# Procedural Level Generation with Diffusion Models from a Single Example

**Shiqi Dai[1], Xuanyu Zhu[1], Naiqi Li[1], Tao Dai[3], Zhi Wang*[1,2,4]**

[1]Shenzhen International Graduate School, Tsinghua University
[2]Tsinghua-Berkeley Shenzhen Institute, Tsinghua University
[3]College of Computer Science and Software Engineering, Shenzhen University
[4]Peng Cheng Laboratory
{daisq21,zhuxy22}@mails.tsinghua.edu.cn, daitao@szu.edu.cn, {linaiqi,wangzhi}@sz.tsinghua.edu.cn

## Abstract

Level generation is a central focus of Procedural Content Generation (PCG), yet deep learning-based approaches are limited by scarce training data, i.e., human-designed levels. Despite being a dominant framework, Generative Adversarial Networks (GANs) exhibit a substantial quality gap between generated and human-authored levels, alongside rising training costs, particularly with increasing token complexity. In this paper, we introduce a diffusion-based generative model that learns from just one example. Our approach involves two core components: 1) an efficient yet expressive level representation, and 2) a latent denoising network with constrained receptive fields. To start with, our method utilizes token semantic labels, similar to word embeddings, to provide dense representations. This strategy not only surpasses one-hot encoding in representing larger game levels but also improves stability and accelerates convergence in latent diffusion. In addition, we adapt the denoising network architecture to confine the receptive field to localized patches of the data, aiming to facilitate single-example learning. Extensive experiments demonstrate that our model is capable of generating stylistically congruent samples of arbitrary sizes compared to manually designed levels. It suits a wide range of level structures with fewer artifacts than GAN-based approaches. The source code is available at https://github.com/shiqi-dai/diffusioncraft.

## Introduction

3D content creation tools form the basis of many emerging metaverse applications. Despite this, current approaches often rely heavily on manual labor, resulting in high costs and inefficiencies. Gaming is considered as the medium that most closely resembles the concept of metaverse. A downstream task in game study, i.e., level generation, utilizes algorithms to generate reasonable game levels automatically. Addressing this task can effectively assist in the efficient and automatic construction of a large number of 3D scenes for game-based metaverse applications.

The game levels studied in this article can be simplified as 2D/3D arrays of tokens, where each token represents a type of object or part of an object. For example, in Minecraft, a tree is composed of "oak wood" and "leaf" tokens. In token-based level generation that has similar meanings to pixels, traditional algorithms are based on search, grammar, or rules (Liu et al. 2021). Recently, approaches based on deep learning, which can automatically learn complex patterns, have made significant breakthroughs. GAN and its variants are currently popular models (Awiszus, Schubert, and Rosenhahn 2020, 2021; Park et al. 2019; Volz et al. 2018). However, deep learning-based methods face two key challenges in level generation: data scarcity and content generation incoherence. Most game levels are designed manually, and it is usually difficult to obtain a large dataset, which is not friendly to data-sensitive models. In order to learn from small data, one of the approaches is to design generative models to learn from a single level. The cutting-edge work is to adopt pyramid-shaped GANs with multi-scale training (Awiszus, Schubert, and Rosenhahn 2020, 2021). The training level is downsampled into different resolutions and groups of GANs learn the internal distribution in a coarse-to-fine manner, starting from the lowest resolution scale. Although this network structure addresses the demand for data-intensive generative models, errors accumulate during generation in multi scales (Wang et al. 2022), leading to numerous low-quality samples, i.e., content incoherence such as unclear shapes and meaningless floating blocks. This kind

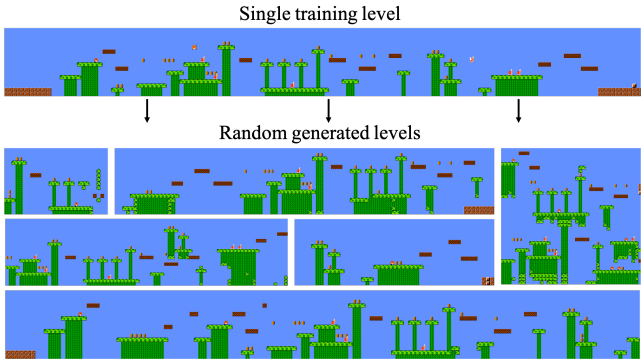


Figure 1: Random generated levels from one Super Mario Bros level. Our model is trained to capture the internal distribution of a single level and can subsequently generate levels of arbitrary sizes.

*Corresponding author.

of incoherence and discontinuities affect the playability and aesthetics of level. In order to find samples that satisfy specified requirements, evolutionary search is used to explore the latent space of a GAN (Volz et al. 2018; Edwards, Jiang, and Togelius 2021; Fontaine et al. 2021). Previous works have demonstrated that GAN-based generative models have a vast latent space, requiring conditional constraints or search algorithms to improve the quality of generated levels.

In this paper, we propose an unconditional procedural level generation approach training a latent diffusion model on a single level. There are two major challenges in this task. One is to decide on which space to perform the diffusion process. Recent works have found that training the diffusion model in such a continuous token space enables faster and more stable convergence. The original level representation varies in discrete space. By utilizing the dense representation method *word2vec* from Natural Language Processing (NLP), we map the one-hot encoding training level onto a high-dimensional embedding space using the semantic labels of tokens. The other one is to enable diffusion models to learn from a single level. Our model is based on DDPM, which successfully leverages diffusion models in image synthesis. We restrict the receptive field of the denoising network and propose a new training algorithm to support training on one level. Compared with GAN-based models, our model is only trained on a single scale to capture the internal distribution of the training level.

Experiments show that our model can generate a large amount of available token-based levels with fewer artifacts than GAN-based results. Our model enables its users to generate levels based on a single reference example, which provides insights and guidance for users in their creation process. The main contributions of this work are summarized as follows:

- We propose a diffusion model trained on a single token-based level. Our model demonstrates the ability to generate diverse, playable, and high-quality samples that locally resemble the training example.
- We draw a connection between level generation using semantic tokens and text generation within the embedding space. It is proved that dense token representation and training loss inspired by text generation improve the percentage of available generated results.
- We introduce two keys to limit the receptive fields of our diffusion model: fully convolutional networks with residual connections and training using large crops, aiming to capture the internal distribution of the game level.
- Experimental results show that our model outperforms the baseline approaches and achieves excellent results in terms of diversity and quality for the generated levels.

## Related Work

Our work is related to two fields: procedural content generation and single example generation. In this section, we will give a brief overview of the main achievements in these areas and explain how they are connected to our method.

### Procedural Content Generation

Procedural Content Generation (PCG) involves the automated or algorithmic creation of game assets like rules, dialog, models and game mechanics with minimal human input (Shaker, Togelius, and Nelson 2016). Traditional PCG techniques encompass search-based, rule-based, grammar-based and solver-based methods. Our work is situated within an emerging research domain, procedural content generation via machine learning (PCGML), which aims to train complex neural models by learning from preexisting human-authored content (Summerville et al. 2018). (Liu et al. 2021) provided an overview of the PCGML field, with a specific emphasis on deep learning.

Studies in both industry and academia have focused a lot on generating playable game levels. Typical model choices for generating game levels comprise variational autoencoders (Snodgrass and Sarkar 2020), generative adversarial networks (GANs) (Park et al. 2019; Torrado et al. 2020; Volz et al. 2018; Giacomello, Lanzi, and Loiacono 2018), and reinforcement learning (Khalifa et al. 2020). Among them, GANs are popular for generating content represented by pixel-based images or 2D/3D array of tiles, such as Super Mario Bros (Volz et al. 2018) and *Doom* (Giacomello, Lanzi, and Loiacono 2018). To acquire target levels that vary across a set of specified gameplay measures, evolutionary computation such as Quality Diversity Algorithms (QDAs) is adopted to explore the latent space of a GAN (Fontaine et al. 2021).

Our work is focused on token-based level generation (Salge et al. 2018), with a particular emphasis on utilizing Minecraft and Super Mario Bros as representative examples from the realms of 2D and 3D games, respectively. Compared with other 3D video games, Minecraft has plenty of tokens. Each token contains a type of object or part of it, such as grass, wood, and air, represented by a semantic label. (Awiszus, Schubert, and Rosenhahn 2021) proposed WorldGAN, an architecture similar to SinGAN that learns from a single level in an unsupervised way. (Sudhakaran et al. 2021) adopted Neural Cellular Automata (NCA) (Mordvintsev et al. 2020) in reconstructing Minecraft artefacts. (Merino, Charity, and Togelius 2023) addressed an application of interactive evolution with latent variable evolution to procedurally generate target Minecraft structures.

### Learning a Generative Model from a Single Example

Exploring generative models that learn from a single example has recently attracted increasing attention from researchers. This task, as the opposite of training on large data, is suitable for the domains which have difficulty in collecting large datasets. A mainstream method is to divide the single example into several parts and design networks to learn the patch distribution. SinGAN (Shaham, Dekel, and Michaeli 2019) introduced a multi-scale pyramid architecture which train a GAN at each scale after downsampling the training image. Several works based on SinGAN have been applied in other domains. (Wu and Zheng 2022) proposed to generate 3D shape with tri-plane hybrid representation (Wu and

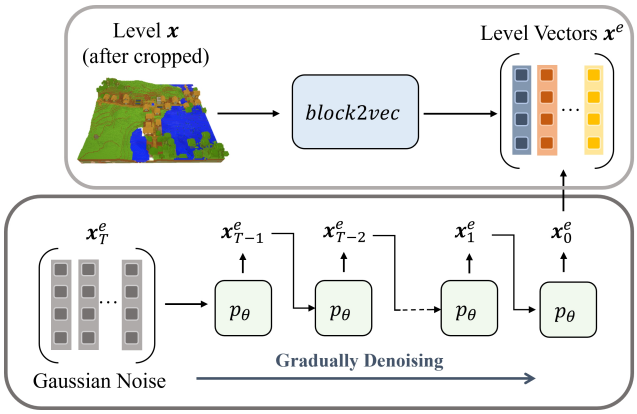


Figure 2: Method overview. Given a piece of game level as input $x$ , we initially train *block2vec* token embeddings to map the cropped level to an implicit latent representation, denoted as $x^e$. Then we train a denoising network $p_\theta$ on it to learn the distribution of latent features. At inference time, we sample a new latent feature using the diffusion model and then map the resulting embeddings to the nearest tokens.

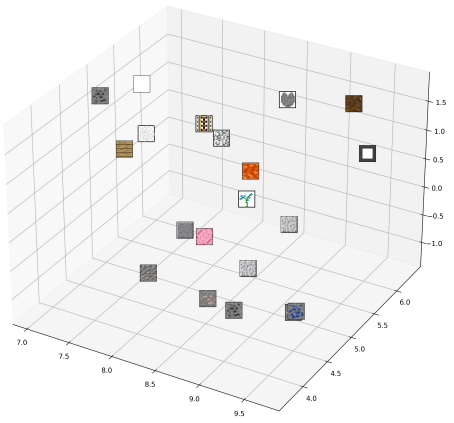

Figure 3: Visualization of the embeddings learned by *block2vec* in the example "vanilla_mineshaft". Each block is represented by its corresponding texture map. These 32-dimensional embeddings are transformed to three dimensions using the Uniform Manifold Approximation and Projection (McInnes, Healy, and Melville 2018).

Zheng 2022). Drop the GAN (Granot et al. 2022) replaced the generator with conventional algorithm module and performs better than convolutional based generator. Previous GAN-Based methods (Awiszus, Schubert, and Rosenhahn 2020, 2021) introduced a downsampling algorithm of tokens and generated available samples both in 2D and 3D level generation. (Siper, Khalifa, and Togelius 2022) viewed level generation as repair and iteratively generated levels from a random starting example.

Unlike GANs, previous works about training diffusion models on a single example only improved one scale training, and they can achieve the same effect as GANs. SinDiffusion (Wang et al. 2022) limited the receptive field of denosing work by lessening the downsample and unsample operation and other components. SinFusion (Nikankin, Haim, and Irani 2022) gave up all dowansample and upsample layers and replaced it with a fully convolutional chain of ConvNext blocks with residual connections. Both of the two works lighten the original denosing network and prevents the model from training global features to cause pattern collapse or overfitting.

## Methodology

In this section, we propose a denoising diffusion probabilistic model (DDPM) that can capture the internal patch distribution of a single token-based game level. The overview of our method is in Fig. 2. The key point is that the receptive field of the denoising network should be designed to be small enough to learn patch-level features, analogously to the use of patch discriminators in GAN-based approaches. With that, the trained diffusion model is able to produce patch-level variations while preserving the global structure.

Directly training a diffusion model on a game level with sparse representation is computationally demanding and hard for the model to converge. To address this issue, we first represent the level as word embedding. Specifically, we make use of the semantic label of each token and encode it into an implicit latent embedding. Given the encoded level of embeddings, we train a latent diffusion model on it with modification on network and training procedure. Based on these designs, the new piece of level generated by our model can be sampled with random noise.

### Level Representation

In previous encoding schemes for levels, the common practice was to employ one-hot encoding to represent discrete tokens within the token space, such as in games like Doom and Mario. However, when faced with games that possess an extensive array of tokens and undergo continuous updates, such as in the case of Minecraft, the use of one-hot encoding leads to an escalating burden on generative models training and GPU memory. To address this issue, a potential solution involves utilizing dense fixed-size vectors to represent each token. This bears resemblance to the task of training world embeddings in the realm of Natural Language Processing (NLP).

For a piece of 2D token-based level, denoted as $L$, it comprises discrete tokens denoted as $t \in T$, resulting in a layout $L = [t_{ij}]_{m \times n}$. Here, $m$ and $n$ signify the count of tokens in the respective dimensions, and $T$ represents the set of all token types within the considered fragment. Each token $t$ is associated with a semantic label comprising one to three words (e.g., "Coin Question Block"). The embedding function $\text{EMB}(t_{ij})$ serves to map each token to a vector within $\mathbb{R}^d$. Consequently, the embedding of a level fragment $L$, measuring $m \times n$ in size, is defined as $\text{EMB}(L) = [\text{EMB}(t_{ij})] \in \mathbb{R}^{mnd}$.

Classic methods for training word embeddings encompass *Word2Vec* (Mikolov et al. 2013), *GloVe* (Pennington, Socher, and Manning 2014), as well as techniques based on pre-trained Language Models (such as BERT (Devlin et al.

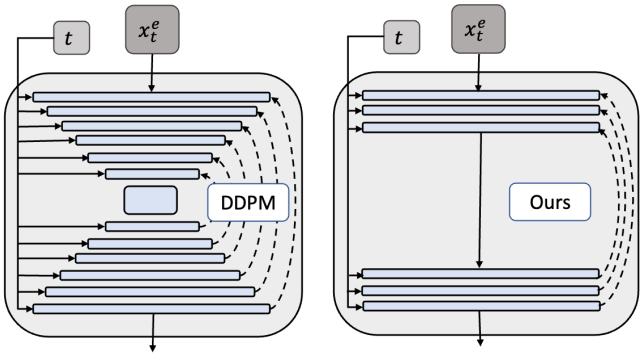


Figure 4: Denoising network architecture. Notably we have removed all upsampling and downsampling operations as well as the intermediate attention layers. This simplification is to confine the network's receptive field to local features of the data, while retaining fully connected residual blocks.

2018), GPT (Radford et al. 2018), CLIP (Radford et al. 2021)), among others. Notably, owing to the considerably smaller number of token types, i.e., $|T|$, within our experimental token-based levels as compared to the expansive vocabulary of general NLP, the task's specific nature greatly reduces the token variety to roughly 100 or fewer, even in the case of a game like Minecraft, which might encompass thousands of tokens. Consequently, the necessity to jointly train generative models and the embedding function EMB, as observed in NLP text generation, is obviated. Instead, training directly on the set $T$ is sufficient. Likewise, this implies that there is no need to derive token representations from a Language Model, since while their embeddings are semantically rich, they are overly complex. A dimension of $d = 32$ suffices for our task, as evidenced by WorldGAN (Awiszus, Schubert, and Rosenhahn 2021).

Empirical results indicate that a simple two-layer skip-gram model is suitable for all existing procedural level generation datasets in Fig. 3. This approach, initially introduced by WorldGAN (Awiszus, Schubert, and Rosenhahn 2021) to overcome the challenges posed by the uncertain channel length (length of one-hot encoding) in Minecraft levels during GAN training, has been expanded to encompass all 2D and 3D token-based level games. This extension attests to the versatility of this representation method, which adeptly transforms one-hot encoded tokens into vectors infused with greater semantic information. Continuous data structures are better suited as inputs for latent diffusion models, and they are more conducive to training.

## Denoising Network Architecture

A straightforward approach for the denoising network involves utilizing the original U-Net structure from DDPM, treating the level embeddings as latent features akin to images. Nonetheless, this methodology is susceptible to "overfitting", whereby the model tends to solely produce level embeddings identical to the input, with minimal variance. To circumvent this limitation, we engineer our denoising network with a confined receptive field, thereby averting the "overfitting" issue.

Fig. 4 delineates the architectural blueprint of our denoising network. To deliberately constrain the receptive field, we excise all necessitated upsampling and downsampling operations, thereby ensuring that the data flow during training remains devoid of spatial dimension reduction across the strata. The pre-existing U-Net structure transmutes into an entirely convolutional sequence of residual blocks, interconnected by residual connections. The count of these resblocks, functioning as a hyperparameter, substantively impacts the diversity of the output from the diffusion model.

## Training Details

**Large Random Crops** In our empirical exploration, we observed that, when confronted with diminutive level dimensions such as $40\times40$ or $16\times200$, the model's capacity to discern the global structure, rather than fixate solely on local facets, remains compromised even with a shallow network depth. To address this, we opt to subject the training data to a process of random cropping, encompassing substantial dimensions. This maneuver ensures that the model acquires the capacity to glean insights from patch-level distribution. It is noteworthy that we attempted data augmentation techniques, such as rotations and flips, yet their impact remained unremarkable, occasionally instigating mode collapse—an outcome characterized by excessive noise within the output. The cropping ratio, positioned as a hyperparameter, thus orchestrates a harmonious balance between the visual quality and diversity of the generated output. The ramifications of cropping size on the diversity of generated quality are delineated within the ablation experiments section.

**Loss** In the course of each iterative training step, our denoising network is trained to anticipate and mitigate noise perturbations. Nevertheless, the stability and convergence of this network within the context of procedural level generation exhibit a degree of fragility. In response, we undertake a strategic adjustment by revising the loss function to target the prediction of the initial state variable, denoted as $x_0$. This strategic adaptation engenders two notable benefits: it accelerates the training process and engenders improved outputs with regard to visual fidelity.

The loss function of our model is as follows:

$$L(\theta) = \mathbb{E}_{\mathbf{x}_0^e,\epsilon}\left[\left\|x_0^e - \tilde{x}_{0,\theta}^e\left(x_t^e, t\right)\right\|^2\right]. \tag{1}$$

Conjointly, our training procedure is shown in Alg. 1.

# Experiments

## Setup

**Datasets** To evaluate the effectiveness and generality of our method, we choose two games that are widely used for procedural content generation: Minecraft and Super Mario Bros (SMB). For Minecraft level generation, the dataset comes from a large handcrafted Minecraft world called DREHMAL:PRIMΩRDIAL (Awiszus, Schubert, and Rosenhahn 2021). The training data includes levels with sizes of $40 \times 40 \times 40$ and $100 \times 40 \times 100$, covering six

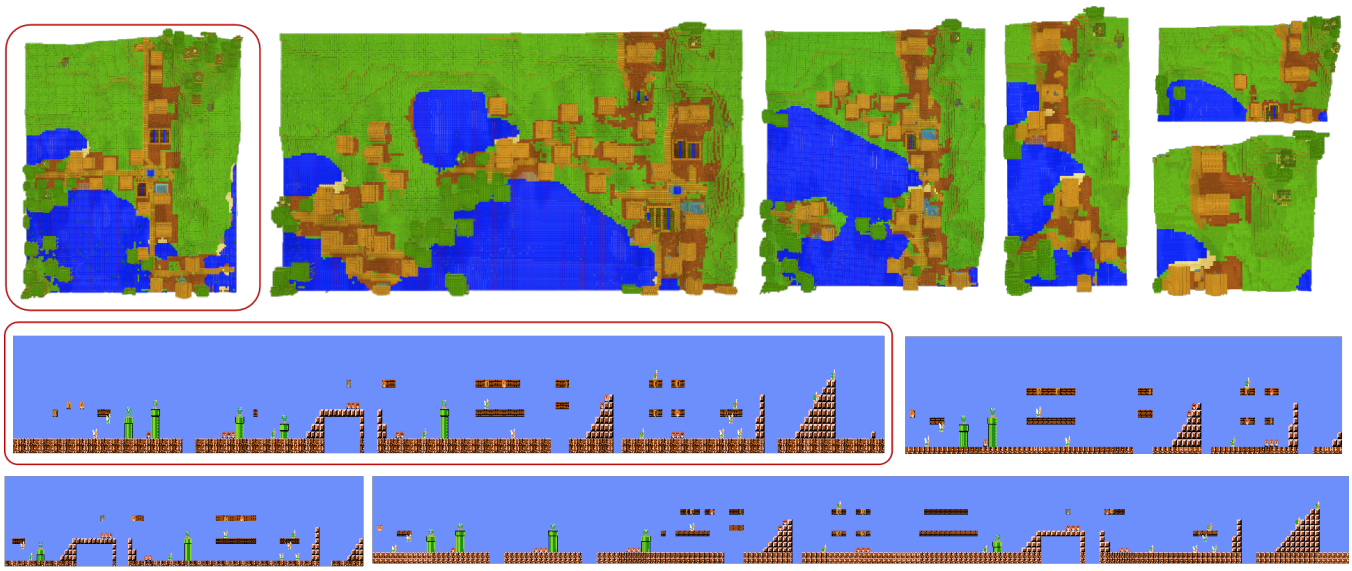

Figure 5: Random samples from a single Minecraft (up) or a single SMB (down) level. The training examples are labeled by red boxes. The size and aspect ratios of the generated results can be adjusted by modifying the spatial dimensions of the sampled noise $x_T^e$.

Algorithm 1: Training on a single level $x$

$x^e \leftarrow \text{EMB}(x)$
**repeat**
  $x_0^e \leftarrow Crop(x^e)$
  $t \sim \text{Uniform}(1, \ldots, T = 50)$
  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  Take gradient descent step on:
  $\nabla_\theta \left\| x_0^e - \widetilde{x}_{0,\theta}^e \left( \sqrt{\overline{\alpha}_t} x_0^e + \sqrt{1-\overline{\alpha}_t}\epsilon, t \right) \right\|^2$
**until** converged

biomes. The SMB level dataset comes from VGLC (Summerville et al. 2016), which is a corpus of video game levels in easily parseable formats. We train our model on each piece of level and evaluate the fidelity and diversity of the generated results.

**Evaluation Metric** To quantitatively evaluate the quality and diversity for generated levels, we adopt the following metrics.

1) Tile Pattern KL-Divergence (Lucas and Volz 2019) : To measure the diversity between the generated content, we used Tile Pattern KL-Divergence (TPKLDiv) to measure the structural similarity between the 20 generated levels. The formula is as follows:

$$D_{KL}(P\|Q) = \sum_{s \in S_P} P'(s) \log \left( \frac{P'(s)}{Q'(s)} \right). \quad (2)$$

$P'(s)$ refers the frequency of token $s$ in the original level and $Q'(s)$ refers its frequency in the generated level. $s \in S_P$ refers the collection of tokens. We use three tile pattern windows ($3 \times 3 \times 3$, $5 \times 5 \times 5$, $10 \times 10 \times 10$) to calculate the TPKL-Div. We measure two kinds of TPKL-Div between the real and learned level distribution, and sum it up with weight $w = 0.5$.

2) Levenshtein Distances (Levenshtein et al. 1966) : In order to assess the diversity of generated samples, we adopt the Levenshtein distance as a measure to quantify the differences among results. The Levenshtein distance, also known as the edit distance, is employed as a metric to quantify the similarity between two strings. It denotes the minimal count of editing operations, including insertions, deletions, and substitutions, required to transform one string into another. Specifically, we transform the level into a linear sequence and replace each token with a number, so that we get a stringified level to calculate distance.

**Implementation Details** The whole framework is implemented by PyTorch. Experiments are performed on NVIDIA Tesla V100. Each level is trained on one GPU with 0.9999 EMA decay. The activation function in network is LeakyReLU. The latent diffusion model has a max time step $T = 1000$. We train it for 50000 iterations with an initial learning rate $5 \times 10^{-4}$ and a batch size of 8.

## Qualitative Evaluation

Fig. 5 showcases the qualitative outcomes of random-generated levels from both Minecraft and SMB. By adjusting the dimensions of the sampled Gaussian noise, we can resize the input to different sizes and shapes, while retaining its essential local characteristics. As can be seen, the style of our generated samples matches that of the level they were trained on, concurrently affording rational local variations in terms of geometry.

In Super Mario Bros (SMB) level generation, we provide more metrics to evaluate our model performance shown

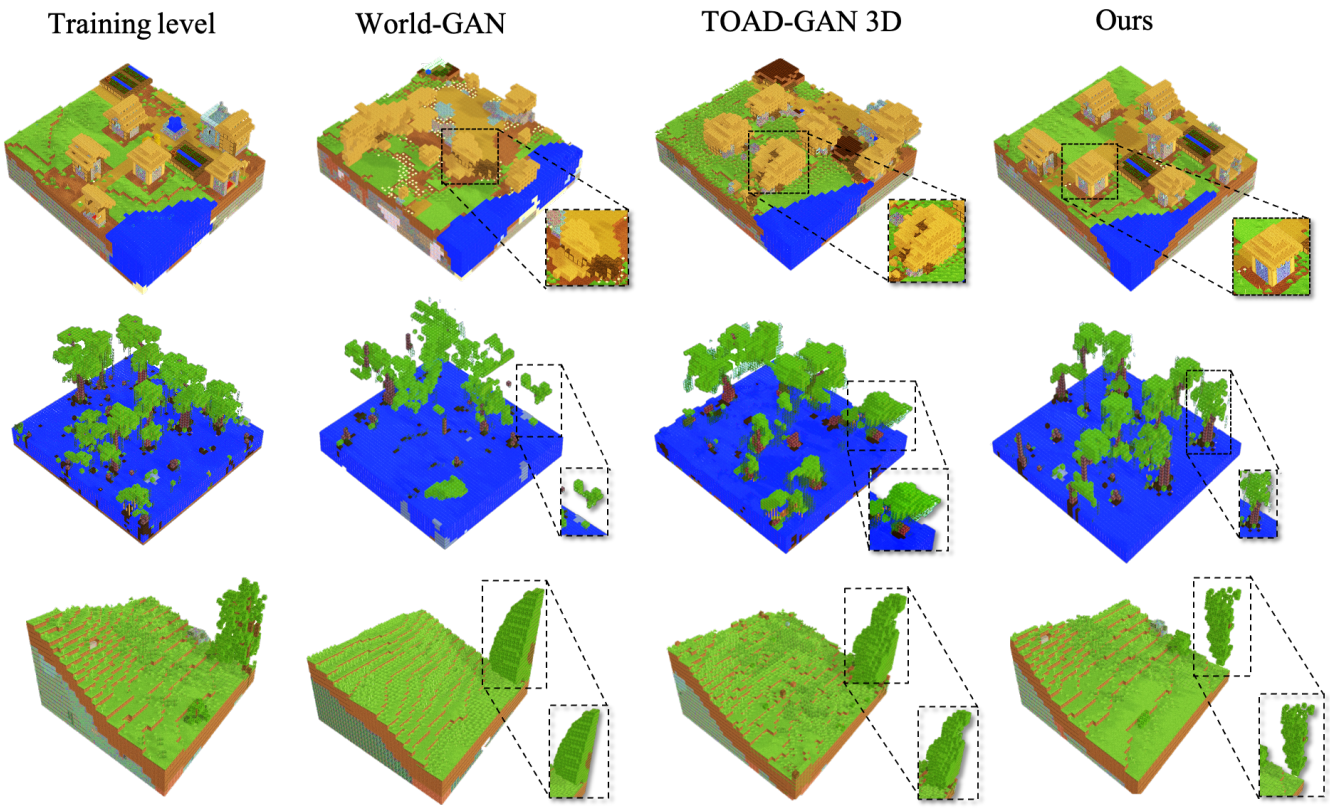


Figure 6: Visual comparison. We compare the generated Minecraft levels from our model with those produced by GAN-based baselines. Notably, our model is capable of generating semantic structures, as evidenced by the detailed house features in the first row. Furthermore, our generated results exhibit fewer artifacts compared to the floating blocks present in the GAN-based results shown in the second row.

in Table 2. To measure playability, we compute the average percentage of level completed by A* agent over 100 generated levels. Also, we perform a subjective visual test to evaluate the generation quality. We randomly select 100 generated levels for each model and invite three volunteers to score each subset based on their aesthetics. Their average scores are counted as user preference. Table 2 indicates that our diffusion-like method performers better than GAN-based baseline.

## Comparison with Previous Methods

We compare our method against World-GAN and TOAD-GAN with Minecraft results. They are GAN-based single level generative model. Since the original TOAD-GAN takes 2D SMB level as input, we have adapted the model architecture to accommodate inputs from 3D Minecraft levels, marked as TOAD-GAN 3D.

The quantitative results are shown in Table 1. Compared to World-GAN and TOAD-GAN 3D, our model performs best in TPKL-Div, which means the distribution of our generated levels is closer to the real level than GAN-based methods. As for average Levenshtein distance between the generated levels, the diversity of our generated levels is in an order of magnitude compared to the baselines.

In addition to the quantitative results, the visual differences among the examples of "village", "swamp", and "plains" are presented in Fig. 6. In particular, levels generated by World-GAN and TOAD-GAN 3D have unknown semantic structures like floating blocks. This is largely due to the fact that GAN-based approaches train on sparse representations and accumulate errors during pyramid pipeline (Wang et al. 2022). Instead, our approach provides more details and fine generation to make the output more reasonable, as evidenced by the house shape and tree shape highlighted in Fig. 6.

## Ablation Study

We conduct ablation studies to validate several design choices of our method. We measure the sample quality using the TPKL-Div Levenshtein distance on 20 samples. Specifically, we compare our proposed method with the following variants:

**W/O Large Random Crops** When learning from small-scale samples ($40 \times 40 \times 40$), we employ the generator (with the depth of 1) without applying random cropping to the input. Additionally, we examine the impact of cropping ratios on the generated samples. Table 3 demonstrates that omit-

| Metrics | Methods | Examples | | | | | |
|---|---|---|---|---|---|---|---|
| | | Village | Swamp | Desert | Plains | Beach | Mineshaft |
| TPKL-Div ↓ | TOAD-GAN 3D | 8.09 | 11.89 | 11.96 | 6.77 | 14.73 | 15.82 |
| | World-GAN | 8.04 | 11.19 | 28.68 | 13.68 | 14.75 | 15.76 |
| | Ours | **5.60** | **9.43** | **11.65** | **4.92** | **14.28** | **12.79** |
| Levenshtein ↑ | TOAD-GAN 3D | 8479.68 | **15579.00** | 1314.69 | 5066.27 | 14221.77 | 9653.01 |
| | World-GAN | **10831.22** | 7653.78 | 2657.54 | **5235.67** | **15903.74** | **9960.29** |
| | Ours | 10324.40 | 14431.89 | **5087.05** | 4148.61 | 15101.45 | 9324.93 |

Table 1: Quantitative evaluation. ↑: a higher metric value is better; ↓: a lower metric value is better. We compare the visual quality with TPKL-Div and results diversity with Levenshtein distance. Bold scores are the best results of the three algorithms.

| Level No. | Model | %Completed | % User Preference |
|---|---|---|---|
| 1-2 | TOAD-GAN | 86.5 | 66.92 |
| 1-2 | Ours | **87.8** | **73.33** |
| 4-2 | TOAD-GAN | 68.8 | 66.05 |
| 4-2 | Ours | **70.2** | **74.43** |

Table 2: SMB level playability and human preference score.

| %Crop Size | TPKL-Div↓ | Levenshtein↑ |
|---|---|---|
| 90 | 5.34 | 7881.32 |
| 80 | 5.60 | 10324.40 |
| 70 | 5.35 | 12782.03 |
| 60 | 5.62 | 13007.54 |
| 50 | 6.03 | 13382.25 |

Table 3: Ablation study on Crop Size. ↑: a higher metric value is better; ↓: a lower metric value is better. It can be observed that the generated outputs achieve enhanced visual quality and diversity when a suitable crop size is employed.

ting random cropping from the training data leads to "overfitting", where the model generates levels that are exactly the same as the original ones. Furthermore, the cropping ratio affects the balance between level diversity and the quality of the generated outputs.

**Objective Parametrization** We suggest having the diffusion model predict $x_0$ directly. In this case, we compare this strategy with the traditional parameterization used in image generation, where a denoising network is used to predict the noise $\epsilon$. Table 4 demonstrates that parametrizing by $x_0$ consistently yields good performance across different dimensions. In single example case, $x_0$ is fixed and therefore easier to predict. Predicting the noise $\epsilon$, makes the diffusion model hard to converge.

| Prediction | TPKL-Div↓ | Levenshtein↑ |
|---|---|---|
| $\epsilon$ | 19.08 | 50492.19 |
| $x_0$ | 7.49 | 94528.69 |

Table 4: Ablation study on Object Parametrization. ↑: a higher metric value is better; ↓: a lower metric value is better. Predicting $x_0$ is better in single example case.

## Conclusions and Future Work

In this work, we introduce an unconditional procedural level generation approach that trains a latent diffusion model on a single level. To extract deeper semantic meaning from level tokens and apply them in diffusion models, we employ the word embedding algorithm called *word2vec* and extend it to *block2vec*. Subsequently, in order to train a diffusion model using a single training data, we impose restrictions on the receptive field of the denoising network. This ensures that the model can effectively learn the internal distribution of latent features. Experimental results demonstrate that our model is capable of generating samples that closely resemble those designed by humans, as seen in existing methods.

Nevertheless, our work does have several limitations. Firstly, it requires more time and computational resources in both training and inference compared to GANs (Kulikov et al. 2023). Additionally, achieving conditional generation that aligns with designer's requirements is an area for further exploration. Future endeavors could involve incorporating user guidance or prompts to enhance the generative process.

## Acknowledgments

This work was supported in part by the National Key Research and Development Project of China (Grant No. 2023YFF0905502), National Natural Science Foundation of China under Grant 62302309, and Shenzhen Science and Technology Program (Grant No. RCYX20200714114523079 and JCYJ20220818101014030). Also, we would like to thank Kuaishou for sponsoring the research.

## References

Awiszus, M.; Schubert, F.; and Rosenhahn, B. 2020. Toadgan: coherent style level generation from a single example. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, 10–16.

Awiszus, M.; Schubert, F.; and Rosenhahn, B. 2021. WorldGAN: a Generative Model for Minecraft Worlds. In *2021 IEEE Conference on Games (CoG)*, 1–8.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Edwards, M.; Jiang, M.; and Togelius, J. 2021. Search-based exploration and diagnosis of TOAD-GAN. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 17, 140–147.

Fontaine, M. C.; Liu, R.; Khalifa, A.; Modi, J.; Togelius, J.; Hoover, A. K.; and Nikolaidis, S. 2021. Illuminating mario scenes in the latent space of a generative adversarial network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 5922–5930.

Giacomello, E.; Lanzi, P. L.; and Loiacono, D. 2018. Doom level generation using generative adversarial networks. In *2018 IEEE Games, Entertainment, Media Conference (GEM)*, 316–323. IEEE.

Granot, N.; Feinstein, B.; Shocher, A.; Bagon, S.; and Irani, M. 2022. Drop the gan: In defense of patches nearest neighbors as single image generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13460–13469.

Khalifa, A.; Bontrager, P.; Earle, S.; and Togelius, J. 2020. Pcgrl: Procedural content generation via reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, 95–101.

Kulikov, V.; Yadin, S.; Kleiner, M.; and Michaeli, T. 2023. Sinddm: A single image denoising diffusion model. In *International Conference on Machine Learning*, 17920–17930. PMLR.

Levenshtein, V. I.; et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, 707–710. Soviet Union.

Liu, J.; Snodgrass, S.; Khalifa, A.; Risi, S.; Yannakakis, G. N.; and Togelius, J. 2021. Deep learning for procedural content generation. *Neural Computing and Applications*, 33(1): 19–37.

Lucas, S. M.; and Volz, V. 2019. Tile pattern kl-divergence for analysing and evolving game levels. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 170–178.

McInnes, L.; Healy, J.; and Melville, J. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Merino, T.; Charity, M.; and Togelius, J. 2023. Interactive Latent Variable Evolution for the Generation of Minecraft Structures. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–8.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mordvintsev, A.; Randazzo, E.; Niklasson, E.; and Levin, M. 2020. Growing neural cellular automata. *Distill*, 5(2): e23.

Nikankin, Y.; Haim, N.; and Irani, M. 2022. SinFusion: Training Diffusion Models on a Single Image or Video. *arXiv preprint arXiv:2211.11743*.

Park, K.; Mott, B. W.; Min, W.; Boyer, K. E.; Wiebe, E. N.; and Lester, J. C. 2019. Generating educational game levels with multistep deep convolutional generative adversarial networks. In *2019 IEEE Conference on Games (CoG)*, 1–8. IEEE.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training.

Salge, C.; Green, M. C.; Canaan, R.; and Togelius, J. 2018. Generative design in minecraft (gdmc) settlement generation competition. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, 1–10.

Shaham, T. R.; Dekel, T.; and Michaeli, T. 2019. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4570–4580.

Shaker, N.; Togelius, J.; and Nelson, M. J. 2016. Procedural content generation in games.

Siper, M.; Khalifa, A.; and Togelius, J. 2022. Path of destruction: Learning an iterative level generator using a small dataset. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, 337–343. IEEE.

Snodgrass, S.; and Sarkar, A. 2020. Multi-domain level generation and blending with sketches via example-driven bsp and variational autoencoders. In *Proceedings of the 15th international conference on the foundations of digital games*, 1–11.

Sudhakaran, S.; Grbic, D.; Li, S.; Katona, A.; Najarro, E.; Glanois, C.; and Risi, S. 2021. Growing 3d artefacts and functional machines with neural cellular automata. *arXiv preprint arXiv:2103.08737*.

Summerville, A.; Snodgrass, S.; Guzdial, M.; Holmgård, C.; Hoover, A. K.; Isaksen, A.; Nealen, A.; and Togelius, J. 2018. Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games*, 10(3): 257–270.

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

Torrado, R. R.; Khalifa, A.; Green, M. C.; Justesen, N.; Risi, S.; and Togelius, J. 2020. Bootstrapping conditional gans for video game level generation. In *2020 IEEE Conference on Games (CoG)*, 41–48. IEEE.

Volz, V.; Schrum, J.; Liu, J.; Lucas, S. M.; 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*, 221–228.

Wang, W.; Bao, J.; Zhou, W.; Chen, D.; Chen, D.; Yuan, L.; and Li, H. 2022. SinDiffusion: Learning a Diffusion Model from a Single Natural Image. *arXiv preprint arXiv:2211.12445*.

Wu, R.; and Zheng, C. 2022. Learning to Generate 3D Shapes from a Single Example. *ACM Transactions on Graphics (TOG)*, 41(6): 1–19.