Lode Enhancer: Level Co-creation Through Scaling

Debosmita Bhaumik
Institute of Digital Games
Msida, Malta
debosmita.bhaumik01@gmail.com

Julian Togelius
Game Innovation Lab, modl.ai
USA, Denmark
julian@togelius.com

Georgios N. Yannakakis
Institute of Digital Games, modl.ai
Malta, Denmark
georgios.yannakakis@um.edu.mt

Ahmed Khalifa
Institute of Digital Games
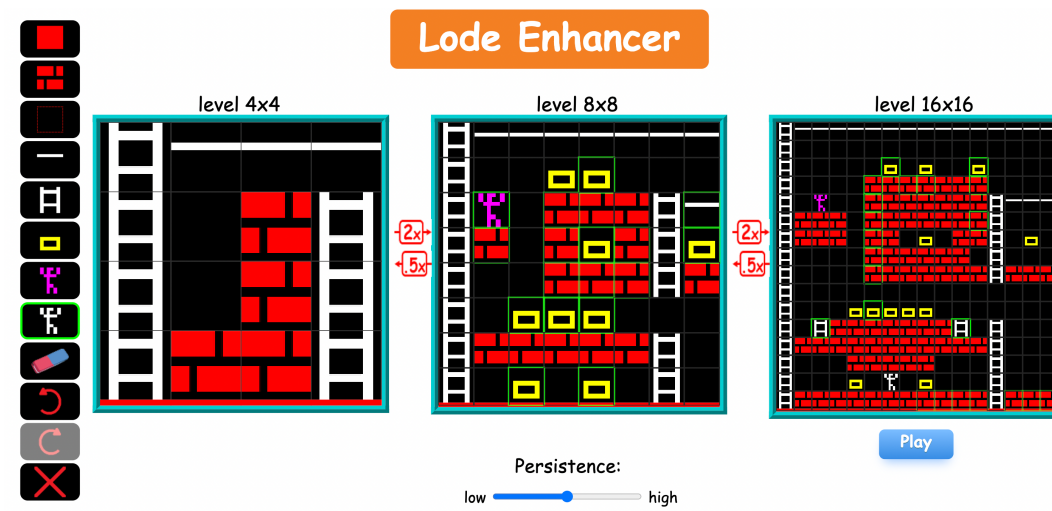Msida, Malta
ahmed@akhalifa.com

Figure 1: Lode Enhancer user interface with an example of an up-scaled level. The system is always running whenever a user makes a change in any canvas the other two get updated. The persistence bar helps to inform the system on which tiles can be replaced during scaling.

ABSTRACT

We explore AI-powered upscaling as a design assistance tool in the context of creating 2D game levels. Deep neural networks are used to upscale artificially downscaled patches of levels from the puzzle platformer game *Lode Runner*. The trained networks are incorporated into a web-based editor, where the user can create and edit levels at three different levels of resolution: 4x4, 8x8, and 16x16. An edit at any resolution instantly transfers to the other resolutions. As upscaling requires inventing features that might not be present at lower resolutions, we train neural networks to reproduce these features. We introduce a neural network architecture that is capable of not only learning upscaling but also giving higher priority to less frequent tiles. To investigate the potential of this tool and guide further development, we conduct a qualitative study with 3 designers to understand how they use it. Designers enjoyed co-designing with the tool, liked its underlying concept, and provided feedback for further improvement.

CCS CONCEPTS

• **Applied computing** → **Computer games**; • **Computing methodologies** → **Neural networks**; **Supervised learning**.

KEYWORDS

neural networks, mixed-initiative, supervised learning, procedural content generation, upscaling

1 INTRODUCTION

Artificial Intelligence (AI)-powered design assistance can take many forms, thereby, experimenting with the relation between human user(s) and AI systems is crucial for establishing human-AI co-creativity. A common trope is that of the human designing something relatively small or lacking detail (i.e. *downscaled* design), and the AI system producing a larger and more detailed (i.e. *upscaled*) version of the human design [21]. This entails that the AI system has some way of making up for the missing information.

In this paper, we explore the idea of using AI-powered upscaling (and, to a lesser extent, downscaling) for design assistance. The core idea is that a designer can design at any level or resolution (i.e. scale) and the AI system will intelligently up- and downscale it as necessary. In particular, a designer can draw a game-level sketch at a small scale and have it upscaled to a complete level, and then edit the upscaled level which gets automatically scaled down so it reflects the complete level.

The prototype system we describe in this paper, *Lode Enhancer*, uses a neural architecture to scale up level sketches from 4x4 to 8x8 tiles, and again from 8x8 to 16x16 tiles (figure 1). The neural networks are trained to upscale downscaled versions of existing Lode Runner level segments, meaning they have learned common Lode Runner level design patterns. The complete system, including trained networks, runs in the browser and produces near-instant results on modern computers. To understand the potential of AI-powered upscaling and downscaling, we did a quantitative analysis on the generated levels then followed by an informal study where we let three game designers use *Lode Enhancer* and interviewed them about their impressions.

2 LODE RUNNER

Lode Runner (Broderbund, 1983) is a classic puzzle platformer game where the goal of the game is to collect all gold pieces without being caught by the enemies. The player can only move left and right and dig through the floor. To go to higher platforms, the player needs to climb ladders. The digging mechanic allows the player to travel downward and it can be used to trap enemies.

Lode Runner comes with 150 levels which makes it a good candidate for procedural level generation using machine learning (PCGML) [18]. Also, the spatial dependencies between different tiles make this game a good test bed for our experiment. Snodgrass and Ontanón [16] used multidimensional markov chain to generate levels for different games including Lode Runner. Thakkar et al. [20] used a trained vanilla and variational autoencoder to generate Lode Runner levels and used evolution strategies to search for playable levels. Steckel and Schrum [17] trained a GAN on Lode Runner levels and used the MAP-Elites algorithms to search the space for diverse playable levels.

3 MIXED-INITIATIVE PCG

There exists various literature on procedural content generation, in particular on the autonomous generation of levels where the system produces levels with minimal human input [14]. In many settings, it is more useful to have a system that can interact with humans, so that a human user and an AI system can design together [9, 22]. Many of these systems are based on making suggestions to the user,

evaluating the user's edits, and/or enforcing constraints of various kinds. For example, Tanagra [15] uses constraint solving to guarantee playability in user designs, and Sentient Sketchbook evaluates strategy maps for various kinds of balance and uses evolutionary algorithms to suggest changes to the maps [11].

While most early mixed-initiative game design assistants relied on search, optimization, and/or constraint solving, a new generation of these tools build on machine learning. For example, RL brush [4] gives level editing suggestions to users generated by reinforcement learning agents, and Lode Encoder [2] allows users to mix and match between suggested levels using variational autoencoders and incentivizes users to create new levels. In Morai Maker [6], the human user and the AI agent collaborate in a turn-based process to generate Super Mario Bros (Nintendo, 1985) levels.

4 MULTILEVEL SCALE MACHINE LEARNING

A staple of sci-fi movies and crime shows, image upscaling [21], is now a commonplace technique that is used both on its own [5] and as part of image generation workflows [13]. Neural networks of various types can be trained to upscale images simply by using datasets where an artificially downscaled image is the input and the original image is the target. As the network learns to upscale, it learns to reproduce the various aspects of a high-resolution image that is not part of its low-resolution counterpart.

Upscaling is commonly used in image generation pipelines. This includes the StyleGAN [8] family of networks, where original images are generated at a low resolution and then upscaled by successive networks. Similarly, Stable Diffusion [13] uses trained upscaling diffusion models to generate high-quality images. Upscaling has not been applied much to mixed-initiative game-level generation yet. The most closely related work is Sentient World [10], where the user designs a low-resolution map, then the system suggests an upscaled version using neuroevolution through novelty search. In Sentient World, the generated maps are a continuous value for the height map which makes the problem similar to images compared to generating levels for tile-based games where there is no correlation between the tile values.

Game level design poses a significantly different problem than image generation, because of the functionality criteria; an image does not need to "work", but a level needs to be completable. For levels based on fixed-size assets (such as tiles/voxels), there is also the phenomenon that this functionality constraint might only exist at the full-resolution version of the level, as the compressed levels are almost not completable. Another common challenge is the discrete aspect of the domain. When you scale up a level it does not mean that the surrounding tiles/voxels suppose to be similar. For example, having one enemy in a low-resolution level does not mean that we need 10 enemies at this position if the level got enlarged 10 times.

5 LODE ENHANCER

Lode Enhancer[1] is an AI-powered level design tool that helps game designers to create "Lode Runner" levels through upscaling. Using the system, the user can draw levels on a small canvas then the system upscales it in an intelligent way and produces a larger and

---

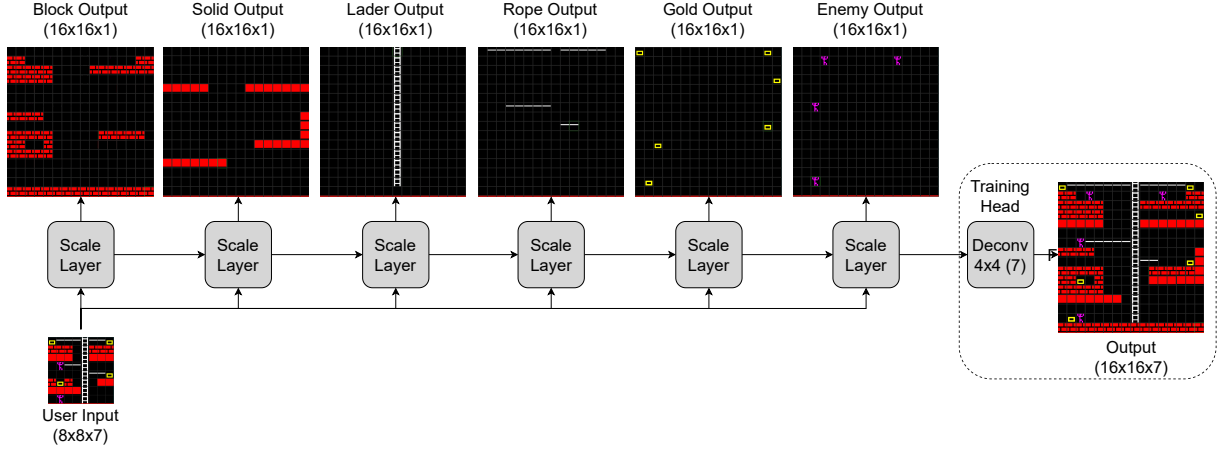[1]http://www.akhalifa.com/lodeenhancer/

Figure 2: The layer scaling network architecture. The network consists of 6 scale layers (one for each tile type) followed by a temporary Deconv layer to help train the network. The input for the first scale layer is ignored and only the user input is used.

more detailed level. Figure 1 shows the full system UI where the user is provided with three different canvases of different scales (4x4, 8x8, and 16x16). The user has a toolbar on the left that can be used to pick a certain tile to draw. Whenever any change happens to any canvas, the canvas is sent to the scaler module which reflects these changes to the other two canvases. Finally, the scaler module uses the persistence module (shown as a slider at the bottom of the UI) to know which tiles can be replaced and which ones should be kept the same.

5.1 Scaler Module

The scaler module is responsible for reflecting any changes that the user makes in any canvas to the other canvases. This is done using an up-scaler system and a down-scaler system. The up-scaler system is responsible for doubling the level size while the down-scaler system is responsible for halving the level size. For example, if the user changes the 8x8 canvas, the system will use the down-scaler to reflect these changes in the 4x4 canvas and the up-scaler to reflect the changes in the 16x16 canvas.

The up-scaler system is modeled using a neural network called layer scaling network (explained in section 6), while the down-scaler uses a traditional nearest neighbor filter to shrink the level. Originally, both systems were modeled using neural networks but in early experiments, we discovered that the down-scaler network learns to almost replicate the nearest neighbor filter.

5.2 Persistence Module

In our early experiments, we found that the user's edits in the 16x16 canvas got overwritten by the scaler module if the user went back and modified the 8x8 or 4x4 canvas. This problem prevented a lot of early testers from going back and modifying the 8x8 or 4x4 canvas after making edits to the 16x16 canvas. To solve this, we introduced the persistence slider that can be seen at the bottom of figure 1. The persistence slider tells the AI how much it should respect the user's edits. For high persistence, the user's edits get the highest

priority and shouldn't be replaced. While in low persistence, the AI can overwrite easily any tile.

The module calculates a confidence value (between 0.5 and 1.0) for each user-drawn tile. To replace a tile, the scaler module compares the probability of the tile from the neural network with its confidence value. The tile gets replaced if the network's probability is higher than the confidence value of the same tile. To calculate this confidence value, we take into account the age of the tile (i.e. how long ago the tile was drawn) as we want the older tiles to be easier to replace than newly drawn tiles. Equation 1 shows how the confidence value (C) is calculated given the age of the tile (a) where C_{max} is the maximum confidence value, a_{min} is the age at which the tile confidence start decreasing linearly, and a_{max} is the age at which the tile can be replaced.

$$C(a) = \begin{cases} C_{max} & a \leq a_{min} \\ \frac{C_{max}-0.5}{a_{max}-a_{min}} \cdot (a_{max}-a) + 0.5 & a_{min} > a \leq a_{max} \\ 0.5 & a > a_{max} \end{cases} \quad (1)$$

Moving the persistence bar changes the a_{min}, the a_{max}, and the C_{max} values. For the lowest persistence a_{min} is 0, a_{max} is 1, and C_{max} is 0.5, while for the highest persistence a_{min} is 20, a_{max} is 100, and C_{max} is 1. The slider just linearly interpolates these values for the in-between ticks.

6 LAYER SCALING NETWORK

Layer scaling network is a new network architecture (shown in figure 2) designed to help with discrete domains with uneven distribution between the different possible values/tiles. This new architecture gives different priorities and computation capacities for the different tiles. For example, in Lode Runner, we want the gold tiles and enemy tiles to have higher priority than normal brick tiles.

To achieve this, our network consists of $n-1$ scale layers (explained in section 6.1) where n is the number of possible tiles. Each layer takes the main user input and outputs a probability of a specific tile. The order of the output is picked based on the rarity of
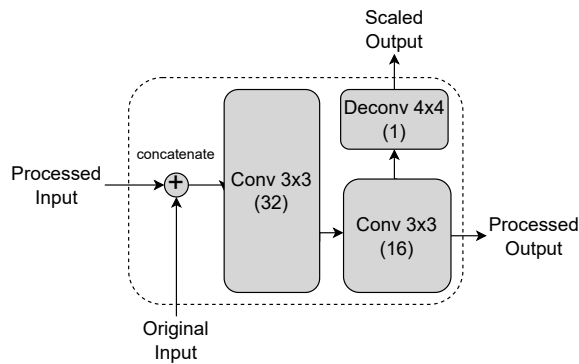
Figure 3: The scale layer used in the new network architecture. The layer takes two inputs (the previous layer output and the user input) and outputs the scaled output and 16-channel output for network connectivity.

(a) 8x8 starting noise   (b) 4x4 starting noise

Figure 4: Examples of scaled levels from different noise sizes.

each tile. For example, in Lode Runner, the enemy tile (rarest) becomes the last layer while the brick tile (most common) is the first layer. For Lode Runner (see figure 2), the network consists of six scale layers: one layer for each tile type except for empty and player tiles. We removed the player tile as we wanted the user to have control over the starting location, while the empty tile is the default tile if nothing overwrites the location.

6.1 Scale Layer

The scale layer (as shown in figure 3) consists of 2 convolution layers and a deconvolution layer. The layer takes two inputs: the original input and the processed input and produces two outputs: the scaled output and the processed output. The original input is the small-scale user-drawn level, this input is always concatenated to the processed input so the layer is conditioned by the user's choices. This is similar to the skip connections in the ResNet [7]. Both inputs are of the same size but have a different number of channels. The scaled output, instead, is twice the size of the input with one channel where this output reflects the probability of a certain tile type. The processed output is the normal output that gets pushed toward the next layer as its processed input.

For Lode Runner, we used two convolutional layers with ReLu activation function followed by a batch normalization layer for the processed output. The scaled output uses a deconvolution layer to scale the input dimension to a bigger dimension. We did not add more layers as the dataset size is not big enough and we did not want to give the network more power to easily overfit.

6.2 Network Feedforward Operation

The user-drawn level (small-scale discrete image with 7 channels) is fed to the network and we collect the different scaled outputs from the scale layers (6 different outputs). To interpret the output, we first create a level of twice the size and set it to empty tiles. We go over the scaled output from the beginning (brick layer) to the end (enemy layer). For each scaled output, if the value is greater than 0.5, we change the tile to the current layer tile, otherwise, we leave it as it is. For example, if we are checking the ladder scaled
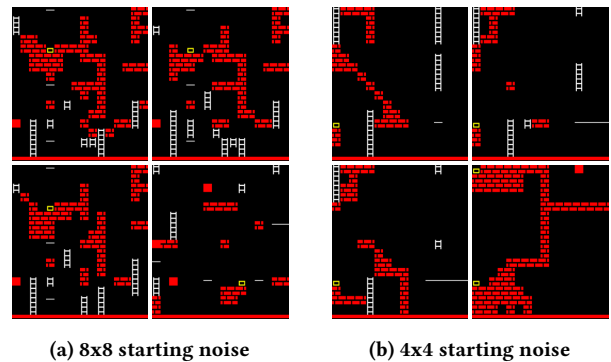
output and the value of the tile at (x=0, y=0) is more than 0.5, we change the tile in the final output at (x=0, y=0) to the ladder tile. If later at the gold scaled output that same tile location has a value of more than 0.5, the tile is changed to gold instead.

6.3 Network Training

To train this architecture, we apply a two-step training method: base training and greedy layer training. The base training adds a deconvolution layer (called training head in figure 2) after the last scale layer. The deconvolution layer produces a scaled output with 7 channels using a softmax activation function (representing all the different tile values). We use Adam optimizer to train the network using categorical cross-entropy loss function for $3,000$ epochs.

After the base training mentioned above, we run greedy layer training to fine-tune each scale layer and train their deconvolution layer. We start by removing the training head and go in order of layers from the first scale layer to the last. We get the corresponding channel from the training data and fine-tune each layer. We use Adam optimizer to fine-tune the layer using binary cross-entropy loss function for $1,000$ epochs with early stopping. After a layer is fine-tuned, we freeze its weights and move to the next layer until all layers are fine-tuned. For Lode Enhancer, we trained two different models: scale from 4x4 to 8x8 and scale from 8x8 to 16x16.

6.4 Dataset

For our experiment we used 150 Lode Runner levels made of 32x22 tiles from the Video Games Level Corpus (VGLC) [19]. To increase the dataset size, we used smaller segments of original levels. We settled down for 4x4, 8x8, and 16x16. For our first network that scales 8x8 levels to 16x16, we applied a 16x16 sliding window method with a stride of 1 which produced $17,850$ level segments in total. Since Lode Runner levels can be reflected across the x-axis, we used that method to augment our dataset to $35,700$ level segments. We then scaled down these segments to half of their size (8x8) using the nearest neighbor filter and paired them together to create the dataset. We repeated the same process using a sliding window of 8x8 to create a dataset for our second network that scales up from 4x4 to 8x8. The dataset for the second network consists of $112,500$ level segments.
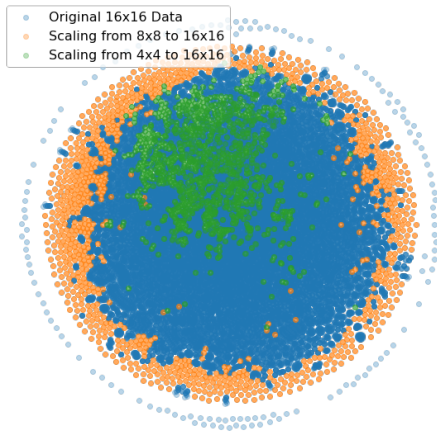
Figure 5: Rendering the training levels with different generated levels from the layer scaling network using t-SNE.

7 LAYER SCALING NETWORK ANALYSIS

In this section, we analyze quantitatively our layer scaling network. We focus on two fronts: a) investigating the effect of network scaling on the input level and b) investigating the advantages of using the proposed architecture.

7.1 Scaling Effect

In this subsection, we explore the effect of the network's scaling. We started by generating 1, 000 4x4 or 8x8 noise inputs that follow the same distribution of the training data. We passed the 8x8 inputs to 3 trained networks to scale the 8x8 to 16x16, giving us 3, 000 16x16 levels. We passed the 1, 000 4x4 inputs through 3 trained networks to scale the 4x4 to 8x8 and then passed the outputs through the other network to scale it to 16x16, giving us 3, 000 16x16 levels.

Figure 4 shows an example of the final scaled levels. It is clear that scaling twice (from 4x4 to 8x8 and then to 16x16) yields better-looking levels compared to scaling once (from 8x8 to 16x16). We believe that having the level iterate more than once in the network helps to smooth the input noise and make the levels fall in a similar distribution to the training data. This can be shown in the t-SNE visualization of all the 16x16 levels (figure 5).

7.2 Network Architecture

In this subsection, we investigate the effect of the proposed network architecture and compare it against a convolutional neural network (i.e. 2 convolutional layers followed by a deconvolutional layer). We also compare the efficiency of the greedy layer training (layer-scaling network) with respect to the training head (scaling network). For this experiment, we generated 1, 000 levels by scaling up 4x4 noise twice to reach the size of 16x16. This experiment is repeated 3 times using 6 different trained networks (i.e. 3 networks that scale from 4x4 to 8x8 and 3 networks that scale from 8x8 to 16x16) resulting to 3, 000 16x16 generated levels per experiment.

We compare the generated levels (figure 6) with the original 16x16 training levels and calculate the minimum tile-pattern KL-divergence (TPKLDiv) score [12]. To calculate the minimum score, we pair every generated level with the closest level from the training

levels (i.e. the one with the minimum TPKLDiv score), and we compute the average score and the confidence interval for all the different pairs. We find that both the convolutional network (*3.485 ± 0.084*) and the scaling network (*3.633 ± 0.041*) yield higher scores compared to the layer scaling network (*2.441 ± 0.031*). Having a lower TPKLDiv score is desired as it means that the local patterns in the generated levels follow a similar distribution to the ones from the training levels.

Figure 7 shows the expressive range analysis for the different networks. We used two main metrics: the number of reachable tiles and the number of empty tiles. The number of reachable tiles is the maximum number of reached tiles by a breadth-first search playing agent that is tested from every single location in the level. Looking at the expressive range, we notice that both convolutional and scaling networks have similar ranges. On the other hand, the layer scaling network yields a more consistent map with a lot of empty tiles and average reachable tiles. This result is unsurprising as having a higher amount of empty tiles reduces the number of reachable tiles since the main character cannot jump. This finding also suggests that many of the generated levels from the layer scaling network are similar to each other, which can be seen in figure 6d. We believe that this is an advantage to our network as it keeps the relationship between the input (figure 6a) and the output (figure 6d) as close as possible. In turn, this means that a small change in the input level will not cause drastic changes in the output level. This characteristic is particularly important for mixed-initiative tools of higher trustworthiness value, as the user can better understand the impact of their actions on the generated level.

8 QUALITATIVE EVALUATION

To evaluate Lode Enhancer and the ideas behind it, and to provide directions for future development, we ran a qualitative study with 2 professional and 1 amateur game designers. All designer participants went through the following protocol: (1) We first allowed the designers to play the game to get familiar with its mechanics. (2) We then gave them a quick tutorial about the tool and how to use it. (3) We asked designers to use the system to create a playable level. (4) We asked the designers to communicate their level design goal and then use the tool to achieve it. (5) We finally asked the designers to complete a questionnaire related to their experience using the tool.

8.1 Observing the Designers

All three designers played the first level shown in the online port of Lode Runner [2] to get familiar with the core game mechanics. After they were comfortable with the game, we run them through a quick tutorial of the tool and then asked them to create their levels (as per our experimental protocol). Throughout this process, we observed them while using the tool and recorded all their interactions with Lode Enhancer. We noticed that all three participants ended up modifying the 16x16 canvas most of the time. That was expected as the goal was to end up with a 16x16 playable level.

Figure 8 shows the 3 different playable levels created by the designers without communicating any intent. In this task, the first
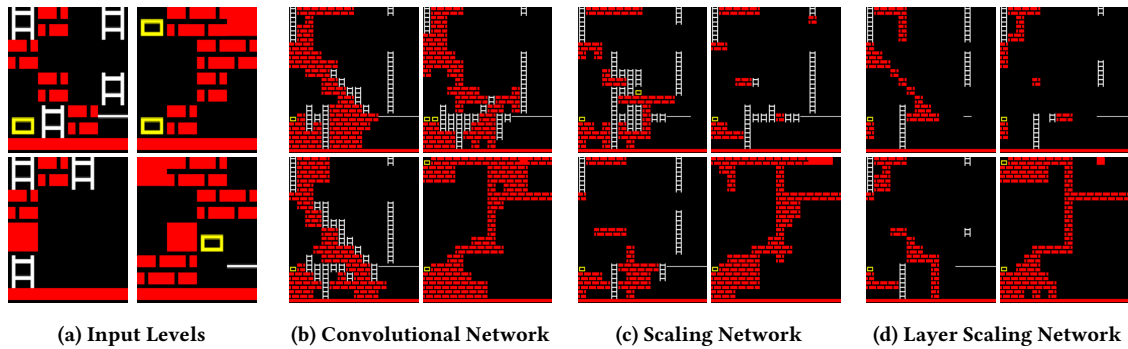
---

[2]https://loderunnerwebgame.com/game/

(a) Input Levels   **(b) Convolutional Network**   **(c) Scaling Network**   **(d) Layer Scaling Network**

Figure 6: Example of generated 16x16 levels from scaling twice the same input noise using different network architectures.

(a) Conv Network   **(b) Scaling Network**   **(c) Layer-Scaling Net**
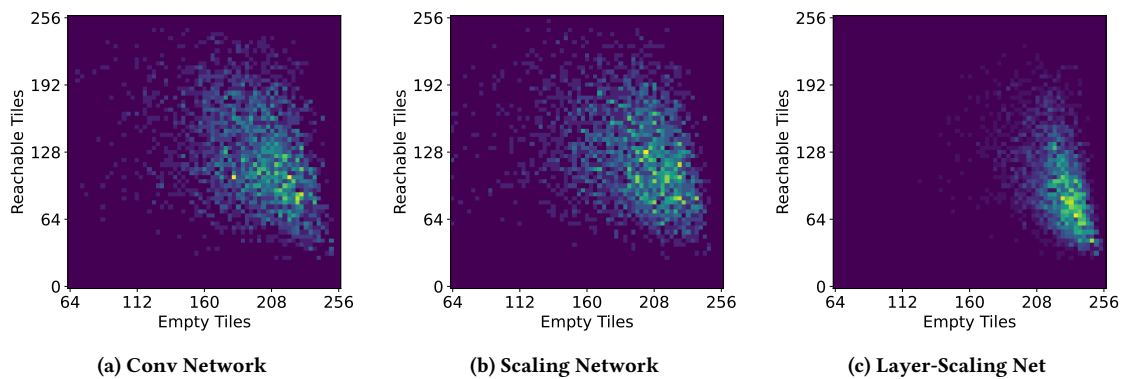
Figure 7: Expressive range analysis from scaling twice the same input noise using different network architectures.

designer (figure 8a) completely ignored the small and middle canvas and started directly manipulating the 16x16 canvas. Figure 9 shows the 3 different playable levels that were created after designers communicated a certain goal. In this task, all three designers started working with a small-scale canvas to get an interesting shape in the bigger canvas and then completed their levels by modifying the 16x16 canvas. All participants managed to create a level that they are satisfied with.

Another observation is that the third designer (figure 8c and 9c) took twice as much time to design their level compared to the others. We assume that the designer was focusing on having a highly detailed level (a lot of solid and brick tile mix) on the canvas, they need to clean some of the up-scaled structures and think about the locations of each brick tile. For all created levels except figure 9c, the designers ignored the 4x4 canvas completely. We think due to the small space of the canvas some designers might have felt restricted. Finally, we noticed that they all ignored the persistence function and did not change the default value for it but since none of them went back and forth between different-size canvases, they never needed its functionality.

8.2 Questionnaire Results

After participants completed both levels, we asked them a number of questions and analyzed their responses.

8.2.1 What did you like about the system? Designers liked how sensible the AI edits are. For example, the first designer tried to design an unreachable space but the upscaler made sure it would be reachable by adding holes and/or ladders. They also liked how it helped them start new ideas and distribute the tiles in different areas pretty fast. For example, when adding a small ladder near some platform, the system was able to connect the ladder to the platform.

8.2.2 What did you dislike about the system? Designers pointed out that the 4x4 grid was too restrictive for design and that is why they avoided using it. Going back and forth between scales was more troublesome than useful especially later in the design process as the AI was suggesting less polished sections. They also pointed out that one of the reasons they avoided the scaler is that they are not familiar with the tool so they did not completely understand the relationship between changing a small canvas and the upscaler output.

8.2.3 How could enhancing/scaling levels improve level designers' workflow and fulfilling their own design goals? For this question, all designers agreed that the system helped them save a little bit of time and created a draft version of the level pretty fast. They also think that it would be more useful if the task was to create a huge level like 128x128 while controlling a small space like 16x16.
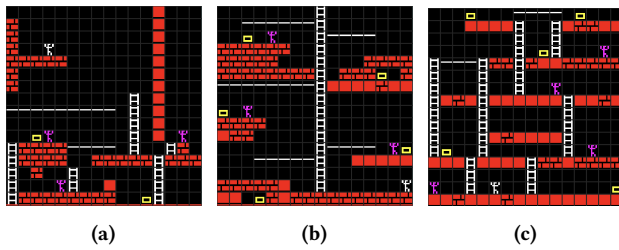
(a)   (b)   (c)

Figure 8: Three playable levels designed by 3 different participants when asked to design a playable level.
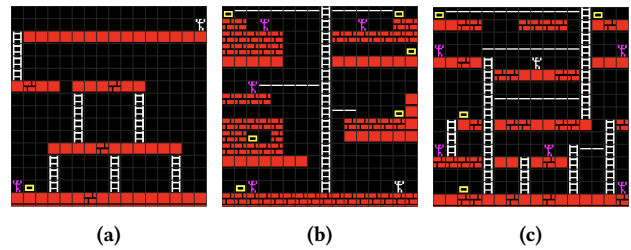

(a)   (b)   (c)

Figure 9: Three goal-oriented levels designed by 3 participants. The goals were set by the participants and communicated to us. The goals are (a) create a level with one gold that the enemy will carry and the only way to win is to trap the enemy to drop the gold, (b) create a level with at least 3 enemies and 5 golds where one of the gold is trapped, and (c) create a challenging level with multiple possible paths.

8.2.4 How did you use the persistence function? Since all designers never went back and forth between different scales, they completely ignored the persistence function. When we asked them about it, they acknowledged that they either forgot about it or did not need it.

8.2.5 What other AI/ML functionality would you like to see in a tool like this? For future functionality, each designer had an interesting insight into what might be useful. One of them wanted the AI to be able to validate the created space and figure out any problems with their design. Another designer wanted to have more control over the up-scaled canvas so there is more than one option for up-scaling. Another suggestion was to provide the user with the ability to teach the upscaler the meaning behind the small section so it can upscale it in the same meaningful way. This suggestion is mostly about making the AI able to recognize the macro-patterns behind a small design and then be able to produce it on a bigger scale, similar to studies by Baldwin et al. [1], and Dahlskog and Togelius [3].

9 DISCUSSION

Any machine-learned design assistance tool will rely on and reproduce patterns in the data it was trained on. For some domains, this might present ethical issues, but in the rather abstract domain of Lode Runner, the main concerns are to which extent it reproduces existing levels and constrains the user's creativity. Informal investigations indicate that the upscaling network does not reproduce any recognizable part of existing levels exactly. The ability to edit freely also means that the editor does not constrain the expression of users, although it can of course bias the user in various directions.

One reason that the designers did not engage much with the persistence functionality is apparently that we did not explain it well enough. This is something that needs to be improved on, given that the main reason why designers did not go back and forth between scales while editing is the worry that edits they had already done would be overwritten by the AI. This is exactly what the persistence functionality is supposed to prevent. Clearly, more work is needed on how to make this smooth and intuitive.

An important takeaway from the qualitative evaluation is that the labor-saving aspects of upscaling only really come into play for large or high-resolution levels. For 16x16 tiles, the effort saved is rather small. Lode Runner might not be the testbed for this, as most Lode Runner levels are not larger than those we use here.

For our new proposed architecture, our quantitative analysis shows promise in the ability of the network to mimic local patterns compared to traditional architectures. Having a small canvas and upscaling it multiple times allows the system to repair its own mistakes and make sure the generated levels are in distribution. This allows the system to generate bigger levels, which is the main takeaway of the qualitative evaluation. We also think that having a different output for each layer allowed the network to have different computation power for different tiles with minimum domain knowledge (i.e. the order of the tiles). Finally, we think that the skip connection allowed the system to learn a good relationship between user input and target output.

While in this study we investigated upscaling as a design assistance functionality in its own right, it is likely that it will be most efficient and useful as part of a multifaceted AI-powered toolset. One functionality that would very likely work well in tandem with the current upscaling (and downscaling) is a repair agent that makes sure that the generated level is playable. Having a repair function and a larger level canvas will likely make the tool more usable and accessible to designers. We believe that the introduced upscaling method has the potential to be a valuable addition to level editing tools similar to diffusion models for image editing tools [3].

10 CONCLUSION

We presented Lode Enhancer, a mixed-initiative level design tool based on the concept of *upscaling* as design assistance. Upscaling uses deep neural networks which are trained on artificially downscaled patches of existing Lode Runner level segments, meaning that they have learned common micro- and macro-patterns of Lode Runner game design. These patterns are then exploited when transforming the missing information of a downscaled level to a more information-rich upscaled level. We also proposed a new architecture we name *layer scaling network* and compared it to traditional architectures. We notice that the new architecture manages to capture the local patterns better than the other networks. We also noticed that the trained networks do not allow a small change in the input to cause severe (undesired or unpredictable) changes in

---

[3]https://exchange.adobe.com/apps/cc/114117da/stable-diffusion

the output level. Finally, we tested our tool through a preliminary qualitative study involving 3 game designers. All designers agreed that the tool is very helpful for creating an initial low-res draft which they can refine at a later stage in a co-creative fashion with the underlying AI. They also agreed that this tool will be even more powerful in complex games or larger-scale maps.

Lode Enhancer is the first mixed-initiative co-creativity [22] prototype that examined and built on the notion of upscaling. We argue that upscaling is a very important task during level design as it can accelerate the drafting process of large levels. As future research steps, we plan to explore how a tool like Lode Enhancer can operate in more complex games with large levels. Also, we consider adding more control on the upscaler module such that the user can have a choice between different versions of the upscaling. Finally, as the levels created usually require some minor edits from the user to make them playable the next version of Lode Enhancer will be equipped with AI agents that can autonomously repair the upscaled levels and turn them into playable ones.

REFERENCES

[1] Alexander Baldwin, Steve Dahlskog, Jose M Font, and Johan Holmberg. 2017. Mixed-initiative procedural generation of dungeons using game design patterns. In *Computational Intelligence and Games*. IEEE.

[2] Debosmita Bhaumik, Ahmed Khalifa, and Julian Togelius. 2021. Lode Encoder: AI-constrained co-creativity. In *Conference on Games*. IEEE.

[3] Steve Dahlskog and Julian Togelius. 2013. Patterns as objectives for level generation. In *Workshop on Design Patterns in Games*. ACM.

[4] Omar Delarosa, Hang Dong, Mindy Ruan, Ahmed Khalifa, and Julian Togelius. 2021. Mixed-Initiative Level Design with RL Brush. In *EvoMUSART*. Springer.

[5] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2014. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*. Springer.

[6] Matthew Guzdial, Nicholas Liao, Jonathan Chen, Shao-Yu Chen, Shukan Shah, Vishwa Shah, Joshua Reno, Gillian Smith, and Mark O. Riedl. 2019. Friend, Collaborator, Student, Manager: How Design of an AI-Driven Game Level Editor Affects Creators. In *Human Factors in Computing Systems*. ACM.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*. IEEE.

[8] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In *Computer vision and pattern recognition*. IEEE.

[9] Antonios Liapis, Gillian Smith, and Noor Shaker. 2016. Mixed-initiative content creation. In *Procedural content generation in games*. Springer.

[10] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. 2013. Sentient World: Human-Based Procedural Cartography: An Experiment in Interactive Sketching and Iterative Refining. In *EvoMUSART*. Springer.

[11] Antonios Liapis, Georgios N Yannakakis, Julian Togelius, et al. 2013. Sentient Sketchbook: Computer-aided game level authoring.. In *Foundation of Digital Games*. ACM.

[12] Simon M Lucas and Vanessa Volz. 2019. Tile pattern kl-divergence for analysing and evolving game levels. In *Genetic and Evolutionary Computation Conference*. ACM.

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Computer Vision and Pattern Recognition*. IEEE.

[14] Noor Shaker, Julian Togelius, and Mark J Nelson. 2016. Procedural content generation in games. (2016).

[15] Gillian Smith, Jim Whitehead, and Michael Mateas. 2011. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *Transactions on computational intelligence and AI in games* 3 (2011).

[16] Sam Snodgrass and Santiago Ontanón. 2016. Learning to generate video game maps using markov models. *Transactions on computational intelligence and AI in games* 9 (2016).

[17] Kirby Steckel and Jacob Schrum. 2021. Illuminating the space of beatable lode runner levels produced by various generative adversarial networks. In *GECCO*. ACM.

[18] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural content generation via machine learning (PCGML). *Transactions on Games* 10 (2018).

[19] Adam James Summerville, Sam Snodgrass, Michael Mateas, and Santiago Ontanón. 2016. The vglc: The video game level corpus. *arXiv:1606.07487* (2016).

[20] Sarjak Thakkar, Changxing Cao, Lifan Wang, Tae Jong Choi, and Julian Togelius. 2019. Autoencoder and evolutionary algorithm for level generation in lode runner. In *Conference on Games*. IEEE.

[21] TV Tropes. 2007. Enhance Button. https://tvtropes.org/pmwiki/pmwiki.php/Main/EnhanceButton. Accessed: 21-3-2023.

[22] Georgios N Yannakakis, Antonios Liapis, and Constantine Alexopoulos. 2014. Mixed-initiative co-creativity. (2014).