Controllable Level Blending between Games using Variational Autoencoders

Anurag Sarkar,[1] Zhihan Yang[2] and Seth Cooper[1]
[1]Northeastern University, Boston, Massachusetts, USA
[2]Carleton College, Northfield, Minnesota, USA
sarkar.an@husky.neu.edu, yangz2@carleton.edu, se.cooper@northeastern.edu

Abstract

Previous work explored blending levels from existing games to create levels for a new game that mixes properties of the original games. In this paper, we use Variational Autoencoders (VAEs) for improving upon such techniques. VAEs are artificial neural networks that learn and use latent representations of datasets to generate novel outputs. We train a VAE on level data from *Super Mario Bros.* and *Kid Icarus*, enabling it to capture the latent space spanning both games. We then use this space to generate level segments that combine properties of levels from both games. Moreover, by applying evolutionary search in the latent space, we evolve level segments satisfying specific constraints. We argue that these affordances make the VAE-based approach especially suitable for co-creative level design and compare its performance with similar generative models like the GAN and the VAE-GAN.

Introduction

Procedural content generation (PCG) refers to automated generation of content for games and has been a popular area of research (Shaker, Togelius, and Nelson 2016) with methods for PCG using a variety of techniques, most commonly search (Togelius et al. 2011), grammars (Smith, Whitehead, and Mateas 2011) and constraint solving (Smith and Mateas 2011). A newly emerging subfield of PCG research is PCGML or PCG via Machine Learning (Summerville et al. 2017), referring to procedurally generating content via models trained on existing game data using machine learning. This allows models to capture properties of games that designers may want to emulate while also helping alleviate the burden of hand-crafting rules and reducing designer bias.

Recently, Sarkar and Cooper (2018) showed that LSTMs trained on levels from *Super Mario Bros.* and *Kid Icarus* can generate levels whose properties are a mix of those of levels from the original games. This work leveraged past successes in ML-based level generation (Summerville et al. 2015) and blending (Guzdial and Riedl 2016b) to move towards implementing a co-creative system similar to the VGDL-based game blending framework proposed by Gow and Corneli (2015) wherein a novel game is generated by combining the mechanics and aesthetics of two existing games. While successful in demonstrating the feasibility of generating blended levels using models trained on data from separate games, the LSTM approach did blending by taking turns in generating segments of the two games. This allows generation of blended levels but not more fine-grained blending of elements from different games within a level segment. Also, though it lets designers specify the proportion of each game to blend in the levels, it does not let them control different level properties in the blending and generation process.

To address these issues, we use Variational Autoencoders (VAEs) to blend levels from different games and show that by capturing the latent design space across both games, the VAE enables more holistic blending of level properties, allows for the generation of level segments that optimize certain functions and satisfy specific properties, and is more conducive to co-creative level design. Specifically, we train a VAE on one level each from *Super Mario Bros.* and *Kid Icarus* which thus learns a latent representation spanning both games. Using evolutionary search in this space lets us evolve level segments satisfying different designer-specified constraints as well as the proportions of *Super Mario Bros.* and *Kid Icarus* elements present within them. Due to the nature of the VAE, designers can also specify the segments they want to blend or generate variations of without having to use the latent representation as in past related work using Generative Adversarial Networks (GANs) (Volz et al. 2018).

This work contributes an exploratory study on using VAEs to blend levels from different games. We present an evaluation of the approach that suggests that VAEs are better suited to blending levels across games than GANs and conclude with a discussion on how VAEs can inform co-creative level design in the future.

Background and Related Work

PCG via Machine Learning Recent years have seen significant research on 2D level generation using machine learning. N-gram models (Dahlskog, Togelius, and Nelson 2014), Markov models (Snodgrass and Ontañón 2014), PGMs (Guzdial and Riedl 2016a) and LSTMs (Summerville and Mateas 2016) have all been used to generate levels for *Super Mario Bros.* while Bayes Nets have been used for generating dungeons for *The Legend of Zelda* (Summerville and Mateas 2015). Similar to our work, Snodgrass and Ontañón (2017) use levels from multiple games, including the two that we used, but differ in using Markov chains rather than VAEs and in focusing on translating levels from one game to another rather than blending levels. Most closely related to our work is that of Jain et al. (2016) and Volz et al. (2018) in using autoencoders and GANs respectively to create *Super Mario Bros.* levels. The latter also applied Covariance Matrix Adaptation to evolve levels with certain properties within the GAN's latent space. We similarly train a VAE and evolve levels in its latent space but unlike GANs, which only accept latent space vectors as inputs, VAEs can accept as inputs both level segments as well as latent vectors, thus offering more control over generation.

Level and Game Blending Past work has focused on blending levels from the same game, specifically *Super Mario Bros.*, by applying principles of conceptual blending (Fauconnier and Turner 1998). Guzdial and Riedl (2016b) blended different Mario level generation models while also exploring different blending strategies for generating levels (Guzdial and Riedl 2017). Beyond levels, Gow and Corneli (2015) proposed a framework for blending entire games. They created a new game *Frolda* by combining the VGDL (Schaul 2014) specifications of *Frogger* and *Zelda*. To move towards using ML-based level blending to implement a similar framework but not restricted to VGDL, Sarkar and Cooper (2018) used LSTMs to blend together levels from *Super Mario Bros.* and *Kid Icarus*. In this paper, we build on this latter work by replacing LSTMs with VAEs. This enables more holistic level blending and the ability to use the VAE's learned latent space to interpolate between levels and evolve new ones based on various properties. Similar to blending games, recent work by Guzdial and Riedl (2018) looked into generating new games by recombining existing ones using a process termed *conceptual expansion*.

Co-Creative Generative Systems Co-creative generative systems (Yannakakis, Liapis, and Alexopoulos 2014) let human designers collaborate with procedural generators, enabling designers to guide generation towards desired content. Such systems have been used for generating platformer levels (Smith, Whitehead, and Mateas 2011), game maps (Liapis, Yannakakis, and Togelius 2013), *Cut-the-Rope* levels (Shaker, Shaker, and Togelius 2013) and dungeons (Baldwin et al. 2017). Within PCGML, the Morai Maker (Guzdial et al. 2017) lets users design Mario levels by collaborating with AI agents based on generative models from past PCGML work. We envision that VAEs can be the basis for similar tools that help designers in making levels that blend properties of multiple games.

Variational Autoencoders Autoencoders (Hinton and Salakhutdinov 2006) are neural nets that learn lower-dimensional representations of data using an unsupervised approach. They consist of an encoder which converts the data into this representation (i.e. the latent space) and a decoder which reconstructs the original data from it. Variational autoencoders (VAEs) (Kingma and Welling 2013) augment vanilla autoencoders by making the latent space model a probability distribution which allows learning a continuous latent space thus enabling random sampling of outputs as well as interpolation, similar to GANs. Thus a level generation approach as in MarioGAN (Volz et al. 2018) could be implemented using VAEs. Similar to how MarioGAN searched the latent space to find desired levels, the same may be possible with VAEs. Like our approach, Guzdial et al. (2018) used autoencoders for generation, focusing on generating level structures conforming to specified design patterns. Their work differs in generating Mario-only structures and using a standard autoencoder while we generate segments spanning both Mario and Icarus using a VAE.

Tile Type	*VGLC*	*Integer*	*Sprite*
SMB Ground	X	0	
SMB Breakable	S	1	
SMB Background	-	2	🟦
SMB Full Question	?	3	
SMB Empty Question	Q	4	
SMB Enemy	E	5	
SMB Pipe Top Left	<	6	
SMB Pipe Top Right	>	7	
SMB Pipe Bottom Left	[	8	
SMB Pipe Bottom Right	]	9	
SMB Coin	o	10	
KI Platform	T	11	
KI Movable Platform	M	12	
KI Door	D	13	
KI Ground	#	14	
KI Hazard	H	15	
KI Background	-	16	🟦

Table 1: Encodings used for level representation

Latent Variable Evolution via CMA-ES Latent variable evolution (Bontrager et al. 2018) refers to using evolutionary search to find desired vectors in latent space. MarioGAN uses Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) (Hansen, Muller, and Koumoutsakos 2003) to find desirable level segments within the latent space of the trained GAN. This allows generating segments that optimize specific features such as Tile Pattern KL-Divergence as introduced by Lucas and Volz (2019). We similarly used CMA-ES to evolve vectors in the latent space of our VAE.

Approach

For the remainder of the paper, we refer to *Super Mario Bros.* as SMB and *Kid Icarus* as KI.

Dataset Level data was taken from the Video Game Level Corpus (VGLC)[1] (Summerville et al. 2016) and consisted of data from one level (Level 1-1) of *SMB* and and one level (Level 5) of *KI*. We chose the former since it was used in MarioGAN and the latter as it contains all the KI elements which is not true for all KI levels in the corpus. We

[1]https://github.com/TheVGLC/TheVGLC

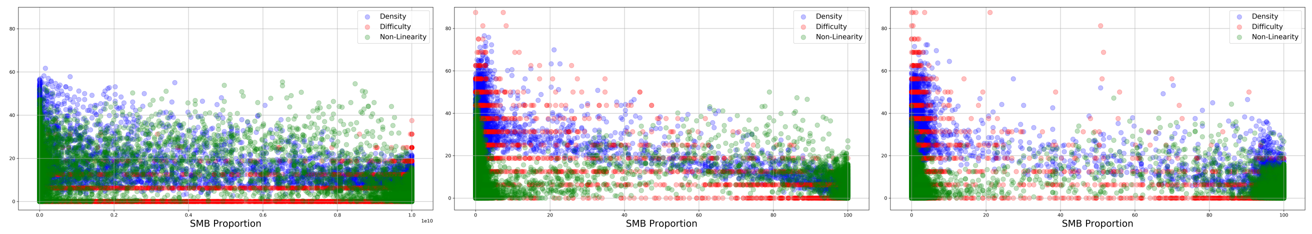

Figure 1: *SMB Proportion* vs *Density/Difficulty/Non-Linearity* for 10000 segments generated by VAE, GAN and VAE-GAN.

used only 1 level per game similar to MarioGAN and also due to the exploratory nature of our work, hoping for quick, promising results and leaving more optimal results derived from a larger corpus for future work. Regarding the choice of games, our ultimate goal is blending games from different genres. To do so, it is worth first blending games from the same genre that differ in significant ways. Thus, we use SMB and KI since both are platformers but differ in orientation. Each level in the VGLC is a 2D character array with each tile represented by a specific character. For training, each tile type was encoded first using an integer, as given in Table 1, and then using a One-Hot encoding. To create training samples, we used a 16x16 window slid horizontally across the SMB level and vertically across the KI level, thus training the VAE on 16x16 level segments from both games. We chose this window to account for the difference in orientation between the games. Hence, the VAE learned to generate 16x16 level segments rather than entire levels. This may be more conducive to a co-creative approach as it lets designers query the VAE for segments they can assemble as desired rather than the VAE constructing a level with a fixed rule for combining generated segments. In MarioGAN, this is not an issue since all segments are from SMB so a full level can be formed by placing them one after another. It is not obvious how to do so when segments themselves may be a blend of SMB (horizonatal) and KI (vertical) gameplay. Using the sliding window, we obtained 187 SMB and 191 KI training segments, for a total of 378 training segments.

Training The VAE encoder had 2 strided-convolutional layers using batchnorm and LeakyReLU activation. This fed into a 64-dimensional bottleneck (i.e. hidden) layer which fed into the decoder. The decoder had 1 non-strided followed by 2 strided-convolutional layers and also used batchnorm but used ReLU instead of LeakyReLU. We used the Adam optimizer with a learning rate of 0.001 and binary cross-entropy as the loss function. For evaluation, we compared the VAE with a GAN and a VAE-GAN. The GAN discriminator and generator had similar architectures to the VAE encoder and decoder respectively. The VAE-GAN's encoder and decoder had the same architecture as those of the VAE while its discriminator had the same architecture as that of the GAN. For all models, we used PyTorch (Paszke et al. 2017) and trained on the 2 levels for 10000 epochs based on MarioGAN using 5000 epochs to train on 1 level.

Generation The trained VAE generates 16x16 segments within the combined *SMB-KI latent level design space*. Generation involves feeding a 64-dimensional latent vector into the VAE's decoder which outputs a 17x16x16 one-hot encoded array representing the segment. Using argmax along the one-hot encoded dimension gives the 16x16 segment in the integer encoding as in Table 1. This can then be converted and stored as an image using tiles from the original games. Thus, like MarioGAN, the VAE can generate segments by using random latent vectors or by using CMA to evolve vectors that optimize given fitness functions, thereby satisfying designer-specified constraints. Unlike GANs, the VAE can also generate segments based on those supplied by designers rather than just using latent vectors. This involves feeding a segment encoded using the VGLC representation into the VAE's encoder which in turn encodes the segment into the learned 64-dimensional latent vector representation. A new segment can then be obtained from this vector using the decoder. Further, one can input two segments, get their corresponding vectors using the encoder and interpolate between them to generate new segments that blend the input segments. Due to these added capabilities, we argue that VAEs are more suited than GANs for co-creative level design systems based on blending different games as it allows designers more explicit control in defining the inputs to the system. Designers may find it more useful to blend or interpolate between segments they define or know the appearance of rather than do so by evolving latent vectors.

Evaluation

To evaluate our approach, we used the following metrics:

- *Density* - the number of solid tiles in a 16x16 segment. A segment with density of 100% has all 256 tiles as solid.
- *Difficulty* - the number of enemies plus hazards in a 16x16 segment. Based on the dimensions, we defined a segment with 100% difficulty to have 16 total enemies and hazards.
- *Non-Linearity* - measures how well segment topology fits to a line. It is the mean squared error of running linear regression on the highest point of each of the 16 columns of a segment. A zero value indicates perfectly horizontal or linear topology.
- *SMB Proportion* - the percentage of non-background SMB tiles in a segment. A segment with 100% *SMB Proportion* has only SMB tiles while 0% has only KI tiles.

These metrics are tile-based properties meant to visualize the generator's expressive range (Smith and Whitehead 2010) rather than encapsulate formal notions of density, difficulty or non-linearity. Additionally, we compared

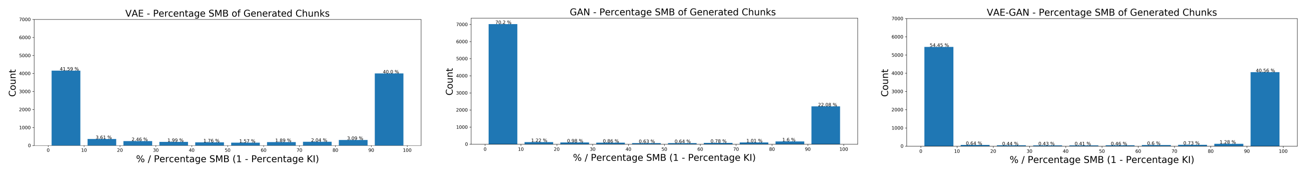
Figure 2: Proportions of SMB and KI elements in segments generated by VAE, GAN and VAE-GAN respectively

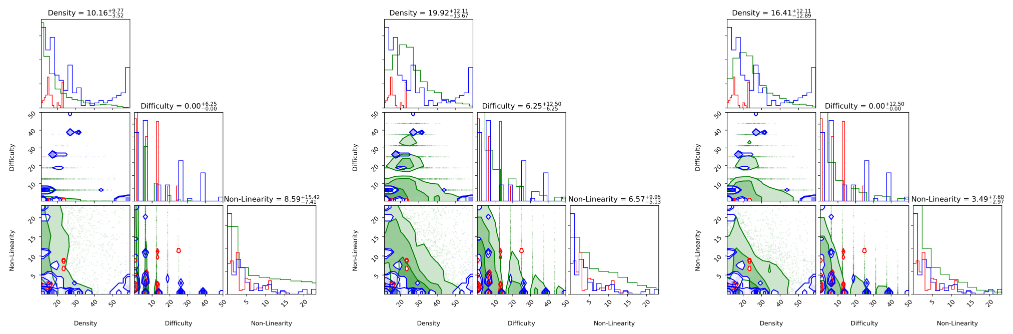
Figure 3: Corner plots for VAE, GAN and VAE-GAN respectively depicting generated (green), SMB (red) and KI (blue) levels.

the VAE's generative performance with that of similar generative models like the GAN and the VAE-GAN. While the GAN cannot encode known levels and segments, it may offer better generation and blending than the VAE in which case using a hybrid model such as the VAE-GAN (Larsen et al. 2016) that combines the benefits of both, may be best for blending levels. Thus, we compared these models in terms of their accuracy in evolving desired segments by using CMA-ES to evolve 100 level segments with target values of 0%, 25%, 50%, 75% and 100% for each of *Density*, *Difficulty*, *Non-Linearity* and *SMB Proportion* and compared the target values to the actual values of the evolved segments. Further, we compared the models in terms of capturing the latent space spanning both games. For this, we computed the above metrics for segments generated from 10,000 latent vectors drawn uniformly at random from a Gaussian distribution.

Results and Discussion

Figure 1 depicts the expressive range of the three models in terms of *Density*, *Difficulty*, *Non-Linearity* and *SMB Proportion* in generated segments. The VAE seems to be best at generating segments whose elements are a mix of those from either game while both the GAN and the VAE-GAN generate segments with mostly SMB or mostly KI elements as evidenced by their plots being sparser in the middle than the VAE plot. This is also suggested by Figure 2. Approximately 18% of VAE-generated segments have elements of both games where as this drops to around 8% for GAN and 5% for VAE-GAN. This implies that the VAE is better than the other models at capturing the latent space spanning both games as well as the space in between, thus making it the best choice among the three for generating blended segments. We also used corner plots (Foreman-Mackey 2016) to visualize the models as shown in Figure 3. Such plots have been used in past PCGML work (Summerville and Mateas 2016) and visualize the output of a model with respect to multiple metrics simultaneously. Based on these, VAE-generated segments adhere more closely to training segments and the space between segments from both games. While the increased variance displayed by segments generated by the GAN and VAE-GAN seems desirable for novelty, as we will discuss, this is due to the segments over-generalizing and ignoring the structures of training segments.

Results of testing the accuracy of evolving segments based on properties are shown in Figure 4. Initially, the GAN seems to perform best in terms of *Density*, *Difficulty* and *Non-Linearity*, followed by the VAE and then the VAE-GAN. It is worth noting though that the GAN does better than the VAE only for 100% *Density* and 75% and 100% *Difficulty*. However such values ignore the structures in training levels since actual SMB and KI segments would have neither 100% solid tiles nor 16 enemies and hazards. This suggests that the VAE's latent space better captures the nature of the training data than the GAN's latent space, thereby struggling to find segments with close to 100% *Density* while the GAN can do so more easily. It is possible that the GAN wasn't trained enough but since we used the same training data, similar architectures and the same number of epochs for both models, this is another benefit of the VAE as it exhibits better performance with similar training.

In terms of blending desired SMB and KI proportions, none of the models do particularly well in evolving segments that are neither 100% KI nor 100% SMB. However, while the VAE does well at least for 50%, the other two do much worse. These results follow from the discussion on Figure 2 and suggest that, with similar training, the VAE learns a latent space that is more representative of the game data (based on Figure 4) while having more variation to enable

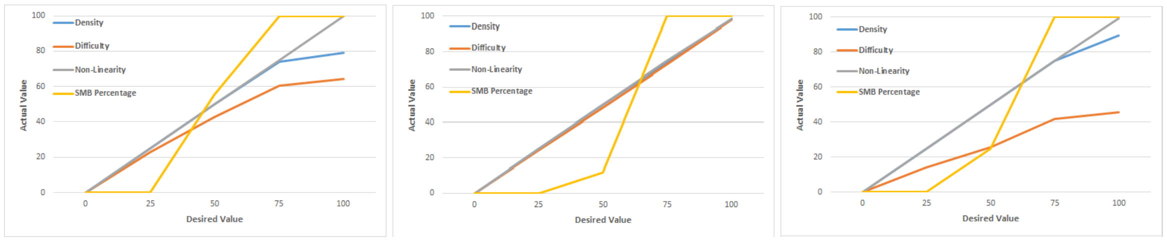

Figure 4: Accuracy of evolving level segments optimizing *Density*, *Difficulty*, *Non-Linearity* and *SMB Proportion*. Values are the average of 100 evolved segments for each desired value. Results for VAE, GAN, VAE-GAN from left to right.

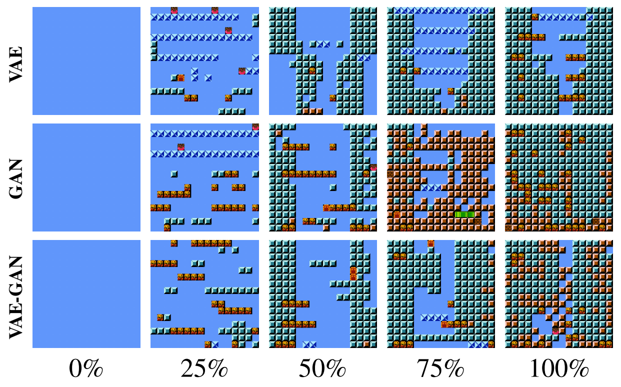

Figure 5: Evolved segments optimizing for *Density*.

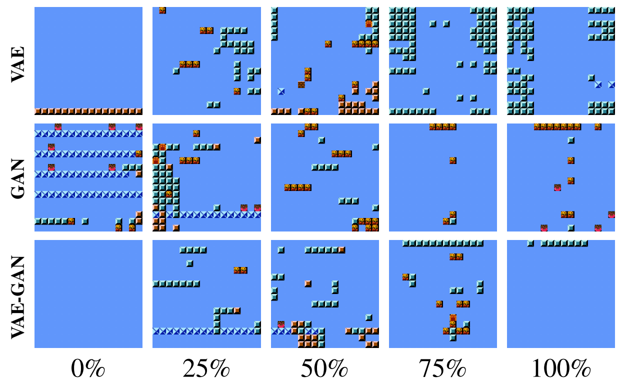

Figure 7: Evolved segments optimizing for *Non-Linearity*.

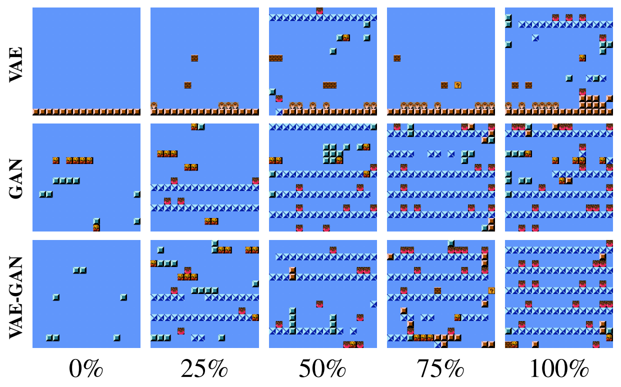

Figure 6: Evolved segments optimizing for *Difficulty*.

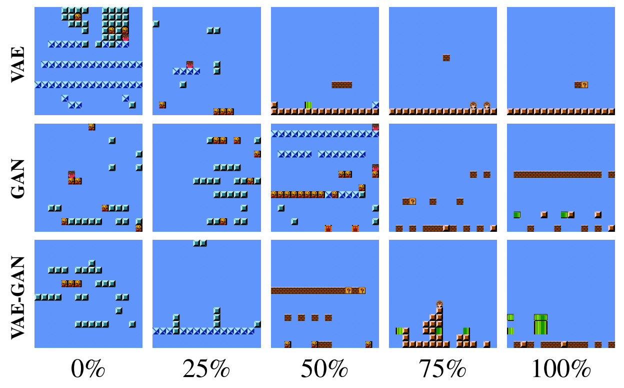

Figure 8: Evolved segments optimizing *SMB Proportion*.

better blending (based on Figures 1 and 2). Thus, in addition to enabling the encoding of segments, VAEs seem to also be better at generation than GANs in the context of level blending. Example evolved segments for each model for different objectives are shown in Figures 5, 6, 7 and 8. While future work should focus on improving the VAE architecture to better evolve segments spanning all values for *SMB Proportion*, overall, our findings suggest that VAEs are better suited to blending levels from different games than GANs.

Application in Co-Creative Design

VAEs trained on levels from multiple games could inform co-creative level design systems that let designers make levels by generating and blending level segments representative of all games used for training via the following affordances:

Interpolation between games By encoding level segments into latent vectors, VAEs enable interpolation between vectors representing segments from different games and generation of segments that lie between the latent space of either game, thus having properties of both games i.e. blended segments. For example, one could interpolate between an SMB segment and a KI segment as in Figure 9.

Alternative connections between segments Interpolating between segments from the same level can generate alternate connections between them. This can help designers edit existing levels by interpolating between two segments from the

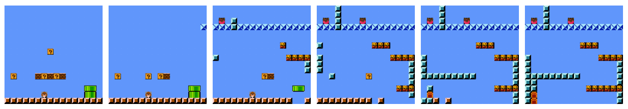
Figure 9: Transitioning from SMB (left) to KI (right) by interpolating between corresponding latent vectors.

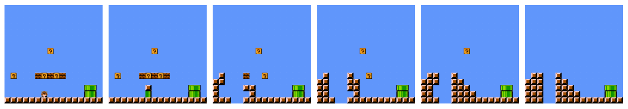
Figure 10: Interpolating between segments of Mario 1-1 generates those not in the actual level as in the middle four.

level to generate new segments that can be combined to form new links between the original two as in Figure 10.

Generating segments satisfying specific properties Search within the VAE's latent space can be used to evolve vectors and thus level segments satisfying specific properties. We saw *Difficulty*, *Density* and *Non-Linearity*, but other optimizations are possible such as maximizing the distribution of certain tiles as in Figure 11.

Generating segments with desired proportions of different games In addition to optimizing for tile-specific properties, we can also optimize for desired proportions of level elements from each game, as in Figure 8.

These affordances make VAEs suited to co-creative level design using latent spaces of multiple games. While GANs can generate segments optimizing specific properties, they do not offer the first two affordances as they only accept latent vectors as input. Also, based on our results, VAEs seem to be better at learning latent spaces spanning multiple games. Moreover, while the LSTM approach can generate entire blended levels, it does not offer any affordance except proportional blending between two games. Even so, this was done by varying the number of segments from each game rather than blending segments themselves. Currently, the VAE is unable to generate whole blended levels. When generating segments blending levels from games with different orientations, it is not obvious how to combine them. The LSTM approach addressed this by training a classifier on SMB and KI level sequences to determine if generated sequences were more SMB-like or KI-like, orienting them accordingly. However, this is harder when segments themselves are blended as in our case. While future work should augment the VAE with entire level generation capabilities, in a co-creative context, segments may offer designers more fine-grained control over level design than whole levels. They are likely to better affect level aesthetics by deciding on the placement of generated segments instead of using entire generated levels.

Limitations and Future Work

We trained a VAE on level data from *Super Mario Bros.* and *Kid Icarus* and argued for its use as the basis for a co-creative

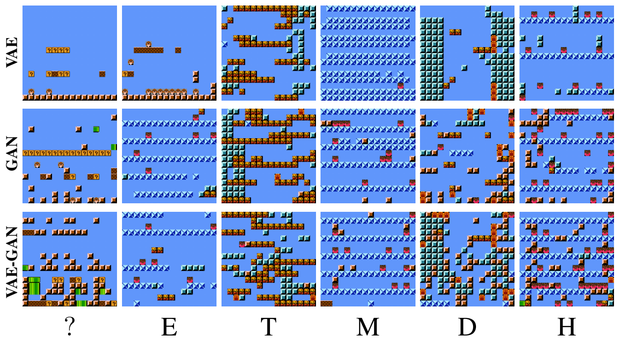

Figure 11: Evolved segments maximizing given tile type.

level design system that enables blending of levels from both games. We discuss limitations and future work below.

Playability The main limitation of this work is that it ignores playability. This may be less problematic when generating segments rather than whole levels as the designer can place segments such that the level is more playable. However, they should ideally be able to query the generator for segments that maintain playability relative to those already generated. This ties into the issue of blended levels possibly requiring new or blended mechanics to be playable. Investigating methods for blending mechanics is necessary future work and could leverage past work in mechanic generation. (Zook and Riedl 2014; Khalifa et al. 2017; Smith and Mateas 2010; Guzdial, Li, and Riedl 2017).

Vector Math in Level Design Space Besides interpolation, other vector operations like addition and subtraction can be used in the latent space to generate segments. Such vector arithmetic could enable feature transfer between games by for example adding a vector representing a game 1 feature to a vector representing a game 2 segment. Such interactions would require *disentanglement* i.e. different latent space dimensions encoding different level features. VAEs capable of learning such disentangled representations are called disentangled or β-VAEs (Higgins et al. 2017) and should be explored for this purpose in the future.

Level Design Tool In this work, we focused on validating the VAE approach to level blending. The next step is to implement the co-creative level design tool described in the previous section, using the VAE as its foundation.

Multiple Games and Genres Finally, future work could consider blended level design spaces spanning more than two games as well as multiple genres. How would one blend a platformer with an action-adventure game? What games, genres, mechanics and levels exist within the latent space between *Mario* and *Zelda*, for example?

References

Baldwin, A.; Dahlskog, S.; Font, J. M.; and Holmberg, J. 2017. Mixed-initiative procedural generation of dungeons using game design patterns. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*.

Bontrager, P.; Roy, A.; Togelius, J.; and Ross, A. 2018. Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*.

Dahlskog, S.; Togelius, J.; and Nelson, M. 2014. Linear levels through N-grams. In *Proceedings of the 18th International Academic MindTrek Conference: Media Business, Management, Content & Services*, 200–206.

Fauconnier, G., and Turner, M. 1998. Conceptual integration networks. *Cognitive Science* 22(2):133–187.

Foreman-Mackey, D. 2016. corner.py: Scatterplot matrices in Python. *The Journal of Open Source Software* 24.

Gow, J., and Corneli, J. 2015. Towards generating novel games using conceptual blending. In *Proceedings of the Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*.

Guzdial, M., and Riedl, M. 2016a. Game level generation from gameplay videos. In *Proceedings of the Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Guzdial, M., and Riedl, M. 2016b. Learning to blend computer game levels. In *Proceedings of the Seventh International Conference on Computational Creativity*.

Guzdial, M., and Riedl, M. 2017. Combinatorial creativity for procedural content generation via machine learning. In *Proceedings of the First Knowledge Extraction from Games Workshop*.

Guzdial, M., and Riedl, M. 2018. Automated game design via conceptual expansion. In *Proceedings of the Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Guzdial, M.; Chen, J.; Chen, S.-Y.; and Riedl, M. 2017. A general level design editor for co-creative level design. In *Proceedings of the AIIDE Workshop on Experimental AI in Games*.

Guzdial, M.; Reno, J.; Chen, J.; Smith, G.; and Riedl, M. 2018. Explainable PCGML via game design patterns. In *Proceedings of the AIIDE Workshop on Experimental AI in Games*.

Guzdial, M.; Li, B.; and Riedl, M. 2017. Game engine learning from video. In *IJCAI*.

Hansen, N.; Muller, S. D.; and Koumoutsakos, P. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation* 11(1):1–18.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; and Lerchner, A. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Hinton, G., and Salakhutdinov, R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Jain, R.; Isaksen, A.; Holmgård, C.; and Togelius, J. 2016. Autoencoders for level generation, repair and recognition. In *Proceedings of the ICCC Workshop on Computational Creativity and Games*.

Khalifa, A.; Green, M.; Perez-Liebana, D.; and Togelius, J. 2017. General video game rule generation. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*.

Kingma, D., and Welling, M. 2013. Auto-encoding variational Bayes. In *The 2nd International Conference on Learning Representations (ICLR)*.

Larsen, A.; Sonderby, S.; Larochelle, H.; and Winther, O. 2016. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on Machine Learning*.

Liapis, A.; Yannakakis, G.; and Togelius, J. 2013. Sentient Sketchbook: Computer-aided game level authoring. In *Proceedings of the 8th International Conference on the Foundations of Digital Games*.

Lucas, S. M., and Volz, V. 2019. Tile pattern kl-divergence for analysing and evolving game levels. In *Genetic and Evolutionary Computation Conference (GECCO)*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Sarkar, A., and Cooper, S. 2018. Blending levels from different games using LSTMs. In *Proceedings of the AIIDE Workshop on Experimental AI in Games*.

Schaul, T. 2014. An extensible description language for video games. *IEEE Transactions on Computational Intelligence and AI in Games* 6(4):325–331.

Shaker, N.; Shaker, M.; and Togelius, J. 2013. Ropossum: An authoring tool for designing, optimizing and solving cut the rope levels. In *Proceedings of the Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Shaker, N.; Togelius, J.; and Nelson, M. 2016. *Procedural Content Generation in Games*. Springer International Publishing.

Smith, A., and Mateas, M. 2010. Variations forever: Flexibly generating rulesets from a sculptable design space of minigames. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games (CIG)*.

Smith, A., and Mateas, M. 2011. Answer set programming for procedural content generation. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):187–200.

Smith, G., and Whitehead, J. 2010. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*.

Smith, G.; Whitehead, J.; and Mateas, M. 2011. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):201–215.

Snodgrass, S., and Ontañón, S. 2014. Experiments in map generation using Markov chains. In *Proceedings of the 9th International Conference on the Foundations of Digital Games*.

Snodgrass, S., and Ontañón, S. 2017. An approach to domain transfer in procedural content generation of two-dimensional videogame levels. In *Proceedings of the Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)*.

Summerville, A., and Mateas, M. 2015. Sampling Hyrule: Sampling probabilistic machine learning for level generation. In *Proceedings of the Eleventh Artificial Intelligence and Interactive Digital 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 on DiGRA and FDG*.

Summerville, A.; Behrooz, M.; Mateas, M.; and Jhala, A. 2015. The learning of Zelda: Data-driven learning of level topology. In *Proceedings of the 10th International Conference on the Foundations of Digital Games*.

Summerville, A. J.; Snodgrass, S.; Mateas, M.; and Ontañón, S. 2016. The VGLC: The Video Game Level Corpus. *Proceedings of the 7th Workshop on Procedural Content Generation*.

Summerville, A.; Snodgrass, S.; Guzdial, M.; Holmgård, C.; Hoover, A.; Isaksen, A.; Nealen, A.; and Togelius, J. 2017. Procedural content generation via machine learning (PCGML). In *Foundations of Digital Games Conference*.

Togelius, J.; Yannakakis, G.; Stanley, K.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):172–186.

Volz, V.; Schrum, J.; Liu, J.; Lucas, S.; Smith, A.; and Risi, S. 2018. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Yannakakis, G.; Liapis, A.; and Alexopoulos, C. 2014. Mixed-initiative co-creativity. In *Foundations of Digital Games Conference*.

Zook, A., and Riedl, M. 2014. Automatic game design via mechanic generation. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.