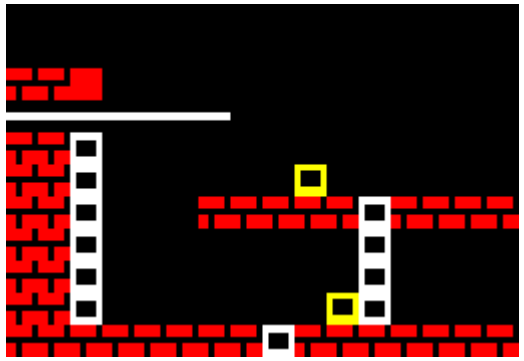
Zelda



Mega Man (MM)



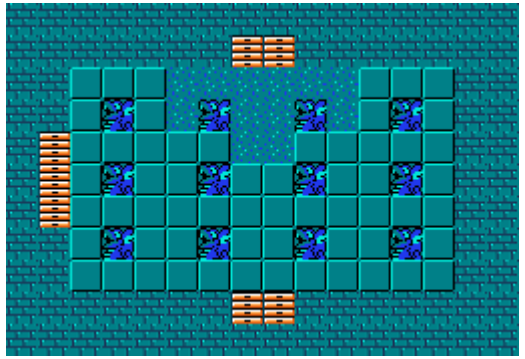
Metroid (Met)



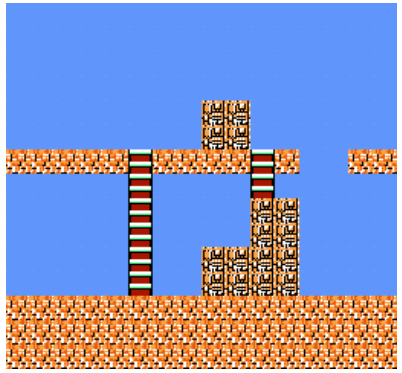
Lode Runner (LR)

- Levels are composed of sections connected together
- Zelda: Discrete rooms connected using doors to form dungeons
- Platformers: Discrete segments connected based on directionality of player movement
- Train CVAE on such discrete segments/rooms with labels indicating how they are connected

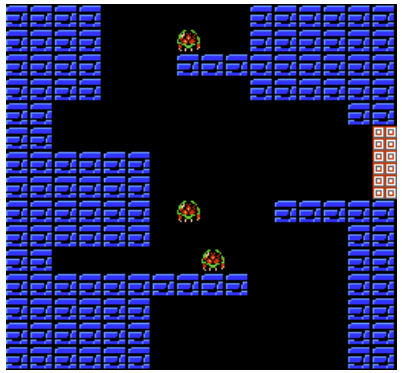
Level Representation and Conditioning



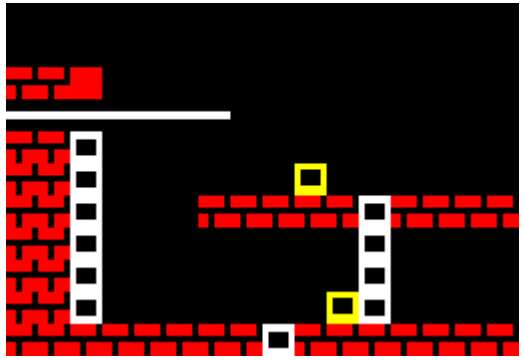
Zelda
<1, 1, 1, 0>



Mega Man (MM)
<0, 0, 1, 1>



Metroid (Met)
<1, 1, 0, 1>



Lode Runner (LR)
<0, 1, 1, 0>

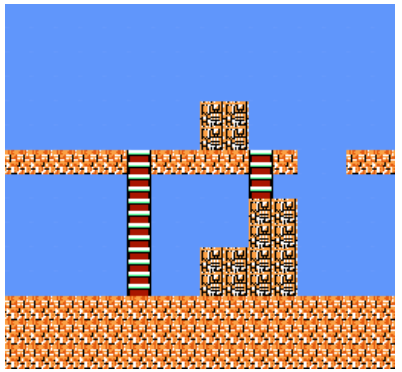
- Labels were binary-encoded vectors of length 4: <Up, Down, Left, Right>
- Each element in label corresponds to a direction (1 – open/door, 0 – closed/no door)
- Zelda: 11x16; labels indicate directions with doors
- Metroid/Mega Man: 15x16; labels indicate directions that are open
- Lode Runner: each 22x32 level divided into 11x16 quadrants; labels indicate direction of attachment to other quadrants

Game and Genre Blending

- For blending, extended labels to also specify games being blended

- For N-game blend, concatenate N-element vector to 4-element direction vector

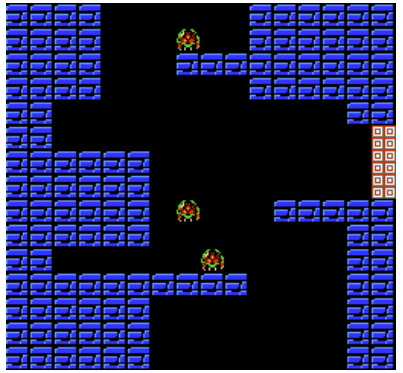
- Trained on 5 blends
  - Metroid-Mega Man
  - Zelda-Lode Runner
  - Zelda-Metroid
  - Zelda-Mega Man
  - Zelda-Mega Man-Metroid



Mega Man (MM)

Game: *<0, 1>* Dir: *<0, 0, 1, 1>*

<0, 1> + <0, 0, 1, 1> = <0, 1, 0, 0, 1, 1>



Metroid (Met)

Game: *<1, 0>* Dir: *<1, 1, 0, 1>*

<1, 0> + <1, 1, 0, 1> = <1, 0, 1, 1, 0, 1>

Evaluation

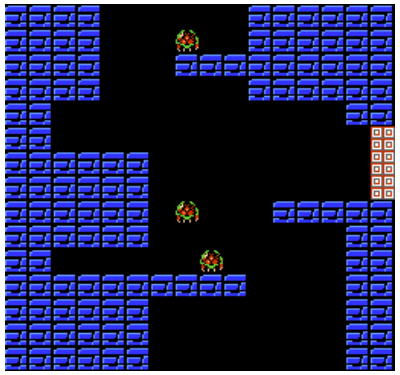
- Three-part evaluation
  - Directional Label Accuracy
  - Blending/Game Label Accuracy
  - Tile-based Metrics

Evaluation: Direction Labeling

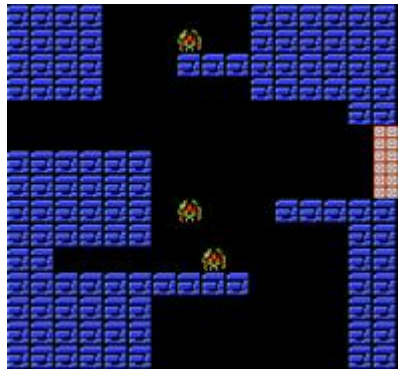
- Testing accuracy of directional labeling

- Zelda – check the location of door tiles in the generated rooms
- For Metroid, MM, LR
  - Trained random forest classifier using directionality labels as class labels
  - Compared predicted label of generated segment vs. label used to generate it
  - Ideally want *exact* matches but sufficient that generated segment has just the desired open directions (*admissible* matches)

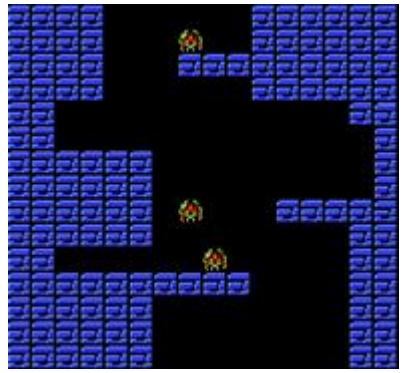
- Separately track match percentages for IN-game and OUT-of-game labels



<1, 1, 0, 1>
Original / Exact Match



<1, 1, 1, 1>
Admissible Match



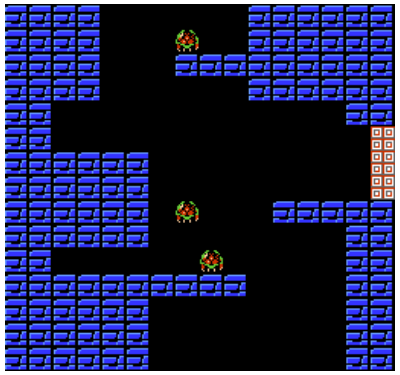
<1, 1, 0, 0>
Not A Match

Evaluation: Direction Labeling

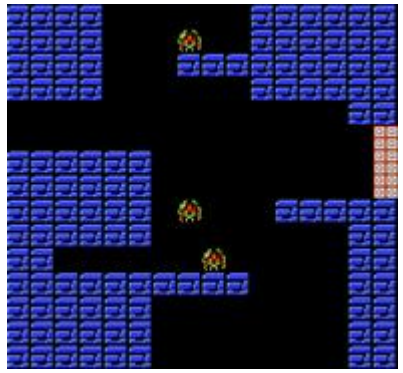
- Testing accuracy of directional labeling

- Zelda – check the location of door tiles in the generated rooms
- For Metroid, MM, LR
  - Trained random forest classifier using directionality labels as class labels
  - Compared predicted label of generated segment vs. label used to generate it
  - Ideally want *exact* matches but sufficient that generated segment has just the desired open directions (*admissible* matches)

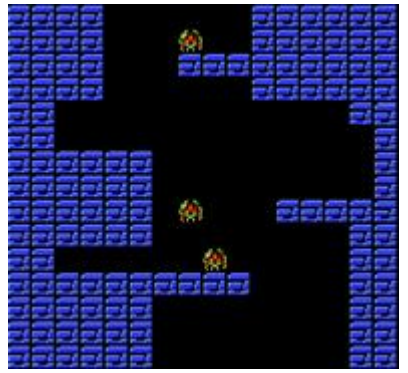
- Separately track match percentages for IN-game and OUT-of-game labels



<1, 1, 0, 1>
Original / Exact Match



<1, 1, 1, 1>
Admissible Match



<1, 1, 0, 0>
Not A Match

Evaluation: Direction Labeling

- For each model, sampled 1000 latents at random
- Used each of 16 possible labels for conditioning → 16000 generated segments per model
- For each generated segment, compared predicted label with label used for conditioning, separately computing %age of *exact* and *admissible* matches for IN and OUT labels

Evaluation: Direction Labeling

- For each model, sampled 1000 latents at random
- Used each of 16 possible labels for conditioning → 16000 generated segments per model
- For each generated segment, compared predicted label with label used for conditioning, separately computing %age of *exact* and *admissible* matches for IN and OUT labels

- Zelda achieved near 100% accuracy suggesting door placement is very reliable using CVAEs
- Next highest percentages are for LR IN-labels (4 out of 16), but poor for OUT labels
- Metroid outperformed MM in all 3 types of matches
- Takeaway: ~90% *admissible* matches when using IN labels suggesting that models can reliably produce openings/doors in desired directions, even if *exact* matches are not as frequent

	Zelda		Metroid			Mega Man			Lode Runner		
Latent	Exact	Admissible	Exact-IN	Admissible-IN	Admissible-OUT	Exact-IN	Admissible-IN	Admissible-OUT	Exact-IN	Admissible-IN	Admissible-OUT
4	99.81	99.84	**69.52**	90.96	71.15	**61.9**	88.81	27.9	**95.48**	**95.48**	39.36
8	**99.96**	**99.96**	69.28	**91.29**	**73.33**	61.62	**89.4**	27.87	94.78	94.78	39.97
16	99.85	99.91	67.63	90.38	72.13	58.9	86.57	28.14	94.78	94.78	39.9
32	99.95	**99.96**	62.34	89.33	64.6	53.78	83.21	**28.99**	94.78	94.78	**40.18**

Evaluation: Blend Labeling

- Evaluate labeling accuracy for blending

- Classifier-based evaluation for each blended model
  - Trained random forest classifier using the game that segment belonged to as class label
  - Sampled 100 latents and used each possible game+directional label to condition the generation of a segment
  - For 2-game blends, $2^{(2+4)}$ = 64 labels → 6400 generated segments
  - For 3-game blend, $2^{(3+4)}$ = 128 labels → 12800 generated segments
  - For each game label, computed %age of times classifier predicted each of the games

Evaluation: Blend Labeling

- Evaluate labeling accuracy for blending

- Classifier-based evaluation for each blended model
  - Trained random forest classifier using the game that segment belonged to as class label
  - Sampled 100 latents and used each possible game+directional label to condition the generation of a segment
  - For 2-game blends, $2^{(2+4)} = 64$ labels → 6400 generated segments
  - For 3-game blend, $2^{(3+4)} = 128$ labels → 12800 generated segments
  - For each game label, computed %age of times classifier predicted each of the games

- Expectations
  - Conditioning with an original game label <01>, <10>, <100>,<010>,<001>)
    --- very high % of predictions for game indicated by 1 in the label
  - Conditioning with blended game label (e.g. <110>, <101>)
    --- more variance among predictions
  - E.g. In <Zelda, Mega Man, Metroid> blend
    --- <100> → high % for Zelda
    --- <011> → low % for Zelda, moderate % for Mega Man and Metroid

Evaluation: Blend Labeling

- Obtained results true to expectation
  - In all cases of 1-game blend labels (i.e. <01>, <10>, <100>, <010>, <001>)
    - Close to 100% for 2-game blends involving Zelda
    - At least ~92% for Metroid-MM
    - At least ~87% for Zelda-Metroid-MM
  - More spread-out predictions when using blend labels i.e. labels with multiple 1s
  - When a game's label element is set to 0, the game is usually predicted less than 1% of the time (except <0,0,0>)

- Takeaway: conditioning can successfully blend desired games

Evaluation: Tile-based Metrics

- Evaluating actual content of generated segments
- Density and Symmetry (normalized 0-1)
- For each game, we compared the mean densities and symmetries of the training segments with those of generated segments (values in bold were significantly different from original)

Zelda	Original	4	8	16	32
Density	0.59 ± 0.08	0.58 ± 0.06	0.58 ± 0.06	$\mathbf{0.59 \pm 0.06}$	0.59 ± 0.07
Symmetry	0.76 ± 0.06	0.76 ± 0.04	0.76 ± 0.04	0.76 ± 0.05	0.75 ± 0.05
Metroid	Original	4	8	16	32
Density	0.39 ± 0.16	$\mathbf{0.41 \pm 0.09}$	$\mathbf{0.4 \pm 0.08}$	$\mathbf{0.41 \pm 0.08}$	$\mathbf{0.4 \pm 0.08}$
Symmetry	0.67 ± 0.12	$\mathbf{0.59 \pm 0.04}$	$\mathbf{0.6 \pm 0.05}$	$\mathbf{0.6 \pm 0.05}$	$\mathbf{0.61 \pm 0.05}$
Mega Man	Original	4	8	16	32
Density	0.38 ± 0.18	0.4 ± 0.13	0.41 ± 0.13	0.39 ± 0.14	0.39 ± 0.13
Symmetry	0.65 ± 0.13	0.62 ± 0.08	0.63 ± 0.07	0.64 ± 0.09	0.63 ± 0.07
Lode	Original	4	8	16	32
Density	0.41 ± 0.17	0.39 ± 0.14	0.39 ± 0.15	$\mathbf{0.38 \pm 0.15}$	0.39 ± 0.15
Symmetry	0.52 ± 0.14	0.51 ± 0.1	0.53 ± 0.1	$\mathbf{0.53 \pm 0.11}$	$\mathbf{0.53 \pm 0.11}$

Evaluation: Tile-based Metrics

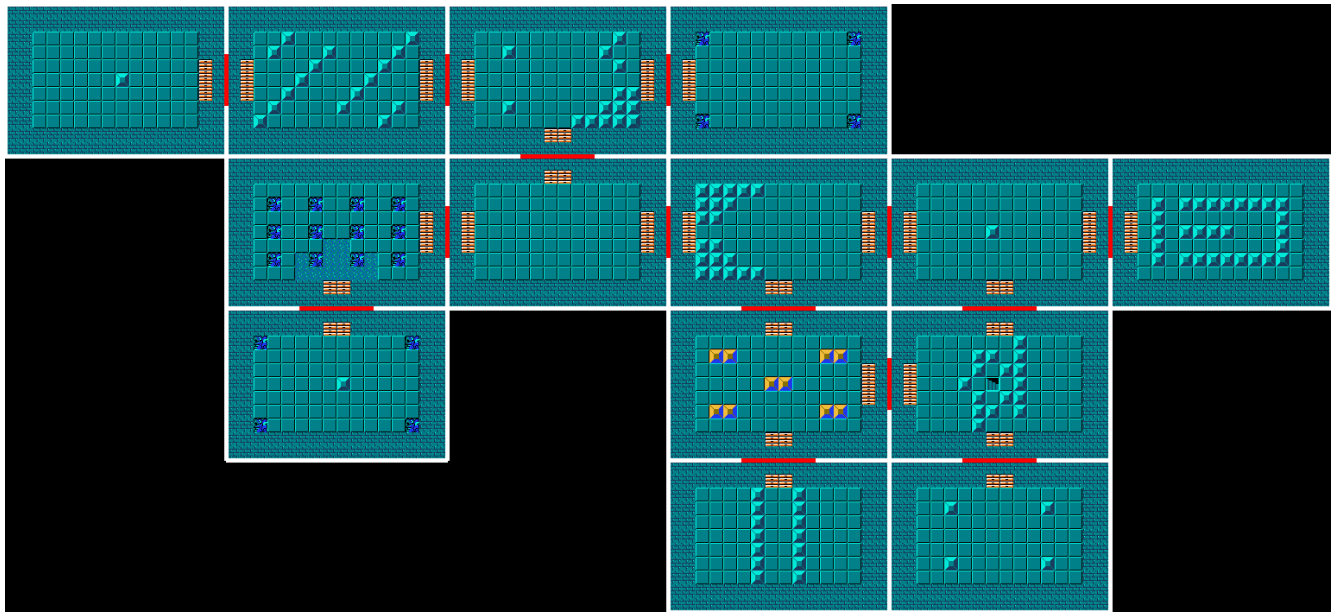
- Evaluating actual content of generated segments
- Density and Symmetry (normalized 0-1)
- For each game, we compared the mean densities and symmetries of the training segments with those of generated segments (values in bold were significantly different from original)

Additional evaluations in the paper:

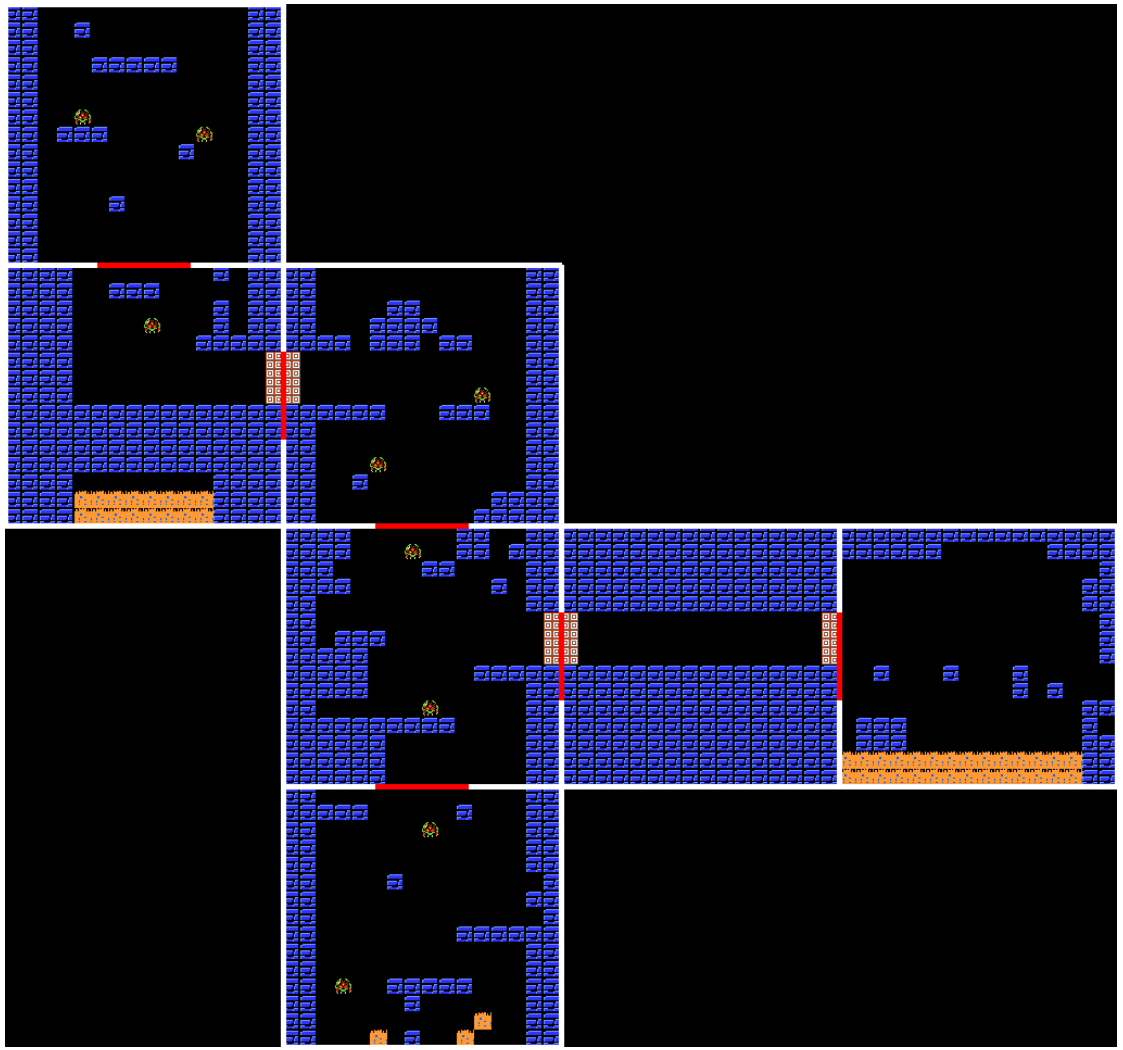
- Novelty of generated segments
- Content of blended segments in terms of distribution similarity

Zeld					32
Densi					$\pm$ 0.07
Symm					$\pm$ 0.05
Metr					32
Densi					**$\pm$ 0.08**
Symm					**$\pm$ 0.05**
Mega					32
Densi					$\pm$ 0.13
Symmetry	0.65 ± 0.13	0.62 ± 0.08	0.63 ± 0.07	0.64 ± 0.09	0.63 ± 0.07

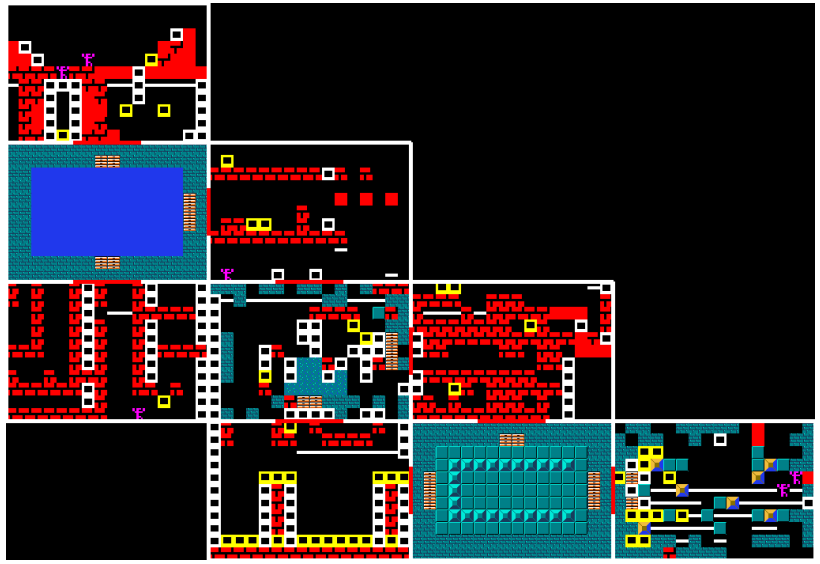
Lode	Original	4	8	16	32
Density	0.41 ± 0.17	0.39 ± 0.14	0.39 ± 0.15	$\mathbf{0.38 \pm 0.15}$	0.39 ± 0.15
Symmetry	0.52 ± 0.14	0.51 ± 0.1	0.53 ± 0.1	$\mathbf{0.53 \pm 0.11}$	$\mathbf{0.53 \pm 0.11}$



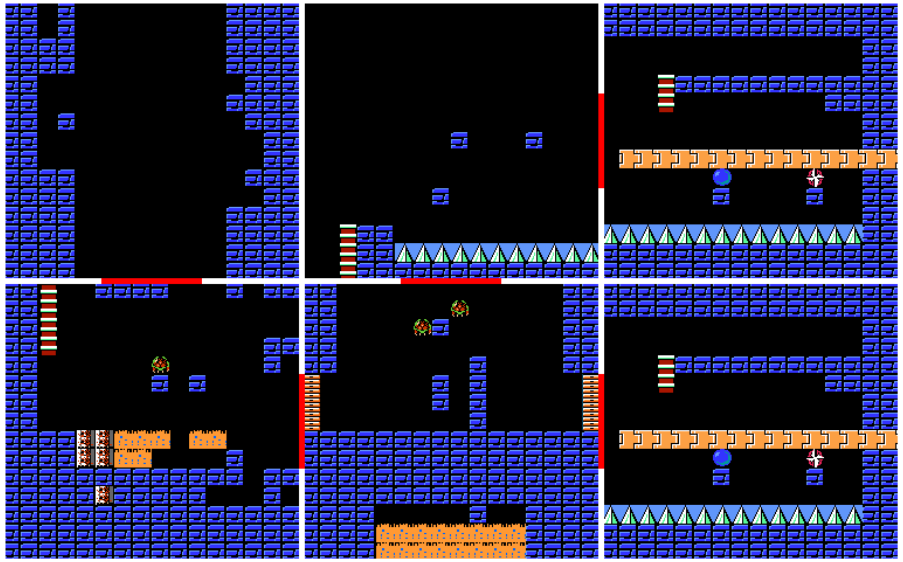
Generated Zelda Level



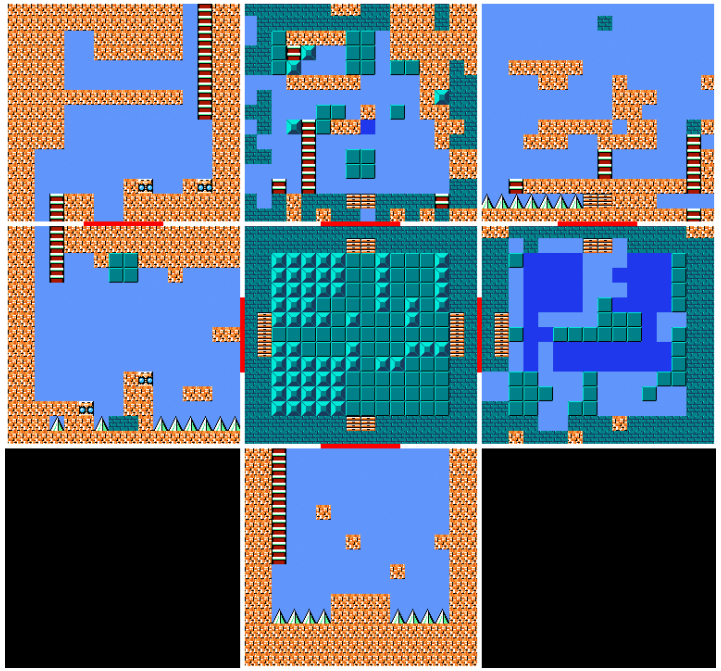
Generated Metroid Level



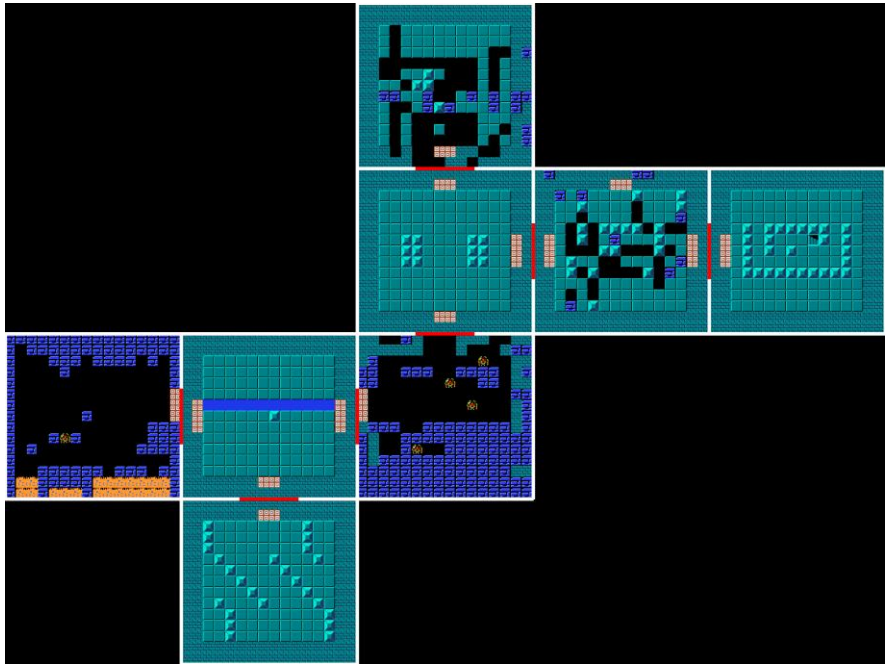
Zelda+Lode Runner



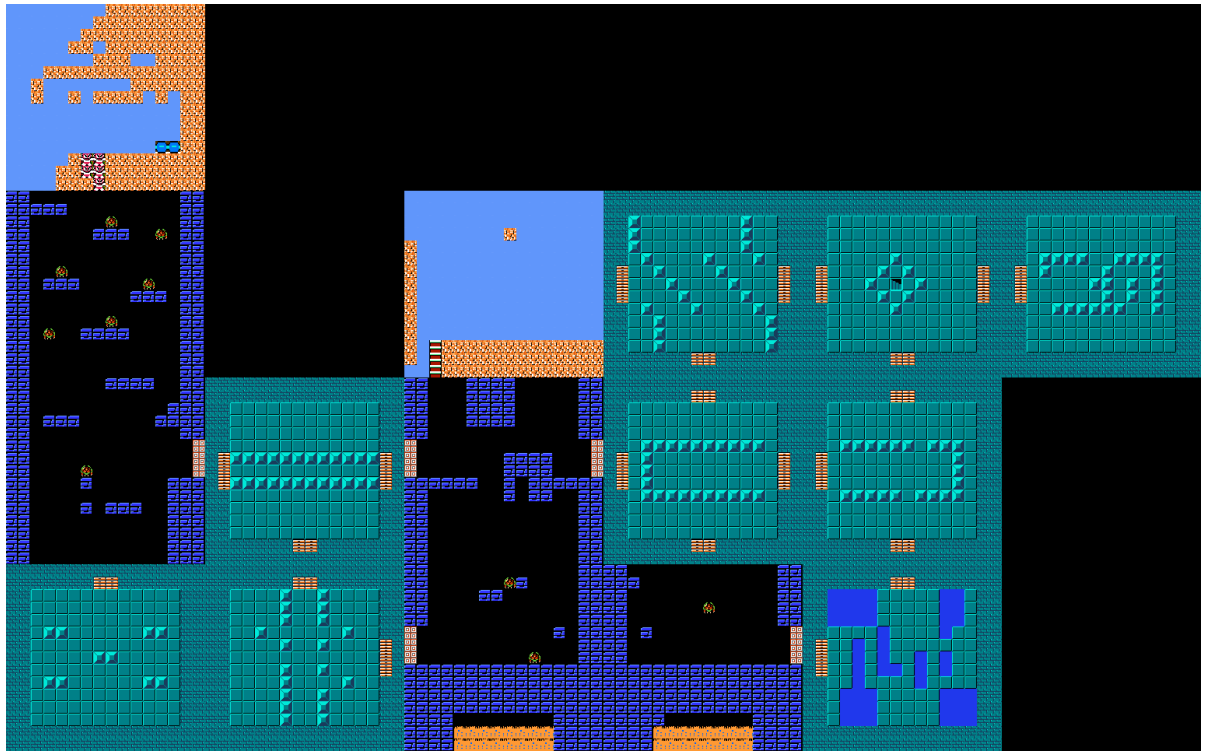
Metroid+Mega Man

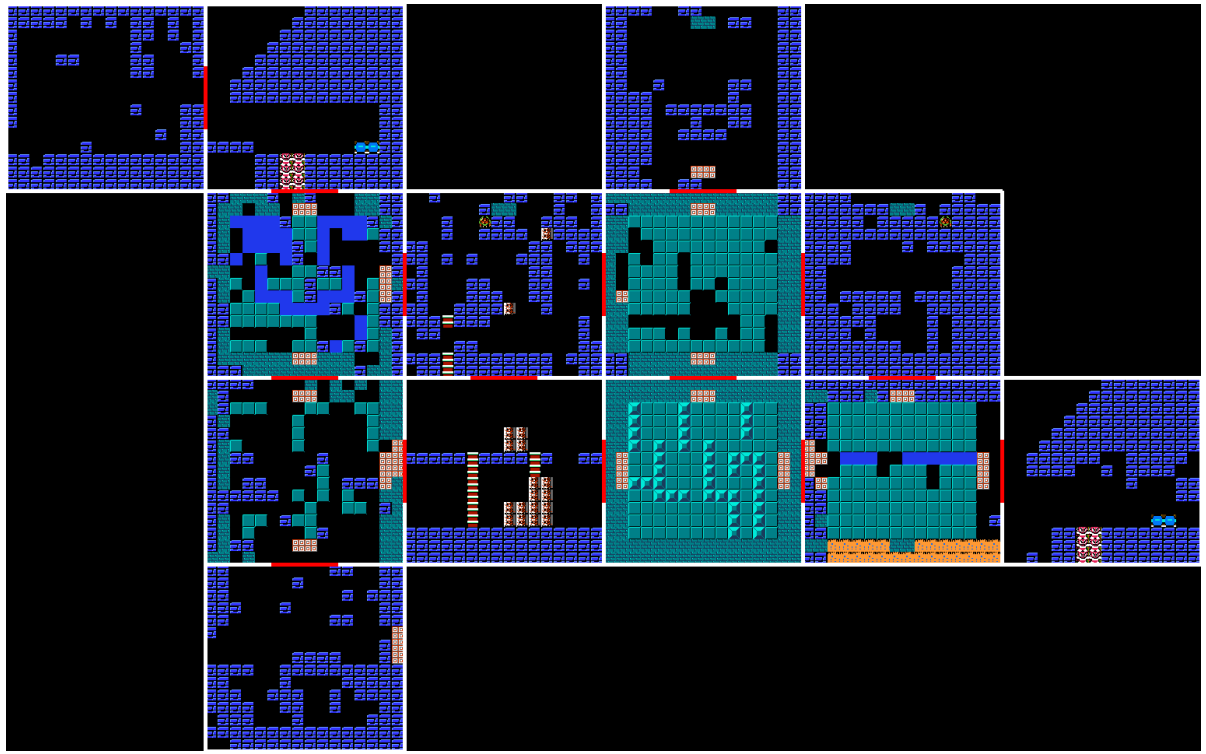


Zelda+Mega Man



Zelda+Metroid





Zelda+Metroid+Mega Man

Conclusion

- Used Conditional VAEs to control directionality of generated segments and rooms
- Enabled generation of whole platformer levels and dungeons
- Enabled blending across different genres

Limitations and Future Work

- Lack of playability evaluations
  - Zelda dungeons in VGLC don’t have lock and key tiles/triforce
  - Hard to test playability for Lode Runner segments
  - Blended levels warrant new, blended mechanics

- Generate fully playable dungeon-platformer blends
  - Use process to generate new mechanics to make blended levels playable
  - Or, use an agent capable of certain mechanics to repair blended levels

- Incorporate into mixed-initiative or automated design tools

- Investigate effect of latent size

Limitations and Future Work

- Lack of playability evaluations
  - Zelda dungeons in VGLC don’t have lock and key tiles/triforce
  - Hard to test playability for Lode Runner segments
  - Blended levels warrant new, blended mechanics

- Generate fully playable dungeon-platformer blends
  - Use process to generate new mechanics to make blended levels playable
  - Or, use an agent capable of certain mechanics to repair blended levels

- Incorporate into mixed-initiative or automated design tools

- Investigate effect of latent size

Contact
Anurag Sarkar
Northeastern University
sarkar.an@northeastern.edu