UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Tese

Aprimoramento da Geração de Mapas em Jogos Digitais por Meio de Estratégias Avançadas em GANs

Daniele Fernandes e Silva

Pelotas, 2025

Daniele Fernandes e Silva

Aprimoramento da Geração de Mapas em Jogos Digitais por Meio de Estratégias Avançadas em GANs

Tese apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Marilton Sanchotene de Aguiar
Coorientador: Prof. Dr. Rafael Piccin Torchelsen

Pelotas, 2025

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação da Publicação

S586a Silva, Daniele Fernandes e

Aprimoramento da geração de mapas em jogos digitais por meio de estratégias avançadas em GANs [recurso eletrônico] / Daniele Fernandes e Silva ; Marilton Sanchotene de Aguiar, orientador ; Rafael Piccin Torchelsen, coorientador. — Pelotas, 2025.
89 f. : il.

Tese (Doutorado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2025.

1. Geração procedural de conteúdo. 2. Geração de mapa. 3. Mapa procedural. 4. Aprendizado profundo. 5. Redes adversárias generativas. I. Aguiar, Marilton Sanchotene de, orient. II. Torchelsen, Rafael Piccin, coorient. III. Título.

CDD 005

Elaborada por Simone Godinho Maisonave CRB: 10/1733

“All we have to decide is what to do with the time that is given to us”.

– GANDALF, *The Lord of the Rings*, J.R.R. TOLKIEN

RESUMO

SILVA, Daniele Fernandes e. **Aprimoramento da Geração de Mapas em Jogos Digitais por Meio de Estratégias Avançadas em GANs**. Orientador: Marilton Sanchotene de Aguiar. 2025. 90 f. Tese (Doutorado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2025.

A geração procedural de conteúdo (PCG) desempenha um papel fundamental na criação de mundos virtuais e na otimização da produção de jogos digitais, permitindo a geração automática de níveis e mapas sem necessidade de intervenção manual. Com os avanços em aprendizado profundo, redes adversárias generativas (GANs) surgiram como uma abordagem promissora para a PCG, oferecendo a possibilidade de criar níveis diversificados e jogáveis. No entanto, ainda há desafios significativos da própria abordagem, como o *mode-collapse*, a instabilidade do treinamento e a dificuldade em garantir que os mapas gerados sejam válidos e coerentes para a jogabilidade. Essa pesquisa explora as políticas de ajuste das GANs aplicadas à geração de mapas para jogos de visão *top-down*, investigando estratégias para otimizar a diversidade e a validade dos mapas gerados. O problema central abordado nesta tese está relacionado à melhoria da qualidade dos níveis gerados por GANs, buscando responder à seguinte questão: como aprimorar modelos de redes generativas adversárias para maximizar a geração de níveis variados e jogáveis em jogos digitais? Para isso, são analisados diversos aspectos do treinamento dessas redes, incluindo a mitigação do *mode-collapse*, a avaliação da qualidade dos mapas gerados, o aumento da expressividade do conjunto de dados e a penalização da rede por heurísticas que favoreçam a jogabilidade. O objetivo principal deste trabalho é definir um conjunto de técnicas e metodologias que otimizem o ajuste de modelos GANs para a geração procedural de níveis em jogos digitais. Para isso, são propostas estratégias que aumentam a expressividade e a qualidade do conjunto de dados, minimizam problemas de treinamento das redes e garantem que os níveis gerados sejam tanto válidos quanto variados. Além disso, busca-se integrar abordagens interativas e heurísticas para tornar o processo generativo mais eficiente e adaptável às necessidades de diferentes tipos de jogos. A abordagem utilizada nesta pesquisa envolve a experimentação com diferentes arquiteturas de GANs, incluindo VanillaGAN, DCGAN e WGANGP, além da incorporação de técnicas como normalização espectral, *bootstrapping* e penalização por heurísticas. São exploradas métricas para avaliar a validade e a jogabilidade dos mapas gerados, bem como estratégias para condicionar a geração dos níveis controladamente. Os experimentos são conduzidos com diferentes conjuntos de dados, comparando abordagens

tradicionais de PCG com as soluções propostas. Os resultados obtidos demonstram que as estratégias desenvolvidas nesta tese contribuem significativamente para a melhoria da geração procedural de níveis, mitigando problemas de *mode-collapse* e instabilidade. Os experimentos evidenciam que técnicas como normalização espectral e penalização por heurísticas aumentam a eficiência do treinamento no ajuste do modelo, melhorando sua capacidade em gerar níveis jogáveis e diversificados. Como conclusão, esta pesquisa destaca a importância do refinamento de políticas de ajuste para GANs na geração procedural de mapas, evidenciando o potencial dessas redes para otimizar o desenho de níveis em jogos digitais. No entanto, desafios ainda permanecem, como a necessidade de desenvolver melhores mecanismos de controle da variabilidade e da complexidade dos mapas gerados. Trabalhos futuros poderão explorar a combinação de GANs com outras abordagens de aprendizado profundo, como modelos generativos condicionais mais avançados, bem como a aplicação dessas técnicas em jogos tridimensionais e outros gêneros, além dos jogos de visão *top-down*.

Palavras-chave: geração procedural de conteúdo; geração de mapa; geração de nível; mapa procedural; aprendizado profundo; redes adversárias generativas.

ABSTRACT

SILVA, Daniele Fernandes e. **Enhancing Map Generation in Digital Games Through Advanced Strategies in GANs**. Advisor: Marilton Sanchotene de Aguiar. 2025. 90 f. Thesis (Doctorate in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2025.

Procedural content generation (PCG) plays a key role in creating virtual worlds and optimizing digital game production, allowing the automatic generation of levels and maps without manual intervention. With advances in deep learning, generative adversarial networks (GANs) have emerged as a promising approach for PCG, offering the possibility of creating diverse and playable levels. However, significant challenges remain, such as mode-collapse, training instability, and difficulty ensuring that the generated maps are valid and consistent for gameplay. This research explores the tuning policies of GANs applied to map generation for top-down games, investigating strategies to optimize the diversity and validity of the generated maps. The central problem addressed in this thesis is improving the quality of levels generated by GANs. It seeks to answer the following question: How can generative adversarial network models be enhanced to maximize the generation of diverse and playable levels in digital games? To this end, several aspects of the training of these networks are analyzed, including mitigating mode-collapse, assessing the quality of the generated maps, increasing the expressiveness of the dataset, and penalizing the network with heuristics that favor gameplay. The main objective of this work is to define a set of techniques and methodologies that optimize the adjustment of GAN models for the procedural generation of levels in digital games. To this end, strategies are proposed that increase the expressiveness and quality of the dataset, minimize network training problems, and ensure that the generated levels are valid and varied. In addition, the aim is to integrate interactive and heuristic approaches to make the generative process more efficient and adaptable to the needs of different types of games. This research uses an approach that involves experimenting with different GAN architectures, including VanillaGAN, DCGAN, and WGANGP, and incorporating techniques such as spectral normalization, bootstrapping, and heuristic penalization. Metrics are explored to evaluate the validity and playability of the generated maps and strategies to condition the generation of levels in a controlled manner. The experiments are conducted with different datasets, comparing traditional PCG approaches with the proposed solutions. The results demonstrate that the strategies developed in this thesis significantly improve procedural level generation and mitigate mode-collapse and instability problems. The experiments show that spectral normalization and heuristic penalization increase training efficiency in model tuning, enhancing its ability to generate playable

and diverse levels. In conclusion, this research highlights the importance of refining adjustment policies for GANs in procedural map generation, highlighting the potential of these networks to optimize level design in digital games. However, challenges remain, such as the need to develop better mechanisms to control the variability and complexity of the generated maps. Future work could explore the combination of GANs with other deep learning approaches, such as more advanced conditional generative models, and apply these techniques in three-dimensional games and other genres, in addition to top-down vision games.

Keywords: procedural content generation; map generation; level generation; procedural map; deep learning; generative adversarial networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Adam	*Adaptive Moment Estimation*
AL	*Adversarial Learning* (Aprendizado Adversarial)
BERT	*Bidirectional Encoder Representations from Transformers*
BN	*Batch Normalization*
BSP	*Binary Space Partitioning*
GAN	*Generative Adversarial Network* (Rede Adversária Generativa)
cGAN	*Conditional Generative Adversarial Network* (Rede Adversária Generativa Condicionais)
CA	*Cellular Automata* (Automatos Celulares)
CNN	*Convolutional Neural Network* (Redes Neurais Convolucionais)
DCGAN	*Deep Convolutional Generative Adversarial Network* (Rede Adversária Generativa Convolucional Profunda)
DCGAN-SN	*Deep Convolutional Generative Adversarial Network with Spectral Normalization* (Rede Adversária Generativa Convolucional Profunda com Normalização Espectral)
DL	*Deep Learning* (Aprendizagem Profunda)
DW	*Drunkard's Walk*
EC	*Evolutionary Computation* (Computação Evolutiva)
GB	Gigabyte
GPU	*Graphics Processing Unit*
IS	*Invalid Sample* (Amostras Inválidas)
LLM	*Large Language Model* (Modelo de Linguagem de Grande Escala)
LSTM	*Long Short-Term Memory*
ME	Métodos Evolutivos
ML	*Machine Leraning* (Aprendizagem De Máquina)
PCG	*Procedural Content Generation* (Geração Procedural de Conteúdo)
PCGML	*Procedural Content Generation via Machine Learning* (Geração Procedural de Conteúdo via Aprendizagem de Máquina

RAM	*Random Access Memory*
ReLU	*Rectified Linear Unit*
RL	*Reinforcement Learning* (Aprendizagem por Reforço)
RMSprop	*Root Mean Squared Propagation*
RPG	*Role-Playing Game*
SGD	*Stochastic Gradient Descent*
SL	*Supervised Learning* (Aprendizado Supervisionado)
SN	*Spectral Normalization*
UFPel	Universidade Federal de Pelotas
USL	*Unsupervised Learning* (Aprendizado Não-Supervisionado)
VAE	*Variational Autoencoder*
VGLC	*Video Game Level Corpus*
VM	*Variability Metric* (Medida de Variabilidade)
VRAM	*Video Random Access Memory*
WGAN	*Wasserstein GAN*
WGAN-SN	*Wasserstein GAN with Spectral Normalization*
WGANGP	*WGAN with Gradient Penalty*
WGANGP-SN	*WGAN with Gradient Penalty and Spectral Normalization*

SUMÁRIO

1 INTRODUÇÃO

Pesquisas na área de jogos digitais vêm impulsionando este mercado e apresentando soluções para a construção de mundos virtuais cujo objetivo é melhorar a experiência e engajamento do jogador (Melhart et al., 2019; Risi; Togelius, 2020; Viana; Santos, 2021). Tipicamente, esta tarefa é realizada por *designers* de jogos que exercem um papel importante na implementação da jogabilidade. Esta é uma tarefa complexa, demandando grande esforço e tempo para a construção dos elementos que contribuem para tornar o jogo mais atrativo e desafiador de se jogar. Nesse processo de construção, aspectos como: enredo, objetivos do jogo, ajuste de interações do jogador, organização do cenário, trilha sonora e níveis de dificuldade potencializam o engajamento do jogador. Para o *designer*, o desafio fica maior quando a experiência do usuário deve ser considerada para o ajuste da jogabilidade (Melhart et al., 2019).

Entender o processo mental e comportamental que mantêm o ser humano motivado pode ser compreendido a partir de estudos na área da psicologia (Oliveira et al., 2021; Csikszentmihalyi; Csikzentmihaly, 1990). Na psicologia positiva, por exemplo, é investigado como os indivíduos percebem, reagem e interagem com os desafios e recompensas do ambiente ao seu redor. Tais processos têm grande influência no desenho de jogos, tendo como uma das bases teóricas a Teoria do Fluxo (Csikszentmihalyi; Csikzentmihaly, 1990). O conceito de Fluxo, proposto por Csikszentmihalyi; Csikzentmihaly (1990), explica esse fenômeno como uma experiência mental na qual uma pessoa está completamente envolvida e focada em uma atividade, a ponto de perder a noção do tempo e do ambiente ao seu redor. No contexto de desenho de níveis, a teoria é abordada como o equilíbrio entre as habilidades do jogador e os desafios apresentados pelo jogo. Também são considerados nesta abordagem a especificação de metas claras e *feedback* imediatos para proporcionar uma experiência que traga alta satisfação ao jogador.

Ao aplicar esses conhecimentos ao desenho de jogos, a criação de experiências não capturam somente a atenção do jogador, mas também mantêm seu interesse e prazer ao longo do tempo, ajustando aspectos como o balanceamento e progressão da dificuldade, inclusão de recompensas tangíveis, estratégias de incentivo aos objetivos

do jogo, entre outros aspectos, devendo alinhar-se com as motivações e capacidades cognitivas dos jogadores (Nacke; Bateman; Mandryk, 2014).

Para otimizar o processo de produção de jogos, os algoritmos de geração procedural de conteúdo (do inglês, *Procedural Content Generation* – PCG) possibilitam a escalabilidade na criação de mapas, eliminando a necessidade de que essa tarefa seja inteiramente manual. Esses algoritmos são desenvolvidos com base em parâmetros, restrições e objetivos pré-definidos, permitindo a geração automática de conteúdo eficientemente (Shaker et al., 2011; Yannakakis; Togelius, 2011; Linden; Lopes; Bidarra, 2014, 2021). No entanto, mesmo com essas melhorias, a definição de regras que atendam às preferências ou necessidades específicas de cada jogador pode tornar o problema mais complexo de modelar.

Dessa forma, o desenvolvimento de uma abordagem mais fácil e intuitiva para a geração procedural de conteúdo em jogos digitais pode otimizar o trabalho do *game designer*, tornando o processo de criação mais eficiente. Além de impactar positivamente na experiência do jogador, ao aprimorar a jogabilidade do nível gerado. Esses benefícios não se restringem apenas aos jogos de RPG, mas podem ser estendidos a diferentes gêneros. Com os avanços e a popularização dos algoritmos de Aprendizado Profundo (em inglês, *Deep Learning* – DL), permitiu-se gerar dados a partir de qualquer domínio, incluindo jogos, eficazmente e com baixo custo computacional.

O DL é um ramo da Inteligência Artificial que permite a um modelo aprender a partir de dados, identificando padrões dos dados sem a necessidade de programação explícita. Sua base são as redes neurais artificiais, que, embora inspiradas no funcionamento das redes neurais biológicas, representam um conjunto de funções compostas por múltiplos blocos chamados neurônios. Esses modelos são treinados com dados reais, permitindo-lhes aprender e resolver problemas de maneira automática (Roberts; Yaida; Hanin, 2022).

Abordagens apresentadas na literatura auxiliam na geração de conteúdo para jogos digitais por meio de algoritmos de aprendizagem de máquina, tais como *Generative Adversarial Networks* (GANs) (Volz et al., 2018; Rodriguez Torrado et al., 2020; Liello et al., 2020; Chen; Lyu, 2022), *Variational Autoencoder* (VAE) (Sarkar; Cooper, 2020, 2021a,b, 2023, 2022) ou *Reinforcement Learning* (RL) (Khalifa et al., 2020; Sestini; Kuhnle; Bagdanov, 2021; Gisslén et al., 2021).

Mesmo que redes neurais profundas sejam utilizadas para a geração de conteúdo de jogos digitais, ainda assim há limitações nos resultados obtidos. Em redes adversárias, por exemplo, os desafios como *mode-collapse* e a instabilidade no treinamento são bem-conhecidos na literatura, devido à abordagem adversarial (tópico detalhado na Seção 2.2), sendo principalmente associadas à necessidade de dados para o treinamento de modelos eficazes. Além disso, outras limitações apresentadas nestes trabalhos incluem mecanismos para aumentar a expressividade do conjunto de dados

para treinamento de DLs, geração diversificada de conteúdo, definição de métricas ou estratégias para a validação de mapas jogáveis que serão discutidas na Seção 2.3.2.

Em função destas limitações e, também, do impacto da geração de conteúdo para a área de jogos digitais baseados em GANs, é oportuno propor a criação e/ou otimização de procedimentos, abordagens e instrumentos computacionais capazes de auxiliar o ajuste das redes, com apoio de heurísticas pré-definidas, a fim de proporcionar a geração diversa de mapas jogáveis.

1.1 Problema de Pesquisa

Com base nas contribuições da literatura sobre geração procedural e na análise dos diversos estudos que aplicam abordagens baseadas em GANs, que serão detalhadas no próximo Capítulo, o problema de pesquisa desta Tese consiste em responder à seguinte questão:

"Como é possível aprimorar modelos de redes generativas adversárias para a geração de níveis variados e jogáveis em jogos digitais?"

Este problema de pesquisa se desdobra nos seguintes aspectos:

- como reduzir os problemas de *mode-collapse* e instabilidade do processo de treinamento das redes?
- como avaliar a qualidade de mapas gerados para jogos digitais?
- como aumentar a expressividade da base de dados para o treinamento?
- como o uso de heurísticas para a penalização da rede pode melhorar seu aprendizado?
- como o uso de métricas que avaliam a validade de um nível para condicionar a rede pode melhorar seu desempenho?

Por fim, é importante ressaltar que os esforços da pesquisa incluem a identificação da arquitetura mais eficaz para a resolução do problema de geração procedural de níveis. Ou seja, encontrar um modelo que gere a maior quantidade possível de níveis variados e válidos, maximizando tanto a diversidade quanto a qualidade das amostras geradas. Esse processo envolve a avaliação/otimização de diferentes arquiteturas de GANs, além de técnicas complementares que melhoram o processo de geração de níveis.

1.2 Objetivos

Esta Tese pretende definir um conjunto de técnicas e metodologias que otimizem o ajuste de modelos de GANs aos dados de níveis de jogos digitais. Além disso, busca-se integrar abordagens interativas e heurísticas para tornar o processo generativo mais eficiente, permitindo uma colaboração mista entre humano e algoritmo na criação de níveis.

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

- proposta de um conjunto de técnicas para aumentar a expressividade e a qualidade do conjunto de dados de treinamento;
- proposta de um conjunto de técnicas para minimizar problemas no ajuste das redes, como *mode-collapse* e instabilidade; e,
- proposta de uma estratégia para que as gerações não sejam apenas válidas, mas também variadas.

1.3 Hipóteses

Nesta Tese foram estabelecidas as seguintes hipóteses para a pesquisa:

- que, a partir de um conjunto de dados limitado, é possível gerar mapas para jogos digitais com auxílio de GANs;
- que a aplicação da normalização espectral em arquiteturas convolucionais pode melhorar o treinamento de GANs para geração de níveis;
- que métricas de qualidade podem ser modeladas como heurísticas para uso no treinamento de GANs para auxiliar no aprendizado de níveis jogáveis;
- que um fator de penalidade baseado no comportamento dos dados quando aplicado à função de custo pode melhorar o treinamento de GANs;
- que adaptações no conjunto de dados de treinamento podem torná-lo mais expressivo e representativo, melhorando o aprendizado da GAN;
- que uma abordagem multimodal pode melhorar o aprendizado de GANs na geração de níveis.

1.4 Contribuições da Tese

As contribuições desta Tese são voltadas para a melhoria da geração procedural de níveis em jogos digitais por meio do uso de GANs, alinhando-se à necessidade de

obter níveis diversos e válidos em relação ao estado da arte. Além disso, busca-se superar desafios conhecidos pela literatura no uso de GANs para essa aplicação.

As principais contribuições desta pesquisa incluem:

- definição de métricas para avaliar a diversidade e validade dos níveis gerados (Seção 3.2).
- proposta de um pipeline otimizado para a geração procedural de níveis com GANs (Capítulo 4);
- investigação do impacto da normalização espectral nas arquiteturas convolucionais das GANs (Seção 4.3);
- desenvolvimento de uma abordagem baseada em penalização por heurísticas (Seção 4.4);
- aplicação de estratégias condicionais para garantir conectividade e jogabilidade nos níveis gerados (Seção 4.6);

Os resultados obtidos demonstram que as estratégias propostas aprimoram a qualidade do treinamento das GANs, mitigando problemas como *mode-collapse* e instabilidade. Além disso, essas estratégias contribuem para a otimização do tempo computacional necessário para a geração de níveis de jogos digitais.

1.5 Estrutura da Tese

A estrutura desta tese está organizada em cinco Capítulos, os quais abrangem tanto a fundamentação teórica quanto os experimentos realizados ao longo da pesquisa.

O Capítulo 1 apresenta o contexto da pesquisa, objetivos principais, motivação e relevância do trabalho. Este Capítulo é o ponto de partida para entender o problema de pesquisa e a organização desta Tese.

O Capítulo 2 apresenta uma visão geral sobre a PCG e sua aplicação para a geração de níveis em jogos digitais. Além disso, o Capítulo aborda o uso de técnicas de *machine learning* (ML) na geração de níveis, sendo explorado de forma mais aprofundada pela autora desta Tese, em um levantamento da literatura. O trabalho foi publicado na revista *Multimedia Tools and Applications*, sob o título *Procedural Game Level Generation with GANs: Potential, Weaknesses, and Unresolved Challenges in the Literature* e será discutido nesta Tese.

O Capítulo 3 descreve a estrutura da abordagem proposta e os métodos desenvolvidos durante o trabalho. Este Capítulo detalha as escolhas metodológicas adotadas

para o desenvolvimento dos experimentos, explicando as justificativas por trás das decisões tomadas ao longo do processo de implementação.

O Capítulo 4 apresenta os resultados dos experimentos realizados, com uma análise dos dados gerados, comparações entre diferentes abordagens e uma discussão sobre as métricas de desempenho adotadas. Dois trabalhos da autora desta Tese foram publicados em conferência e serão apresentados nesse Capítulo. Os artigos, publicados no *Seminário Integrado de Software e Hardware* e no *Simpósio Brasileiro de Jogos e Entretenimento Digital*, são intitulados *Dungeon level generation using generative adversarial network: an experimental study for top-down view games* e *How to improve the quality of GAN-based map generators*, respectivamente.

Por fim, o Capítulo 5 encerra a Tese destacando as contribuições do trabalho para a área de geração de níveis e discussões sobre suas limitações. Além disso, este Capítulo demonstra perspectivas para trabalhos futuros, como aplicação de novas abordagens na área de PCG e ML para jogos.

2 REFERENCIAL TEÓRICO E TECNOLÓGICO

Este Capítulo apresenta os principais conceitos e fundamentos teóricos que embasam esta Tese, fornecendo o contexto necessário para compreender as abordagens e decisões empregadas durante a pesquisa.

2.1 Geração Procedural de Conteúdo

A geração procedural de conteúdo é uma técnica utilizada em computação para criar conteúdos digitais de forma automatizada, baseada em algoritmos e regras predefinidas. A PCG pode ser descrita como um conjunto de técnicas cujo objetivo é produzir diferentes configurações de um dado conteúdo, por meio do uso de algoritmos modelados para controlar parâmetros pré-estabelecidos. O principal objetivo da PCG é produzir conteúdos que sejam variados, escaláveis e consistentes, sem a necessidade de intervenção manual.

2.1.1 História

Os algoritmos de geração procedural têm origem na área da matemática aplicada, para modelar o comportamento real de fenômenos naturais. Com o avanço das técnicas, essas modelagens foram adaptadas para otimização computacional, permitindo a geração de conteúdos como imagens, texturas e modelos tridimensionais eficientemente (Blatz; Korn, 2017). Isso permitiu a aplicação dos algoritmos para a geração de diversos conteúdos em diferentes áreas de pesquisa (Mandelbrot, 1983; Perlin, 1985; Prusinkiewicz; Lindenmayer, 2012).

Conforme descrito na revisão da literatura de Blatz; Korn (2017), os algoritmos de PCG foram inicialmente utilizados em estudos de sistemas biológicos, especificamente da área de botânica. Robert Brown observou o comportamento do movimento aleatório das partículas, como o deslocamento de grãos de pólen em um fluido. Esse estudo serviu de inspiração para a modelagem matemática conhecida como *Wiener Process*. A partir dessa técnica, novos modelos matemáticos foram surgindo para atender diferentes especificidades das representações de fenômenos naturais.

Os fractais, por exemplo, são estruturas matemáticas capazes de representar formas naturais de natureza infinita e em múltiplas escalas (Mandelbrot, 1983), sendo bastante utilizados para representações de árvore, folhas e montanhas. Utilizando estratégias de recursão, como os conjuntos de Julia e Mandelbrot, os fractais permitem criar formas complexas a partir de regras matemáticas relativamente simples.

Diferente dos fractais, a técnica de *Perlin Noise*, desenvolvida por Ken Perlin, não se baseia em recursões. Em vez disso, utiliza gradientes aleatórios aplicados sobre uma grade regular, interpolados para produzir valores contínuos. O efeito visual dessa técnica permite a criação de estruturas mais orgânicas, sendo muito aplicado em texturas para a criação de imagens e em simulações gráficas (Perlin, 1985).

Outra abordagem envolveu sistemas baseados em regras, como os propostos por Prusinkiewicz; Lindenmayer (2012), utilizados para a modelagem de vegetação. Esses sistemas utilizam gramáticas formais, conhecidas como sistemas de Lindenmayer, para gerar estruturas botânicas complexas e detalhadas (Prusinkiewicz; Lindenmayer, 2012). Essa técnica é empregada na criação de representações de ecossistemas, conseguindo capturar a diversidade de algumas formas naturais que seguem padrões hierárquicos e ramificados.

A partir do ano 2000, a área de Inteligência Artificial se popularizou devido à evolução do poder computacional e da quantidade de dados disponíveis na época. Muitos algoritmos de ML entraram em destaque, em especial DL. Nesse contexto, a evolução dos algoritmos generativos trouxeram avanços significativos: os *autoencoders* surgiram em 2007, as GANs em 2014 e os *Large Language Model* (LLM) por volta de 2018.

Os algoritmos descritos até aqui não tiveram como foco a aplicação em jogos digitais. No entanto, a aplicação dessas técnicas no campo dos jogos surgiu a partir da necessidade de otimizar o armazenamento e de gerar conteúdos mais diversos e dinâmicos. Jogos como Rogue (1980), foram pioneiros utilizando técnicas de PCG em jogos, com o uso de algoritmos como divisão espacial e geração de mapas aleatórios.

2.1.2 Geração de Níveis

Na área de jogos digitais, a PCG pode ser aplicada para a construção de níveis, vegetação (Li et al., 2021), música (Maniktala et al., 2020), enredo (Hausknecht et al., 2020), por exemplo. Esta Tese tem como foco a geração de níveis, entendendo o nível como a representação de um conjunto de elementos e regras que determinam a jogabilidade de um jogo.

Existem diversos algoritmos disponíveis para a geração procedural de níveis, cada um adequado a diferentes estilos de criação. Dentre eles, destacam-se os autômatos celulares (Neumann; Burks, 1966; Linden; Lopes; Bidarra, 2014; Kreitzer; Ashlock; Pereira, 2019), métodos evolutivos (Kerssemakers et al., 2012; Volz et al., 2018; Earle

et al., 2022), agentes (Khalifa et al., 2020; Sestini; Kuhnle; Bagdanov, 2021; Gisslén et al., 2021) e, mais recentemente, a geração procedural via aprendizagem de máquina (PCGML) (Summerville et al., 2018).

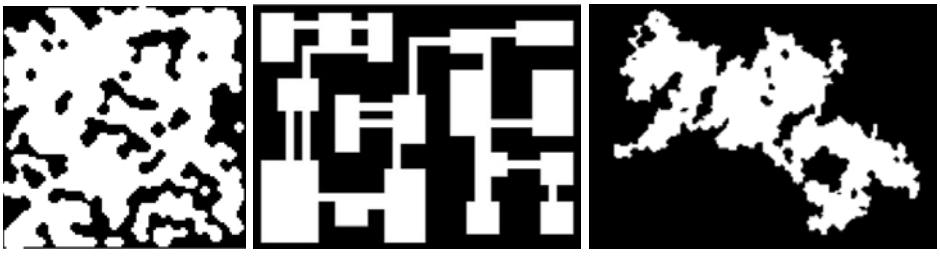
No trabalho de Viana; Santos (2021), os autores classificam a geração de conteúdo para diferentes tipos de jogos com base em diversos fatores, como o gênero do jogo, a representação dos dados, a dimensionalidade e os algoritmos utilizados em PCG. Para jogos de visão *top-down*, conhecidos como *roguelike*, caverna, calabouço ou labirinto, podem ser observadas duas formas de representação dos dados para a geração de conteúdo: uma na forma de grade e outra em grafo.

A estrutura em grade é muito utilizada para a geração de mapas, sendo estes compostos de um conjunto de *tiles*, referindo-se aos elementos capazes de afetar a jogabilidade, tais como: barreiras, itens, inimigos, entre outros. A estrutura em grafo está muito presente na representação das conexões entre salas (Gutierrez; Schrum, 2020).

Os algoritmos tradicionais mais utilizados na geração procedural de conteúdo incluem abordagens baseadas em regras locais, divisão espacial, caminhadas aleatórias e otimização evolutiva.

Os Autômatos Celulares (do inglês, *Cellular Automata* – CA) são amplamente empregados na criação de mapas e terrenos, seguindo regras locais simples para definir a evolução das células no espaço. Apesar de sua simplicidade, esses algoritmos conseguem produzir padrões complexos e estruturas orgânicas, tornando-os úteis para a geração de terrenos e cavernas (à esquerda, na Figura 1).

Figura 1 – Exemplos de estruturas geradas através dos algoritmos tradicionais de PCG para mapas. Resultados de geração através dos algoritmos CA (esquerda), BSP (centro) e DW (direita).



Os métodos de divisão espacial, como as árvores BSP (*Binary Space Partitioning*), operam dividindo o espaço recursivamente em regiões menores, permitindo a criação de mapas estruturados, frequentemente utilizados em jogos que requerem múltiplas salas e corredores interconectados (ao centro, na Figura 1).

A abordagem conhecida como *Drunkard's Walk* (DW), ou *Random Walk*, baseia-se no deslocamento aleatório de um agente a partir de um ponto inicial, simulando o movimento de um "bêbado". Esse método é aplicado para gerar caminhos e túneis de forma imprevisível, sendo útil para a criação de *layouts* de cavernas ou labirintos (à

direita, na Figura 1).

Por fim, os Métodos Evolutivos (ME), como os algoritmos genéticos, são utilizados para gerar conteúdos que atendam a critérios específicos de qualidade. Inspirados na evolução natural, esses métodos utilizam operadores como seleção, cruzamento e mutação para explorar múltiplas possibilidades de geração, permitindo a adaptação do conteúdo gerado a requisitos previamente estabelecidos.

A Tabela 1 apresenta um resumo dos algoritmos discutidos, destacando suas principais vantagens e limitações no processo de geração de conteúdo. Esses algoritmos, embora distintos, podem ser aplicados complementarmente entre si, dependendo do contexto e das necessidades específicas de geração (Minini; Assuncao, 2020).

Tabela 1 – Resumo dos algoritmos tradicionais de PCG aplicado em jogos digitais.

Algoritmo	**Vantagens**	**Limitações**
CA	Criação de padrões orgânicos e complexos a partir de regras simples; bom para gerar cavernas e terrenos naturais	Dificuldade em controlar o resultado final; pode ser computacionalmente intensivo para grandes áreas.
BSP	Produz *layouts* bem estruturados, ótimo para jogos que exigem ambientes com várias salas e corredores.	Pode resultar em *layouts* repetitivos e pouco orgânicos; requer planejamento cuidadoso das regras de divisão.
DW	Simples de implementar, gera mapas com caminhos orgânicos e interconectados.	Pode resultar em caminhos aleatórios e pouco interessantes; difícil de controlar o comprimento e a direção dos caminhos
ME	Capacidade de encontrar soluções ótimas ou satisfatórias; pode ser usado para otimizar a qualidade do conteúdo gerado.	Requer a definição de métricas de qualidade claras; pode ser computacionalmente caro e demorado.

2.2 Geração Procedural de Níveis e *Machine Learning*

Estudos na área de jogos digitais têm avançado na exploração de novos algoritmos e metodologias para a PCG, ampliando as abordagens inteligentes além da computação evolutiva, que, por muitos anos, foi a principal estratégia adotada nesse domínio. Hendrikx et al. (2013) propuseram uma taxonomia composta por seis camadas de conteúdo de jogo (Elementos, Espaço, Sistemas, Cenários, Desenho e Conteúdo derivado), visando classificar as técnicas de PCG conforme os diferentes tipos de conteúdo que podem ser criados. Essa estrutura oferece uma visão abrangente e organizada das abordagens de PCG, facilitando a compreensão e análise das diversas metodologias aplicadas no desenvolvimento de jogos digitais. Essa taxonomia representa um ponto de partida para compreender as múltiplas abordagens aplicadas à geração de conteúdo em jogos. De maneira semelhante, Smelik et al. (2014) classificaram pesquisas de PCG conforme os tipos de conteúdo gerados em mundos virtuais, como terrenos, vegetação, rios, estradas, edifícios e cidades. Além disso, Shaker; Togelius; Nelson (2016) consolidaram o conhecimento existente ao publicar o primeiro livro-texto dedicado ao PCG em jogos, fornecendo uma visão abrangente

sobre o campo.

Trabalhos subsequentes aprofundaram-se na revisão de abordagens específicas para a geração de conteúdo. Summerville et al. (2018) analisaram modelos de ML treinados a partir de dados existentes para a criação automática de novos conteúdos, destacando o termo PCGML para os algoritmos que utilizem o aprendizado de máquina para geração de conteúdo. Essa pesquisa foi posteriormente expandida por CONSTRAINT-BASED PCGML APPROACHES (2022), onde analisam diferentes estratégias de PCGML aplicadas à geração de conteúdo para jogos digitais. Complementando esse panorama, Liu et al. (2021) investigaram o uso de métodos de aprendizado profundo na geração de conteúdo, discutindo direções futuras, como geração assistida e transferência de estilo.

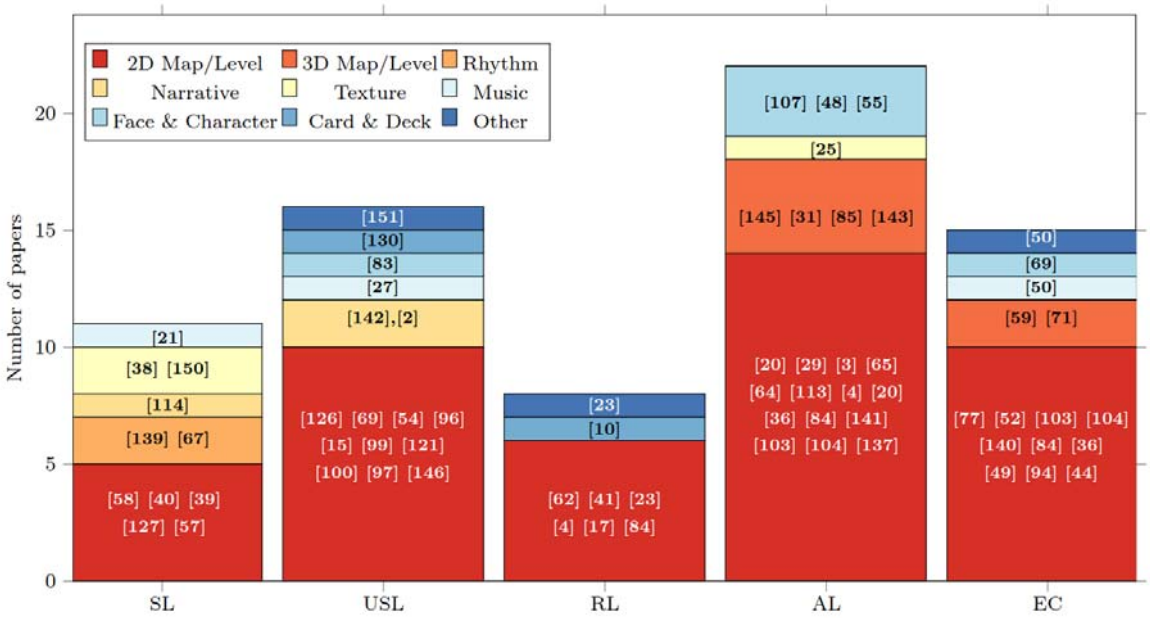
O estudo de Liu et al. (2021) evidencia uma tendência de pesquisas voltadas à geração de níveis e mapas, com destaque para ambientes bidimensionais (Figura 2). No contexto de geração de níveis, os autores analisam como a literatura aplica diferentes métodos a esse domínio. No Aprendizado Supervisionado, por exemplo, o foco está em predizer o comportamento do *gameplay* visando avaliar e/ou ajustar um gerador conforme os dados do jogador. Ou seja, os algoritmos em questão estão preocupados em, a partir de metadados, gerar mapas condicionados a um comportamento específico.

O RL vem sendo gradualmente explorado para a geração de mapas. No entanto, a tarefa de treinamento dos agentes é desafiadora, por envolver explorar o ambiente (neste caso, o processo de representação de mapas) e aprender uma política otimizada com base em recompensas recebidas ao longo do tempo, o que não é trivial. O aprendizado dos agentes depende de fatores como: a escolha do algoritmo, o equilíbrio entre *exploration* e *exploitation*, o fator de desconto, e a modelagem do ambiente (regras de ações, estados e recompensas). Diferente do processo de decisão de Markov, essa abordagem precisa aprender a partir de tentativas e erros, e não de um modelo prévio do ambiente.

No Aprendizado Não-Supervisionado, os algoritmos treinam com dados não rotulados, ou seja, sem informações explícitas sobre a categoria ou significado dos exemplos. No contexto de jogos, destacam-se VAEs, GANs e LSTMs, cada um com suas particularidades na geração de níveis. Devido à natureza das GANs, Liu et al. (2021) classificam os trabalhos dessa categoria no Aprendizado Adversarial, conforme demonstrado na Figura 2. Isso evidencia a popularidade das GANs e suas variantes na literatura. Além disso, os autores ressaltam que as GANs se mostram ideais na criação de elementos visuais, como sprites, paisagens e mapas, ao estruturar o conteúdo em forma de matrizes 2D de *tiles* ou imagens baseadas em píxeis.

Entre as diversas revisões da literatura que abordam PCG abrangentemente, alguns trabalhos direcionam o foco da revisão para gêneros de jogos específicos, como

Figura 2 – Trabalhos da literatura apresentados por métodos – Aprendizado Supervisionado (SL), Aprendizado Não-Supervisionado (USL), Aprendizado por Reforço (RL), Aprendizado Adversarial (AL) e Computação Evolutiva (EC) – e tipos de conteúdo, segundo a pesquisa de Liu et al. (2021).



jogos de masmorra (Linden; Lopes; Bidarra, 2014; Viana; Santos, 2021) e jogos de quebra-cabeça (De kegel; Haahr, 2020). Outras abordagens concentraram-se em técnicas específicas, como algoritmos evolucionários e meta-heurísticas (Togelius et al., 2011), além de métodos baseados em transformação de conhecimento (Sarkar et al., 2023). Ademais, técnicas como VAEs e RL (Khalifa et al., 2020; Gisslén et al., 2021; Bontrager; Togelius, 2021) continuam a ser exploradas na geração de níveis de jogos por meio de aprendizado de máquina. Apesar dos avanços mencionados e da popularidade das GANs entre os pesquisadores da área, não foram identificadas pesquisas que apresentem um panorama da aplicação de GANs para o desenho de níveis em jogos, evidenciando uma lacuna relevante a ser explorada. As únicas exceções, no entanto, são dos estudos de Hughes; Zhu; Bednarz (2021) e de Liu et al. (2021), que mencionam brevemente a geração de níveis baseada em GANs, porém sem se aprofundar nos desafios e particularidades desse modelo.

As GANs são arquiteturas de redes neurais profundas pertencentes à classe de aprendizado de máquina conhecida como aprendizado não supervisionado. Nas Seções seguintes, serão apresentados os principais conceitos e variações dessas arquiteturas, fornecendo a base necessária para a compreensão dos temas abordados ao longo deste trabalho.

2.2.1 Vanilla *Generative Adversarial Network*

Introduzidas por Goodfellow et al. (2014, 2020), as GANs consistem em duas redes neurais (um *gerador* e um *discriminador*) competindo entre si. O objetivo dessa arquitetura, conhecida como VanillaGAN, é aprender uma distribuição probabilística que melhor represente o conjunto de dados. O *gerador* tem a função de aprender a mapear um *seed* – também conhecido como ruído ou espaço latente – em dados novos que seguem a distribuição aprendida pelo gerador. O *discriminador* tem como função aprender a distinguir dados reais – que mais se aproximam da distribuição dos dados de treino – dos dados sintetizados pelo gerador.

As redes são treinadas juntas, de forma que a competição entre elas beneficie o ajuste do modelo de ambas, através de *backpropagation*. Isso acontece, pois o discriminador ajuda o gerador a sintetizar amostras mais próximas da distribuição do conjunto de dados, enquanto o gerador produz amostras não vistas para treinar o discriminador a rotular corretamente o dado entre real e falso. O ajuste das redes acontece através da função de custo, descrita na Equação 1.

$$V(G,D) = \mathbb{E}_{p_{data}(x))} log D(x) + \mathbb{E}_{p_g(x)} log(1 - D(G(z)) \tag{1}$$

O discriminador tenta maximar a função de custo $V(G,D)$ aprendendo a classificar o conjunto de dados x em real – seguindo a função $D(x)$ – e os dados gerados $G(z)$ em *fake* – seguindo a função $D(G(z))$, onde z se refere ao espaço latente como entrada do gerador. Essa rotulagem é feita através da entropia cruzada (do inglês, *cross-entropy*), representada pelo log de cada termo da equação. O gerador atua diretamente no segundo termo da equação para minimizar a função de custo $(1-D(G(z)))$ maximizando a saída de $G(z)$, ou seja, o gerador vai tentar enganar o discriminador a rotular os dados gerados como sendo reais.

2.2.2 *Conditional Generative Adversarial Network*

Na GAN anteriormente descrita, não há um mecanismo eficiente para controlar as características das amostras geradas durante o treinamento. Para solucionar essa limitação, Mirza; Osindero (2014) propuseram a inclusão de uma informação adicional como condição na entrada da rede, resultando na arquitetura conhecida como *Conditional* GAN (cGAN). Essa modificação permite que o modelo gere amostras orientadas por rótulos ou atributos específicos. O ajuste da rede nessa abordagem é representado pela Equação 2.

$$V(G,D) = \mathbb{E}_{p_{data}(x))} log D(x|y) + \mathbb{E}_{p_g(x)} log(1 - D(G(z|y)) \tag{2}$$

Com a introdução de um dado conhecido y junto ao espaço latente z, o comportamento da rede passa a ser condicionado por essa informação adicional. Essa

modificação mantém a estrutura básica da arquitetura original, alterando somente as camadas de entrada e saída do gerador, bem como a camada de entrada do discriminador. Dessa forma, o gerador passa a mapear o espaço latente condicionado ao dado extra, seguindo a notação $G(z|y)$, permitindo a geração de amostras orientadas por atributos específicos.

2.2.3 *Deep Convolutional Generative Adversarial Network*

Amplamente utilizadas na literatura, as GANs tradicionais apresentam desafios conhecidos relacionados à instabilidade e à convergência. A instabilidade diz respeito à dificuldade do gerador em produzir amostras coerentes com a distribuição dos dados de treinamento. Uma das principais estratégias adotadas para mitigar esse problema foi a substituição das camadas totalmente conectadas por camadas convolucionais profundas, conforme proposto por Radford; Metz; Chintala (2016).

Assim como a instabilidade, a dificuldade de convergência do gerador ocorre, em muitos casos, devido ao rápido ajuste da rede discriminadora, que compromete o processo de aprendizado do gerador. Na literatura, essa dificuldade também é associada ao problema conhecido como *mode-collapse*, no qual o gerador falha em capturar toda a complexidade da distribuição de dados, restringindo-se à geração de uma única amostra ou de um conjunto limitado de amostras. Uma possível causa desse problema é o desequilíbrio no treinamento das redes. Fundamentada na teoria dos jogos, a GAN deve manter um equilíbrio dinâmico entre o gerador e o discriminador para que ambos sejam ajustados proporcionalmente, alcançando o chamado equilíbrio de Nash (Ratliff; Burden; Sastry, 2013).

2.2.4 Wasserstein *Generative Adversarial Networks with Gradient Penalty*

Outra abordagem para mitigar o desequilíbrio entre as redes foi a proposta de substituir a função de custo tradicional por uma função baseada na distância de Wasserstein (Arjovsky; Chintala; Bottou, 2017), juntamente com a adição de uma penalidade no discriminador (Gulrajani et al., 2017), visando desacelerar seu ajuste em relação ao gerador.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \mathbb{E}_{x\ \mathbb{P}_r}[D(x)] - \mathbb{E}_{\tilde{x}\ \mathbb{P}_g}[D(\tilde{x})] \tag{3}$$

A função de custo da arquitetura Wasserstein GAN (WGAN) é definida conforme a Equação 3, onde $\tilde{x} = G(z)$ refere-se aos dados gerados a partir de um espaço latente z. Os autores utilizam o termo “crítico” para se referir ao discriminador, pois, em sua abordagem, o crítico não é treinado para classificar, mas sim para estimar a distância entre as distribuições. Nessa configuração, os autores enfatizam que o treinamento do crítico ideal contribui para melhorar o ajuste do gerador. O processo de treinamento é

baseado na minimização da distância entre as distribuições de dados reais e gerados, com o crítico tentando maximizar essa distância, enquanto o gerador trabalha para reduzir a discrepância, gerando dados mais próximos da distribuição dos dados reais $D(\tilde{x})$.

A WGAN trouxe melhorias em relação à instabilidade da rede, no entanto, ainda persistiam problemas de convergência e resultados de baixa qualidade. Para resolver tais limitações, Gulrajani et al. (2017) propuseram uma forma de penalização da norma do gradiente do crítico, como demonstra a Equação 4.

$$L = \mathbb{E}_{\tilde{x}\ \mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\ \mathbb{P}_r}[D(x)] + \underbrace{\lambda \mathbb{E}_{\hat{x}\ \mathbb{P}_{\hat{x}}}[(\parallel \nabla_{\hat{x}} D(\hat{x}) \parallel_2 -1)^2]}_{\text{penalidade}} \tag{4}$$

A arquitetura WGAN *with Gradient Penalty* (WGANGP) teve a inclusão desta penalidade na função original da WGAN, junto ao coeficiente de penalidade definido por λ. Para cada iteração da época, durante o processo de treinamento, o gerador é ajustado n iterações de crítico. Dessa forma, é possível treinar o crítico até o seu modelo ótimo sem apesentar o problema de *mode-collapse*.

O ajuste para o crítico ótimo ocorre devido à restrição 1-Lipschitz aplicada à função discriminadora, que contribui para a estabilidade do treinamento. A propriedade 1-Lipschitz aplicada ao crítico significa que a variação entre as saídas do crítico para duas amostras x_1 e x_2 não pode ser maior que a variação entre as próprias amostras. A Equação 5 demonstra a restrição, cujo objetivo é limitar a taxa de variação da saída $D(x)$.

$$\parallel D(x_1) - D(x_2) \parallel \leq \parallel x_1 - x_2 \parallel_2 \tag{5}$$

O conceito garante que não haja um crescimento descontrolado no gradiente desta função durante o treinamento, forçando que o gradiente da saída $D(x)$ seja uniformemente limitado. A penalidade é aplicada para impor a restrição 1-Lipschitz, de forma que a norma do gradiente ($\parallel \nabla_{\hat{x}} D(\hat{x}) \parallel_2$), calculada em relação às amostras reais e gerada com a saída do crítico, deva se aproximar de 1. Caso contrário, é calculada uma contribuição de penalização para limitar o gradiente e manter a suavidade na saída do crítico. Dessa forma, essa estratégia de penalização age como uma forma de regularização, auxiliando a manter a estabilidade do treinamento.

2.3 Estado da Arte

A revisão do estado da arte apresentada aqui contém trechos essenciais para a compreensão do uso de GANs na geração de níveis, sendo aprofundada na *survey* publicada em Silva; Torchelsen; Aguiar (2025), intitulada *“Procedural game level generation with GANs: potential, weaknesses, and unresolved challenges in the literature”*.

No artigo são apresentados os principais trabalhos relacionados à geração de níveis utilizando GANs, permitindo uma visão geral do estado da arte nesse campo e destacando desafios, contribuições relevantes e problemas em aberto.

2.3.1 Principais Problemas

A análise da revisão permitiu identificar os principais problemas da área de geração de níveis onde as GANs são aplicadas. Os trabalhos revisados foram organizados em três categorias principais: *Geração de Níveis*, *Coesão Estrutural* e *Controle da Geração*, cada uma abordando aspectos específicos da geração procedural de conteúdo em jogos.

Na categoria *Geração de Níveis*, foram identificados cinco desafios centrais:

Mapa Completo: criação de níveis inteiros para diferentes gêneros de jogos;

Segmentos/Salas: geração de partes específicas ou salas individuais nos níveis;

Iterativo/Dinâmico: abrange métodos de geração adaptativa que ajustam o nível em tempo real;

Múltiplos Jogos: abordagens generalizáveis para diferentes tipos de jogos; e,

Mapas Realistas: geração de ambientes mais próximos da realidade por meio de imagens de satélite.

A categoria *Coesão Estrutural* inclui trabalhos que propõem mecanismos para reparar ou modificar as GANs, garantindo que os níveis gerados atendam a critérios específicos de jogabilidade e desenho. Dentro dessa categoria, destacam-se três desafios principais:

Correção/Reparo de Elementos do Jogo: garantir a correta disposição dos elementos conforme regras estruturais e de desenho;

Conectividade entre Segmentos/Salas: métodos para assegurar transições coerentes entre diferentes partes do nível; e,

Caminho Conectado: garantir a navegabilidade fluida e acessível ao longo do mapa.

Por fim, a categoria *Controle da Geração* reúne estudos focados na controlabilidade do processo de geração de níveis, possibilitando ajustes específicos para atender a requisitos de desenho e jogabilidade. Os principais desafios dessa categoria incluem:

Ajuste de Elementos: controle sobre a disposição e organização dos componentes do jogo;

Ajuste de *Layout*: modificação da topologia dos níveis para atender a exigências específicas; e,

Ajuste de Dificuldade: estratégias para balancear o nível de desafio e proporcionar uma experiência adequada aos jogadores.

Esta análise dos problemas, destacadas em Silva; Torchelsen; Aguiar (2025), é fundamental para identificação das lacunas de pesquisa na áreea de geração de níveis com GANs. Esses desafios, resumidos na Tabela 2, evidenciam a complexidade do processo e a necessidade de novas abordagens para aprimorar a qualidade, coerência e controle sobre os conteúdos gerados. A partir dessa análise, estabeleceu-se como meta buscar soluções que abordam diretamente as limitações de: mapa completo, mapas realistas, correção/reparo de elementos, caminho conectado, ajuste de elementos e ajuste de layout. As abordagens propostas serão apresentadas no Capítulo 3 e os resultados demonstrados no Capítulo 4.

Tabela 2 – Revisão dos problemas de pesquisa sobre geração de níveis categorizados por gênero de jogo.

	Problema	**Plataforma**	**Caverna**	***Puzzle***	**Outros**
Geração de Níveis	Mapa completo	(Awiszus; Schubert; Rosenhahn, 2020), (Rajabi et al., 2021), (Schubert; Awiszus; Rosenhahn, 2021), (Steckel; Schrum, 2021)	(Giacomello; Lanzi; Loiacono, 2018), (Giacomello; Lanzi; Loiacono, 2019), (Park et al., 2019), (Kumaran; Mott; Lester, 2019), (Gutierrez; Schrum, 2020), (Giacomello; Lanzi; Loiacono, 2023), (Silva et al., 2023), (Silva; Torchelsen; De aguiar, 2024)	(Hald et al., 2020), (Abraham; Stephenson, 2023), (Volz et al., 2020)	(Beckham; Pal, 2017), (Guérin et al., 2017), (Awiszus; Schubert; Rosenhahn, 2020), (Kelvin; Anand, 2020), (Ping; Dingli, 2020), (Awiszus; Schubert; Rosenhahn, 2021), (Schubert; Awiszus; Rosenhahn, 2021), (Voulgaris; Mademlis; Pitas, 2021), (Nunes; Dias; Santos, 2023)
	Segmentos/ Salas	(Volz et al., 2018), (Liello et al., 2020), (Capps; Schrum, 2021), (Mirgati; Guzdial, 2023)	(Kim et al., 2020), (Zhang et al., 2020), (Rodriguez Torrado et al., 2020), (Gutierrez; Schrum, 2020), (Ramos; Santos; Dias, 2023), (Irfan; Zafar; Hassan, 2019), (Usman; Anwar; Rauf, 2023)		
	Iterativo/ Dinâmico		(Kim et al., 2020), (Wang; Liu; Yannakakis, 2021)		(Wang; Liu; Yannakakis, 2021)
	Múltiplos Jogos	(Mirgati; Guzdial, 2023)	(Kumaran; Mott; Lester, 2019), (Irfan; Zafar; Hassan, 2019)		
	Mapas Realistas				(Ramos; Santos; Dias, 2023), (Nunes; Dias; Santos, 2023)
Coesão Estrutural	Correção/Reparo de Elementos do Jogo	(Liello et al., 2020), (Capps; Schrum, 2021), (Steckel; Schrum, 2021), (Wang; Liu, 2022)	(Zhang et al., 2020), (Gutierrez; Schrum, 2020), (Ferber et al., 2024)	(Hald et al., 2020), (Abraham; Stephenson, 2023), (Volz et al., 2020)	
	Conectividade entre Segmentos/Salas	(Capps; Schrum, 2021), (Wang; Liu, 2022), (Nam; Hsueh; Ikeda, 2023)	(Gutierrez; Schrum, 2020), (Kim et al., 2020)		
	Caminho Conectado	(Steckel; Schrum, 2021)	(Silva et al., 2023), (Silva; Torchelsen; De aguiar, 2024)		
Controle da Geração	Ajuste de Elementos	(Schrum et al., 2020)	(Rodriguez Torrado et al., 2020), (Schrum et al., 2020)	(Hald et al., 2020)	
	Ajuste de *Layout*		(Gutierrez; Schrum, 2020)	(Hald et al., 2020), (Volz et al., 2020)	(Wang; Liu; Yannakakis, 2021), (Guérin et al., 2017), (Ping; Dingli, 2020), (Kelvin; Anand, 2020)
	Ajuste de Dificuldade	(Rajabi et al., 2021), (Volz et al., 2018), (Wang; Liu, 2022)			

2.3.2 GANs e Variantes

Na literatura, observa-se uma ampla variedade de redes utilizadas no treinamento de GANs para geração de níveis, com destaque para as redes convolucionais. A pesquisa de Silva et al. (2023) revelou a vulnerabilidade, quanto ao *mode-collapse*, de redes totalmente conectadas com camadas lineares, como a VanillaGAN, destacando a superioridade das camadas convolucionais para o treinamento de níveis (discutido mais aprofundadamente na Seção 4.1). Resultados semelhantes podem ser encontrados no trabalho de Usman; Anwar; Rauf (2023).

Um dos pioneiros, aplicando GANs na geração de níveis, é o trabalho de Giacomello; Lanzi; Loiacono (2018), que explorou a PCG para geração de níveis para o jogo DOOM, utilizando as arquiteturas WGANGP e WGANGP condicional. Esses modelos resultaram na geração de imagens representando diferentes classes de estruturas dos níveis de DOOM. O estudo demonstrou uma melhoria na qualidade das gerações ao utilizar a rede condicional, uma tendência corroborada por outros trabalhos (Ping; Dingli, 2020), evidenciando a eficácia do treinamento com rótulos na orientação do aprendizado da rede e no controle sobre o conteúdo gerado.

Kumaran; Mott; Lester (2019) propõem uma arquitetura para gerar níveis de jogo para múltiplos jogos enquanto compartilham um espaço latente comum. A arquitetura consiste em um único gerador capaz de gerar N resultados distintos, cada um correspondendo a um jogo. Da mesma forma, N discriminadores são treinados individualmente para cada jogo. Mirgati; Guzdial (2023) propõem uma arquitetura que combina um VAE e uma GAN para gerar segmentos de múltiplos jogos de plataforma.

O TOAD-GAN (Schubert; Awiszus; Rosenhahn, 2021) adapta a arquitetura SinGAN para funcionar com jogos baseados em *tokens*, como *Super Mario Bros.*, introduzindo modificações no processo de redução de escala e na determinação da importância dos *tokens*, permitindo assim a geração de conteúdo 2D realista. No entanto, a geração de conteúdo 3D exige considerações adicionais devido ao aumento do tamanho das amostras e do espaço necessário em GPU. Enquanto o TOAD-GAN foca na geração de conteúdo para jogos 2D baseados em *tokens*, o World-GAN (Awiszus; Schubert; Rosenhahn, 2021) estende essa abordagem para ambientes 3D, como *Minecraft*.

O World-GAN incorpora convoluções 3D e *embeddings* densos de *tokens*, chamados *block2vec*, para lidar com estruturas tridimensionais. Baseado em *word2Vec*, o *block2Vec* mapeia blocos de nível para vetores de características em um espaço contínuo, de modo a capturar as relações e semelhanças entre os blocos. Além disso, o World-GAN avalia o conteúdo gerado com base na similaridade de padrões, variabilidade e manuseio de *tokens* raros. Em um trabalho subsequente, Awiszus; Schubert; Rosenhahn (2023) expandem o World-GAN utilizando *embeddings* baseados na linguagem natural do modelo BERT, concluindo que essa abordagem supera métodos

anteriores.

Para a geração de níveis em jogos do tipo *bullet hell*, Wang; Liu; Yannakakis (2021) apresentam duas arquiteturas para a geração de séries temporais: TimeGAN e *Periodic Spatial GAN*. A arquitetura TimeGAN integra um codificador e decodificador estocásticos, uma rede adversarial de séries temporais e uma técnica de regularização baseada em previsão de sequência, garantindo a coerência temporal das séries geradas. Por outro lado, a arquitetura *Periodic Spatial GAN* busca gerar dados espaciais com padrões periódicos. Os trabalhos mencionados e suas principais características, incluindo uma breve descrição das arquiteturas utilizadas (formatos de entrada e saída), são resumidos na Tabela 3. Nela, é possível visualizar as arquiteturas e estratégias já exploradas na literatura para cada tipo de problema de geração.

Tabela 3 – Visão geral de trabalhos da literatura detalhando a contribuição do trabalho, dados de entrada, saída e arquitetura proposta.

Geração	**Entrada condicional**	**Saída**	**Arq.**	**Ref.**
Nível Completo	-	Baseado em *tile*	WGAN	(Volz et al., 2018), (Steckel; Schrum, 2021)
			DCGAN	(Kumaran; Mott; Lester, 2019), (Park et al., 2019), (Gutierrez; Schrum, 2020)
			GAN, DCGAN, WGAN, WGANGP, DCGAN-SN, WGAN-SN, WGANGP-SN	(Silva et al., 2023)
			DCGAN-SN	(Silva; Torchelsen; De aguiar, 2024)
			timeGAN + periodicSpacialGAN	(Wang; Liu; Yannakakis, 2021)
			WGANGP	(Abraham; Stephenson, 2023)
			DCGAN + RL	(Rajabi et al., 2021)
			GenCO	(Ferber et al., 2024)
			GlobalGAN	(Volz et al., 2020)
		Baseado em imagem	WGANGP + CMA-ES	(Giacomello; Lanzi; Loiacono, 2019)
			WGANGP	(Giacomello; Lanzi; Loiacono, 2018), (Giacomello; Lanzi; Loiacono, 2023)
		Block2vec	World-GAN	(Awiszus; Schubert; Rosenhahn, 2021)
	Vetor de características de *embedding*	Baseado em *tile*	CESAGAN	(Rodriguez Torrado et al., 2020)
	Máscara	Baseado em *tile*	WGAN + CNN	(Ping; Dingli, 2020)
	Token	Baseado em *tile*	TOAD-GAN	(Awiszus; Schubert; Rosenhahn, 2020), (Schubert; Awiszus; Rosenhahn, 2021)
	Vetor de características de *embedding*	Baseado em *tile*	WGANGP-PE	(Hald et al., 2020)
	Características do nível	Baseado em imagem	WGANGP	(Giacomello; Lanzi; Loiacono, 2018), (Giacomello; Lanzi; Loiacono, 2023)
Segmento de nível	-	Baseado em *tile*	DCGAN	(Irfan; Zafar; Hassan, 2019)
			MarioGAN + Fractional-Conv	(Wang; Liu, 2022)
			multiWGAN	(Capps; Schrum, 2021)
			VAE-GAN	(Mirgati; Guzdial, 2023)
			WGAN	(Zhang et al., 2020), (Schrum et al., 2020)
		Baseado em imagem	GAN, DCGAN, WGAN	(Usman; Anwar; Rauf, 2023)
	Nível anterior	Baseado em *tile*		(Nam; Hsueh; Ikeda, 2023)
	Vetor de restrição binária	Baseado em *tile*	CAN	(Liello et al., 2020)
	Imagem atual	Baseado em imagem	GameGAN	(Kim et al., 2020)
Terreno	*Sketch*	Baseado em *tile*	Pix2Pix	(Guérin et al., 2017), (Kelvin; Anand, 2020)
	IMR	Heightmap	DCCWGAN	(Ramos; Santos; Dias, 2023)
	-	Heightmap e textura	Pix2Pix	(Beckham; Pal, 2017)
	-	Heightmap	DCGAN, WGAN, ProgGAN, VAE + WGAN	(Nunes; Dias; Santos, 2023)
	Mapas de dispersão	Baseado em imagem	GAN-terrain	(Voulgaris; Mademlis; Pitas, 2021)

Além disso, com a revisão realizada, foi possível identificar diversas métricas para avaliar o desempenho da geração do modelo treinado, que podem ser categorizadas em três aspectos principais: jogabilidade, variabilidade e topologia. Métricas de jogabilidade visam garantir que os níveis gerados sejam minimamente jogáveis e vencíveis. Métricas de variabilidade procuram avaliar a diversidade e a novidade entre as amostras geradas. Já as métricas de topologia avaliam a fidelidade da geração em relação aos dados reais. Essa distinção das métricas entre os diferentes gêneros de jogos permite uma análise mais abrangente das possibilidades de utilização para avaliar o desempenho de um modelo treinado, conforme resumido na Tabela 4.

Tabela 4 – Visão geral das métricas que avaliam as saídas dos modelos treinados.

	Jogabilidade		**Variabilidade**		**Topologia**	
Plataforma	A*	(Volz et al., 2018), (Liello et al., 2020), (Awiszus; Schubert; Rosenhahn, 2020), (Schubert; Awiszus; Rosenhahn, 2021), (Steckel; Schrum, 2021), (Capps; Schrum, 2021), (Nam; Hsueh; Ikeda, 2023), (Mirgati; Guzdial, 2023)	Tile Pattern KL-divergence	(Awiszus; Schubert; Rosenhahn, 2020), (Schubert; Awiszus; Rosenhahn, 2021)	Acurácia	(Mirgati; Guzdial, 2023)
	Reinforcement Learning e Deep Q-networks	(Rajabi et al., 2021)	Métrica de novidade	(Capps; Schrum, 2021)	Linearidade	(Mirgati; Guzdial, 2023)
	Restrições de jogabilidade	(Volz et al., 2018), (Liello et al., 2020), (Capps; Schrum, 2021)	Level embedding	(Awiszus; Schubert; Rosenhahn, 2020), (Schubert; Awiszus; Rosenhahn, 2021)	Tolerância	(Mirgati; Guzdial, 2023)
	Avaliação empírica	(Schrum et al., 2020)	Norma L1 Avaliação empírica	(Liello et al., 2020) (Schrum et al., 2020)	Interessância	(Mirgati; Guzdial, 2023)
Caverna	A*	(Rodriguez Torrado et al., 2020), (Zhang et al., 2020), (Gutierrez; Schrum, 2020), (Rodriguez Torrado et al., 2020), (Zhang et al., 2020)	Métrica de duplicatas	(Rodriguez Torrado et al., 2020), (Zhang et al., 2020)	Erro de canto	(Giacomello; Lanzi; Loiacono, 2018)
	Agente do GVGAI	(Kumaran; Mott; Lester, 2019), (Irfan; Zafar; Hassan, 2019)	Busca por novidade	(Lehman; Stanley, 2010)	Erro de codificação	(Giacomello; Lanzi; Loiacono, 2018)
	Reinforcement Learning	(Kim et al., 2020)	Algoritmo de Dijkstra	(Kumaran; Mott; Lester, 2019)	Entropia	(Giacomello; Lanzi; Loiacono, 2018)
	Algoritmo de Dijkstra	(Park et al., 2019)	Variância pixel a pixel	(Gutierrez; Schrum, 2020), (Silva; Torchelsen; De aguiar, 2024)	Teste Kolmogorov–Smirnov (Giacomello; Lanzi; Loiacono, 2019), (Giacomello; Lanzi; Loiacono, 2023)	
	Restrições de jogabilidade	(Park et al., 2019), (Rodriguez Torrado et al., 2020), (Zhang et al., 2020), (Silva et al., 2023), (Silva; Torchelsen; De aguiar, 2024), (Usman; Anwar; Rauf, 2023)	Diversidade	(Ferber et al., 2024)	Norma L2 (Giacomello; Lanzi; Loiacono, 2019)	
	Consistência de pixel a longo prazo	(Kim et al., 2020)	Medidas de distância	(Kumaran; Mott; Lester, 2019), (Rodriguez Torrado et al., 2020), (Zhang et al., 2020)	Percentis (controlabilidade da geração))	(Giacomello; Lanzi; Loiacono, 2023)
	Correspondência	(Voulgaris; Mademlis; Pitas, 2021)	Avaliação empírica	(Schrum et al., 2020)	Structural Similarity (SSIM)	(Giacomello; Lanzi; Loiacono, 2018)
	Plauibilidade	(Voulgaris; Mademlis; Pitas, 2021)			Fidelidade	(Ferber et al., 2024)
	Avaliação empírica	(Schrum et al., 2020)			Avaliação empírica	(Ramos; Santos; Dias, 2023), (Nunes; Dias; Santos, 2023), (Kelvin; Anand, 2020), (Guérin et al., 2017)
Puzzle	Peças quebradas	(Hald et al., 2020)	Largura, altura	(Abraham; Stephenson, 2023)	Simetria	(Hald et al., 2020), (Volz et al., 2020)
	Ilhas de cor.	(Hald et al., 2020)	Densidade, forma	(Abraham; Stephenson, 2023)		
	Distribuição de peças	(Hald et al., 2020)	Frequência dos blocos	(Abraham; Stephenson, 2023)		
	Destruição de blocos e velocidade dos blocos	(Abraham; Stephenson, 2023)				
Outros	A*	(Awiszus; Schubert; Rosenhahn, 2020), (Schubert; Awiszus; Rosenhahn, 2021)	*Embedding* de níveis	(Awiszus; Schubert; Rosenhahn, 2020), (Schubert; Awiszus; Rosenhahn, 2021)	Fidelidade	(Ferber et al., 2024)
	Métrica de cobertura	(Wang; Liu; Yannakakis, 2021)	Tile Pattern KL-divergence	(Awiszus; Schubert; Rosenhahn, 2020), (Schubert; Awiszus; Rosenhahn, 2021), (Awiszus; Schubert; Rosenhahn, 2021)		
	Momentum médio	(Wang; Liu; Yannakakis, 2021)	Distância de Levenshtein	(Awiszus; Schubert; Rosenhahn, 2021)		
	Frequência de tiro	(Wang; Liu; Yannakakis, 2021)	Diversidade	(Ferber et al., 2024)		

Além da revisão das métricas de avaliação, também foi possível identificar bases de dados utilizadas na literatura para o treinamento de GANs na geração de níveis de jogos. A principal base de dados encontrada nos estudos, o *Video Game Level Corpus*[1] (VGLC), reúne diversos jogos clássicos, com diferentes formas de representação, como matrizes de *tiles*, *heightmaps* e grafos, dependendo do jogo. A Tabela 5 apresenta a quantidade de níveis disponíveis para cada jogo, considerando somente os títulos mais utilizados nos estudos. Observa-se que, geralmente, o número de níveis é bastante limitado, o que pode impactar negativamente o aprendizado de modelos de DL, dificultando a generalização e a qualidade da geração.

Tabela 5 – Quantidade de níveis disponíveis em cada jogo do repositório TheVGLC.

Jogo	**Quantidade de Níveis**
Doom	36
Kid Icarus	6
Lode Runner	150
Mega Man	10
Super Mario Bros.	15
The Legend of Zelda	18

2.4 Considerações sobre o Capítulo

A realização da revisão da literatura e o levantamento dos trabalhos relacionados permitiram a identificação dos métodos e das técnicas atualmente utilizadas para geração de mapas procedurais, bem como as métricas para avaliação destes métodos e os conjuntos de dados utilizados. A partir da revisão do estado da arte também foi possível verificar que os trabalhos utilizando variações da arquitetura cGAN apresentam melhores resultados na aprendizagem, embora estas abordagens apresentem limitações nos resultados apresentados, principalmente quando aplicadas com um conjunto muito pequeno de dados. A utilização de um mecanismo de *bootstrap* pode ser um direcionamento viável para minimizar a dificuldade de poucos exemplos. A dificuldade de validar os experimentos também é um ponto-chave que deve se ter maior atenção. A abordagem por heurística pode ser utilizada não somente para validar os resultados, mas pode auxiliar no ajuste da rede durante o processo de treinamento.

Este Capítulo apresentou conceitos fundamentais para a compreensão do trabalho. Dentre os principais, foram apontados algoritmos de aprendizado de máquina para a geração procedural de níveis, com destaque para a utilização de GANs. Em seguida, foram apresentados e discutidos o funcionamento de diferentes modelos de GANs, tais como: GAN, cGAN, DCGAN, WGAN e WGANGP. No próximo Capítulo, serão apresentados os experimentos conduzidos para a geração de mapas procedurais,

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

detalhando a abordagem proposta a partir da fundamentação teórica e da análise do estado da arte. Cada experimento será introduzido com seu problema, objetivo e solução, complementando os anteriores e construindo progressivamente a metodologia e os resultados obtidos.

3 ABORDAGEM PROPOSTA PARA GERAÇÃO DE NÍVEIS JOGÁVEIS

O trabalho desta Tese faz parte do projeto em desenvolvimento pelo Grupo de Pesquisas em Jogos Digitais da Universidade Federal de Pelotas (UFPel), que busca soluções para desafios relacionados à geração procedural, customização/controle da geração de níveis e recomendação de níveis personalizados, entre outros aspectos. O grupo de pesquisa já conta com um jogo desenvolvido, originado a partir do trabalho de Padilha (2022). Os algoritmos desenvolvidos como parte da abordagem proposta nesta Tese foram testados e avaliados com base nos dados sintetizados a partir desse jogo, sendo também comparados com outros algoritmos para avaliação de resultados.

Figura 3 – *Gameplay* do jogo desenvolvido por Padilha (2022). O jogador inicia em um mapa estático de um vilarejo (cima), ao entrar no portal roxo este é transportado para uma *dungeon* aleatória (esquerda). A visualização do mapa para o jogador é parcial, sendo possível observá-la por completo somente em modo de edição (direita).

O jogo em questão é um RPG baseado em calabouços (*dungeons*), desenvolvido no motor de jogo Unity[1]. A jornada do jogador tem início em um vilarejo, onde é possível explorar o ambiente e interagir com os elementos dispostos no mapa. Como ilustrado na Figura 3, nesse cenário há um portal de cor roxa que transporta o jogador para um novo ambiente do tipo calabouço. Os calabouços são gerados proceduralmente, a fim de garantir diversidade nos níveis apresentados. Dessa forma, o jogador não consegue memorizar padrões fixos de progressão, tornando a experiência mais dinâmica e desafiadora. Essa abordagem evita a repetitividade e a monotonia, contribuindo para uma jogabilidade mais envolvente e imersiva.

Ao longo desta Tese, além dos dados obtidos a partir do jogo proposto por Padilha (2022), serão apresentadas outras estruturas de dados, selecionadas conforme a necessidade das investigações em cada experimento. Todas as bases utilizadas, métricas e técnicas aplicadas são detalhadas nas Seções seguintes.

3.1 Dados Sintetizados

Os dados sintetizados pelo jogo de Padilha (2022) são representados no formato de grade, composto por elementos de jogo chamados *tiles*. Os valores atribuídos à grade são inteiros distintos que representam a respectiva posição dos *tiles* e possuem representação visual conforme o mapeamento de cores detalhados na Tabela 6.

Tabela 6 – Representação dos *tiles* usados na geração de níveis do jogo desenvolvido por Padilha et al. (Padilha, 2022).

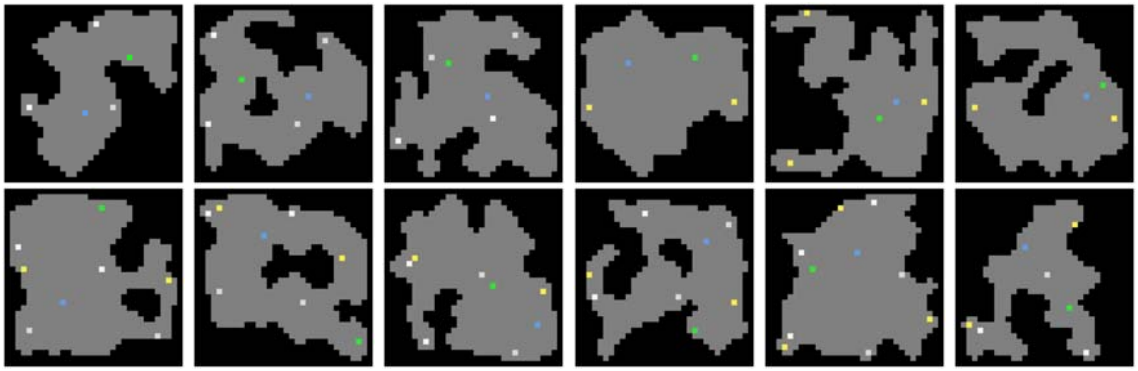
Tipo de dado	Valor	Cor
Chão	0	
Parede	1	
Jogador	2	
Portal	3	
Esqueleto	4	
Canhão	5	
Moeda	6	

Os mapas propostos por Padilha (2022) utilizam Autômatos Celulares (do inglês, *Cellular Automata* – CA) (Neumann; Burks, 1966) para a construção do terreno. A implementação possui um algoritmo pseudoaleatório para inserção e organização dos elementos de jogo – inimigos, moeda, portal e posição do jogador. Os mapas gerados obedecem às seguintes regras para serem considerados válidos: (1) um único elemento indicando a posição do portal; (2) um único elemento indicando a posição do jogador; e, (3) *tiles* do tipo chão devem ser acessíveis ao jogador.

[1] https://unity.com/

Os parâmetros utilizados no algoritmo pseudoaleatório de Padilha iniciam com valores padrões, mas podem ser alterados pelo próprio jogador. Estes valores ajustam o *layout* do mapa de acordo com: quantidade de inimigos no mapa; quantidade de moedas no mapa; espaçamento entre os inimigos; espaçamento entre as moedas; tamanho do mapa em comprimento e largura; e, topologia do mapa (aberto ou labirinto).

Figura 4 – Mapa de tamanho 32×32 gerado pelo jogo proposto por Padilha (2022), representado conforme a Tabela 6.



Para os experimentos, foram gerados mapas de tamanho 32×32 (Figura 4) para o conjunto de treino. Os mapas gerados são salvos em arquivo individual para serem manipulados fora do motor de jogo. Para a construção das estruturas de carregamento e processamento de dados, responsáveis pela geração procedural dos mapas, foi utilizado o *framework* PyTorch na sua versão 1.12, compatível com a versão 10.2 do *toolkit* NVIDIA CUDA e a biblioteca CuDNN em sua versão 7.6.

Para a organização dos dados, foi implementada a classe `Dataset`, onde é realizado o carregamento e pré-processamento dos mapas extraídos da Unity. Com apoio da biblioteca Numpy, os dados são recebidos – em uma única matriz com valores inteiros – e codificados por *one-hot enconding*. São aplicadas transformações de *flip* (vertical e horizontal) e rotação (somente nas orientações múltiplas de 90°) para aumentar a expressividade do conjunto de dados (Ping; Dingli, 2020).

Além dos dados fornecidos por Padilha (2022), foi implementado o algoritmo Drunkard's Walk (DW) para a geração de mapas, compondo um novo conjunto de dados. O uso dessa nova base visa avaliar a capacidade de generalização da abordagem proposta. O algoritmo DW foi escolhido por gerar padrões de terreno com comportamento semelhante aos dados originais, preservando a natureza orgânica e caótica das estruturas geradas. Além disso, para testar o modelo em diferentes tipos de saída da rede (baseado em *tiles* e *heightmaps*), foi adquirida uma base de dados complementar contendo imagens da Terra[2], representando variações reais de altitude em diversas regiões.

[2]https://www.kaggle.com/datasets/tpapp157/earth-terrain-height-and-segmentation-map-images

3.2 Métricas de Qualidade para Avaliação do Mapa

Visando mensurar, avaliar e validar as saídas geradas pelos diferentes algoritmos, desenvolvemos métricas de qualidade baseadas em heurísticas específicas. Entende-se por qualidade do mapa toda a composição e disposição dos elementos de jogo que garantam a jogabilidade, sendo estes destacados como regras de validação definidos na Seção 3.1.

3.2.1 Validade ou Jogabilidade

A validade ou jogabilidade avalia a capacidade de um nível ser jogável. Na literatura, muitos estudos utilizam agentes ou algoritmos de busca para avaliar a jogabilidade. No entanto, optou-se por não empregar esses mecanismos devido ao alto custo computacional necessário para avaliar inúmeras amostras geradas. Em contrapartida, outras pesquisas fazem o uso de heurísticas para avaliar o comportamento de um nível, especialmente no que diz respeito à frequência de determinados elementos no cenário.

Nessa Tese, foi escolhida a abordagem de heurísticas mais simples, que estabelecem as regras do comportamento esperado dos dados. Assim, foi possível utilizar essa medida para avaliar o desempenho das GANs durante a validação do treinamento das redes. A Equação 6 representa a heurística modelada para avaliar a validade de um mapa.

Considerando que o mapa é representado por uma matriz M, composta por e elementos de jogo (apresentados na Tabela 6), sua validade é determinada pelo atendimento a três critérios fundamentais. Os itens 1 e 2 referem-se à correta disposição dos elementos obrigatórios no mapa, enquanto o item 3 refere-se à coerência topológica do terreno.

$$M_e = \begin{cases} \text{itens 1 e 2:} & \sum M_e(i,j) = 1; e \in 2,3 \\ \\ \text{item 3:} & f(M_0) = 1 \end{cases} \tag{6}$$

Um mapa também é classificado como válido quando todas as células de chão formam um único componente conexo. A conexão é definida pela vizinhança ortogonal, desconsiderando vizinhos diagonais. A função $f(M_0)$ retorna a quantidade de caminhos em uma matriz bidimensional representada pelo canal de elementos do tipo chão, utilizando a função *label* do pacote *ndimage* pertencente à biblioteca *scipy*.

Para essa verificação, é aplicado um algoritmo de preenchimento de área para a contagem de blocos. A presença de um único bloco torna o mapa válido, enquanto a ausência ou a existência de múltiplos blocos o classifica como inválido. Além disso, busca-se manter o equilíbrio estrutural dos mapas, a fim de garantir que o número de

células do tipo “chão” seja predominante em relação ao de células do tipo “parede”. Portanto, foi adicionada a restrição de que a razão $\frac{n_{floor}}{n_{tiles}}$ (n_{floor} representa o número total de células chão e n_{tiles} o número total de células no mapa) deve estar no intervalo $[40\%, 70\%]$ para o mapa ser considerado válido.

3.2.2 Variabilidade

A obtenção de mapas distintos é um aspecto essencial no uso de algoritmos PCG. Na literatura, não há um padrão adotado para calcular a variabilidade. Neste trabalho, a variabilidade é calculada com base na diversidade da topologia do terreno em relação às diferentes amostras geradas.

Para mensurar a variabilidade entre os mapas gerados, é utilizada a Equação 7.

$$P_{25}(\sigma^2(M_{floor})) \tag{7}$$

Nessa equação, M_{floor} representa o canal de chão do tensor do mapa, e $\sigma^2(M_{floor})$ calcula a variância (ao nível de píxel) de 1.000 mapas, o que só é possível devido à natureza binária dos dados. Em vez de calcular a média da variância em todos os píxeis, é considerado o percentil 25^{th}, definido como P_{25}. Esse valor indica que 75% dos píxeis apresentam variabilidade superior ao resultado dessa equação.

3.3 Função de Custo

Duas abordagens de penalização que modificam a função de custo foram propostas, visando melhorar a qualidade e a diversidade dos mapas gerados. A primeira abordagem utiliza o treinamento com amostras válidas e inválidas, forçando a rede a aprender a distinguir configurações não aceitáveis e priorizando as configurações esperadas. A segunda leva em conta a variabilidade das amostras geradas para penalizar a rede, evitando a convergência para soluções pouco diversificadas.

3.3.1 Amostras Inválidas – *Invalid Sample* (IS)

Tradicionalmente, em GANs, o discriminador é treinado a distinguir entre dados reais e os dados sintetizados pelo gerador. Neste caso, além de aceitar mapas válidos como referência, o discriminador também recebe mapas inválidos, sendo treinado para rejeitá-los. Essa estratégia foi incorporada como um termo adicional na função de custo (terceiro termo da Equação 8), atuando como um mecanismo de penalização para amostras inválidas.

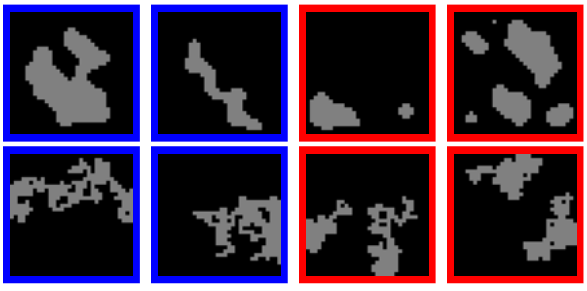
$$
\begin{aligned}
V(G,D) = {} & \mathbb{E}_{p_{\text{data}}(x)} \log D(x) + \mathbb{E}_{p_g(x)} \log(1 - D(G(z))) \\
& + \underbrace{\mathbb{E}_{p_g(x)} - \log D(G(z))}_{\text{penalização por IS}} \\
& + \underbrace{(1 - P_{25})}_{\text{penalização por VM}}
\end{aligned}
\tag{8}
$$

A função de custo proposta é baseada na divergência de Jensen–Shannon (Equação 8). A ideia de incorporar a penalização por IS é treinar o discriminador para rejeitar não apenas os dados sintéticos do gerador, mas também amostras que violam critérios estruturais, como a conectividade entre *tiles* de chão (presentes do conjunto de dados inválidos). Esse comportamento é modelado através da equação $\mathbb{E}_{p_g(x)} - \log D(G(z))$. Nessa penalização, $z \sim p(z)$ representa o espaço latente obtido a partir de uma distribuição Gaussiana, sendo o resultado de $G(z)$ a saída do gerador G a partir desse *seed*. O valor $logD(G(z))$ corresponde à probabilidade atribuída pelo discriminador D de que a amostra seja considerada real, através da entropia-cruzada. O logaritmo negativo transforma essa probabilidade em uma penalidade, onde a amostra identificada como inválida resulta em penalizações maiores para o gerador.

A penalização proposta pretende ampliar a perda do gerador, caso o gerador se aproxime dos dados inválidos. Como resultado prático, o gerador está sendo incentivado a evitar a produção de amostras que não satisfazem aos critérios estruturais do domínio, como a conectividade de *tiles*.

A Figura 5 apresenta algumas amostras, sintetizadas de cada conjunto de dados, utilizadas no processo de treinamento. As amostras válidas, destacadas em azul, são aquelas que atendem ao critério de conectividade, ou seja, apresentam uma estrutura de mapa considerada válida para o terreno. Por outro lado, as amostras inválidas, destacadas em vermelho, contêm *tiles* do tipo “chão” desconectados, que não satisfazem o critério estabelecido. O processo de preparação dos dados usado foi o mesmo descrito na Seção 3.1.

Figura 5 – Amostras utilizadas no processo de treinamento. Em azul estão as amostras válidas, ou seja, onde o critério de conectividade é atendido. Em vermelho, são as amostras inválidas, com a presença de vários blocos no mapa que não atendem aos critérios de conectividade.



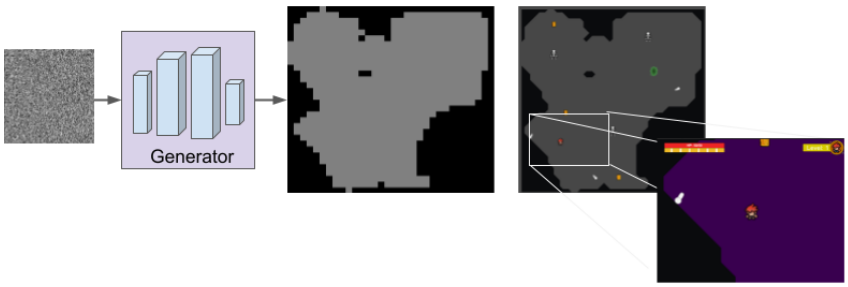
3.3.2 Medida de Variabilidade – *Variability Measure* (VM)

Outro termo adicionado à Equação 8 foi a VM, visando aprimorar a diversidade das amostras geradas. VM é calculada da mesma forma como descrito na Seção 3.2.2. Durante o treinamento, todas as amostras do lote são avaliadas, e sua variabilidade é medida para ser incorporada como um fator de penalização na função de custo. Quando o valor de variabilidade tende a zero, maior é o impacto da penalização, incentivando a geração de amostras mais diversas.

3.4 Metodologia dos Experimentos

Esta Seção apresenta uma visão geral dos experimentos realizados, detalhando a fundamentação das decisões tomadas ao longo do ajuste progressivo do modelo. A metodologia adotada explorou e integrou técnicas e estratégias de diversas áreas do conhecimento, aplicando-as ao campo dos jogos digitais, visando aprimorar cada etapa do treinamento de uma GAN voltada para a geração de níveis.

Figura 6 – Geração de mapas baseado em *tiles*. Em tempo de execução, um novo nível é gerado a partir de um espaço latente e, então, o nível pode ser executado diretamente em um jogo implementado a partir de uma estrutura de dados baseada em *tiles*.



Por fim, a rede geradora da arquitetura treinada é usada para a geração de níveis, transformando um vetor latente (*seed*) em um mapa completo, pronto para ser aplicado em jogos baseados em *tiles*, conforme ilustrado na Figura 6.

3.4.1 Experimento 1

O **Experimento 1** teve como objetivo compreender o comportamento das GANs e identificar os principais desafios associados ao seu treinamento. Para isso, foram definidas arquiteturas baseadas na literatura, e diferentes hiperparâmetros foram ajustados para avaliar seu impacto na qualidade das amostras geradas.

A primeira arquitetura testada foi a VanillaGAN (Tabela 7). As configurações do gerador desta arquitetura utilizam a função de ativação *tanh*, na camada de saída, e o otimizador SGD (sigla no inglês para *Stochastic Gradient Descent*), sendo o *learning rate* definido em 0,25 e *momentum* 0,5, valores-base da literatura. A rede neural foi treinada por 10.000 épocas com variações no *learning rate* do otimizador SGD.

Tabela 7 – Implementação da arquitetura VanillaGAN baseada em Goodfellow et al. (2014).

Camada	Discriminador		Gerador	
	Tipo	**in_features, out_features**	**Tipo**	**in_features, out_features**
1	Linear-LeakyReLU	(7168,512)	Linear-ReLU	(30,256)
2	Linear-LeakyReLU	(512,256)	Linear-BN-ReLU	(256,1024)
3	Linear-Sigmoid	(256,1)	Linear-Tanh	(1024, 7168)

A arquitetura é formada por camadas convolucionais, *batch normalization* e camadas de *upsample*, como observado na Tabela 8.

Tabela 8 – Implementação da arquitetura DCGAN baseada em Radford; Metz; Chintala (2016).

Camada	Discriminador		Gerador	
	Tipo	**n_filters, k_size, stride**	**Tipo**	**n_filters, k_size, stride**
1	Conv-LeakyReLU-Dropout	(7,16), (3,3), (2,2)	Conv-BN-ReLU-Upsample	(20,128), (3,3), (1,1)
2	Conv-LeakyReLU-Dropout-BN	(16,32), (3,3), (2,2)	Conv-BN-ReLU-Upsample	(128,128), (3,3), (1,1)
3	Conv-LeakyReLU-Dropout-BN	(32,64), (3,3), (2,2)	Conv-Tahn	(128,64) (3,3), (1,1)
4	Conv-LeakyReLU-Dropout-BN	(64,128), (3,3), (2,2)	Conv-Tahn	(64,7) (3,3), (1,1)
5	Linear-Sigmoid	(512,1)		

A segunda arquitetura testada foi a DCGAN, com adaptações na topologia da camada de entrada para a utilização adequada das camadas convolucionais 2D da arquitetura. A *seed* de entrada (ruído) foi alterada de um vetor para uma matriz. Dessa forma, é possível gerar tamanhos diferentes de mapa variando somente o tamanho da entrada. Radford; Metz; Chintala (2016) destacam a seguinte parametrização para a rede: utilização do otimizador Adam com *learning rate* 0,0002 e β_1 0,5.

O conjunto de dados para esse experimento foi composto por **somente 12 mapas**. A quantidade reduzida de amostras foi intencional para observar o comportamento das GANs no aprendizado com **um pequeno número de amostras**. Vale ressaltar que diversos trabalhos que utilizam a base de dados do repositório VGLC para a geração de mapas relatam dificuldades na obtenção de dados suficientes (Rodriguez Torrado et al., 2020; Awiszus; Schubert; Rosenhahn, 2020) e da necessidade de mais mapas diversos para garantir a aplicabilidade em jogos digitais.

3.4.2 Experimento 2

Diversas técnicas de regularização foram revisadas na literatura visando mitigar a instabilidade no treinamento de GANs. Algumas delas, como o uso de *dropout*, *batch normalization* e penalidades no gradiente, já vinham sendo utilizadas nos experimentos anteriores. No entanto, identificou-se na literatura a técnica de *Spectral Normalization* (SN), proposta por Miyato et al. (2018), como uma alternativa promissora para estabilizar o treinamento de GANs. Essa abordagem atua diretamente sobre as matrizes de pesos das camadas convolucionais, restringindo sua norma espectral. Ao limitar o maior valor singular dessas matrizes no discriminador, a SN impede que ele aprenda funções excessivamente sensíveis, contribuindo para a estabilidade do treinamento do gerador e discriminador.

Tabela 9 – Ajustes na arquitetura DCGAN.

Camada	Discriminador		Gerador	
	Tipo	**n_filters, k_size, stride**	**Tipo**	**n_filters, k_size, stride**
1	Conv-LeakyReLU-Dropout	(7,16), (3,3), (2,2)	Conv-BN-ReLU-Upsample	(20,64), (3,3), (1,1)
2	Conv-LeakyReLU-Dropout-BN	(16,32), (3,3), (2,2)	Conv-BN-ReLU-Upsample	(64,32), (3,3), (1,1)
3	Conv-LeakyReLU-Dropout-BN	(32,64), (3,3), (2,2)	Conv-Tahn	(32,7) (3,3), (1,1)
4	Linear-Sigmoid	(1024,1)		

O **Experimento 2** comparou as arquiteturas convolucionais sem e com a técnica de regularização SN para o cenário de geração de níveis. Nesse experimento, algumas camadas da arquitetura também foram ajustadas visando reduzir a quantidade de parâmetros, buscando melhorar a convergência durante o treinamento a partir de uma base de dados com poucas amostras. Os ajustes para esse experimento estão descritos na Tabela 9.

3.4.3 Experimento 3

O **Experimento 3** explorou ajustes na função de custo como estratégia para aprimorar o treinamento da GAN. Foram modelados e analisados dois fatores de penalização visando minimizar problemas como dificuldade de convergência e *mode-collapse* do gerador.

Foram propostas três modificações na arquitetura para aprimorar não somente a validade dos mapas gerados, mas também sua variabilidade: (1) penalização na função de custo de acordo com amostras inválidas anotadas no conjunto de dados; (2) novo termo na função de custo que penaliza o gerador quando o lote sintetizado apresenta baixa variabilidade; e, (3) técnica de *data augmentation* baseada em *bootstrapping* na qual mapas gerados durante o treinamento são reincorporados ao conjunto de dados para fortalecer o aprendizado e melhorar a diversidade das amostras. O objetivo dessas alterações foi para aprimorar o processo de treinamento da GAN, direcionando o aprendizado por meio da incorporação de heurísticas específicas.

A arquitetura utilizada neste experimento foi a DCGAN, com alterações na função de custo originalmente baseada na divergência de Jensen–Shannon. As modificações propostas estão detalhadas na Seção 3.3, que analisam comportamentos de variabilidade e validade das amostras. Ademais, foi utilizada a abordagem de *bootstrapping* como mecanismo para aumentar a expressividade do conjunto de dados de treinamento. Para isso, a estratégia utilizada para salvar o melhor modelo e avaliar os dados para incorporar ao conjunto de dados segue: (1) A cada época, o modelo é validado, e a melhor versão é salva/sobrescrita; (2) A cada 1.000 épocas de treinamento, o melhor modelo é carregado; (3) É gerado um lote de 1.000 amostras com o melhor modelo; e, (4) O lote é avaliado, sendo inserido ao conjunto de dados somente os dados considerados válidos.

Dados a partir da implementação dos algoritmos *Cellular Automata* (CA) e *Drunkard's Walk* (DW) foram sintetizados para o treinamento das GANs nos Experimentos

2 e 3. O conjunto de dados do CA foi composto por 346 mapas de tamanho 32×32, divididos em 173 mapas válidos e 173 inválidos. Já o conjunto de dados do DW consistiu em 1.000 mapas do mesmo tamanho, contendo 575 mapas válidos e 425 inválidos.

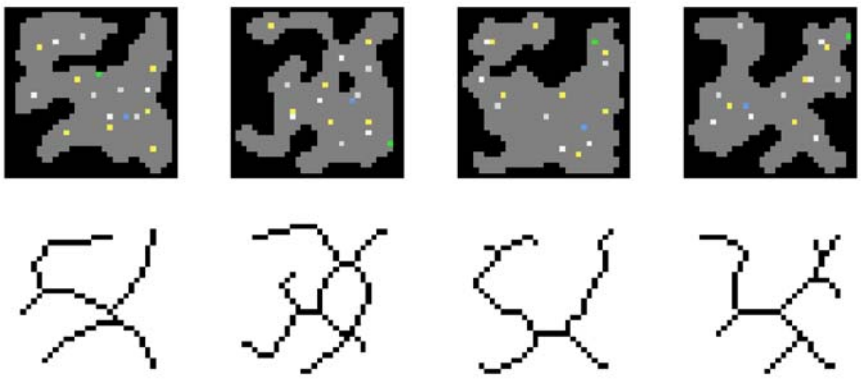
A partir desses experimentos, foi utilizada a plataforma Weights & Biases (wandb) para realizar uma varredura sistemática de hiperparâmetros por meio da otimização Bayesiana. Foram incluídos na busca parâmetros como o *learning rate*, o tamanho do lote de treinamento e a quantidade de canais da matriz representada como espaço latente. A avaliação do Experimento 1 foi guiada pelas métricas de validade do critério de conectividade de chão, critério de distribuição de elementos e a combinação de ambos os critérios. Enquanto para o Experimento 3 foram usadas as métricas de validade para conectividade e a variabilidade entre amostras.

3.4.4 Experimento 4

O **Experimento 4** buscou mitigar a sensibilidade da rede em relação à métrica de caminhos conectados, utilizando um esboço (*sketch*) como entrada condicional para guiar a geração. Nesse experimento, foram testadas duas bases de dados: mapas baseados em *tiles* e mapas baseados em *heightmaps*.

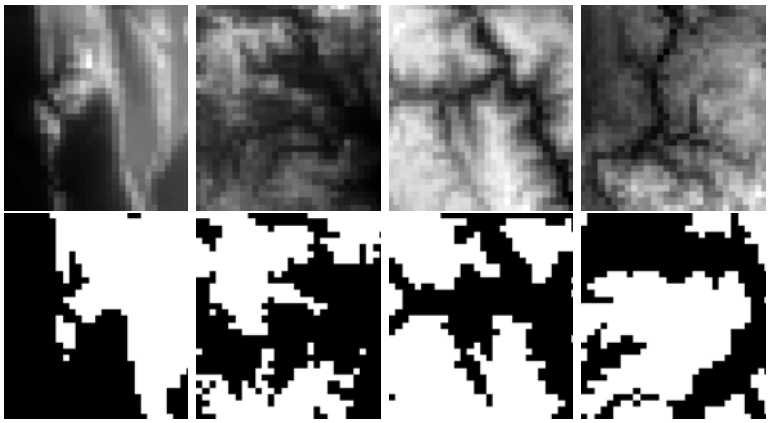
No treinamento com os dados de Padilha (2022), baseados em *tiles*, foi adotado um procedimento específico para a extração dos *sketches* a partir das amostras. Inicialmente, é extraído o canal correspondente às paredes do nível, o qual é então invertido, de modo que as regiões caminháveis (ou seja, os *tiles* de chão) passem a compor o foco da análise. Em seguida, aplica-se o algoritmo de *Morphological Thinning* sobre essa máscara invertida, resultando em uma representação esquelética dos caminhos disponíveis no mapa. Esse *sketch* final é utilizado como entrada condicional para a geração de novos níveis.

Figura 7 – Pré-processamento da entrada condicional na geração de níveis baseado em *tiles*. Amostra real do conjunto de dados (cima), usada para gerar o *sketch*; *sketch* pré-processado (baixo).



Já na base de dados de relevo, obtidos através do Kaggle, foi adotado um procedimento diferente para garantir que cada amostra possua um *sketch* correspondente. Primeiro, calcula-se a média da altura das amostras. Regiões com valores abaixo dessa média são classificadas como terreno caminhável. A partir dessa segmentação, é gerada uma máscara binária que serve como entrada condicional para o modelo (Figura 8). Para avaliar o potencial de aprendizado da GAN com um número reduzido de amostras, o treinamento foi realizado utilizando somente 100 exemplos do conjunto de dados de tamanho 32×32. Para fins de comparação, amostras não apresentadas à rede foram utilizadas na avaliação.

Figura 8 – Pré-processamento da entrada condicional na geração de *heightmaps*. Amostra real do conjunto de dados (cima), usada para gerar o *sketch*; *sketch* pré-processado (baixo).



4 RESULTADOS

Este Capítulo apresenta os achados obtidos a partir dos experimentos conduzidos para a geração de níveis utilizando GANs. As observações dos Experimentos 2 e 3 foram publicadas em Silva et al. (2023) e Silva; Torchelsen; De aguiar (2024), respectivamente. Mais detalhes de como os Experimentos foram conduzidos, principalmente no que se refere ao ajuste manual dos hiperparâmetros, podem ser vistos no Apêndice A.

4.1 Ajuste de Hiperparâmetros

A partir dos experimentos preliminares para a geração de níveis utilizando arquiteturas baseadas em GANs, observou-se a necessidade de ajustes nos hiperparâmetros específicos de cada modelo encontrado na literatura. Nos primeiros testes, visando compreender o comportamento do treinamento das arquiteturas, foram inicialmente utilizados os valores de hiperparâmetros recomendados nos estudos originais. Em seguida, iniciou-se o processo de ajuste dos principais parâmetros, como o *learning rate*, o tamanho do lote de treinamento e a quantidade de canais na matriz, ou tamanho do vetor, de espaço latente.

É importante destacar que modificações na arquitetura dos modelos, na função de custo ou mesmo no tamanho do conjunto de dados podem impactar diretamente na configuração ideal dos hiperparâmetros. Essa dependência torna o ajuste manual uma tarefa onerosa e demorada, além de suscetível a inconsistências.

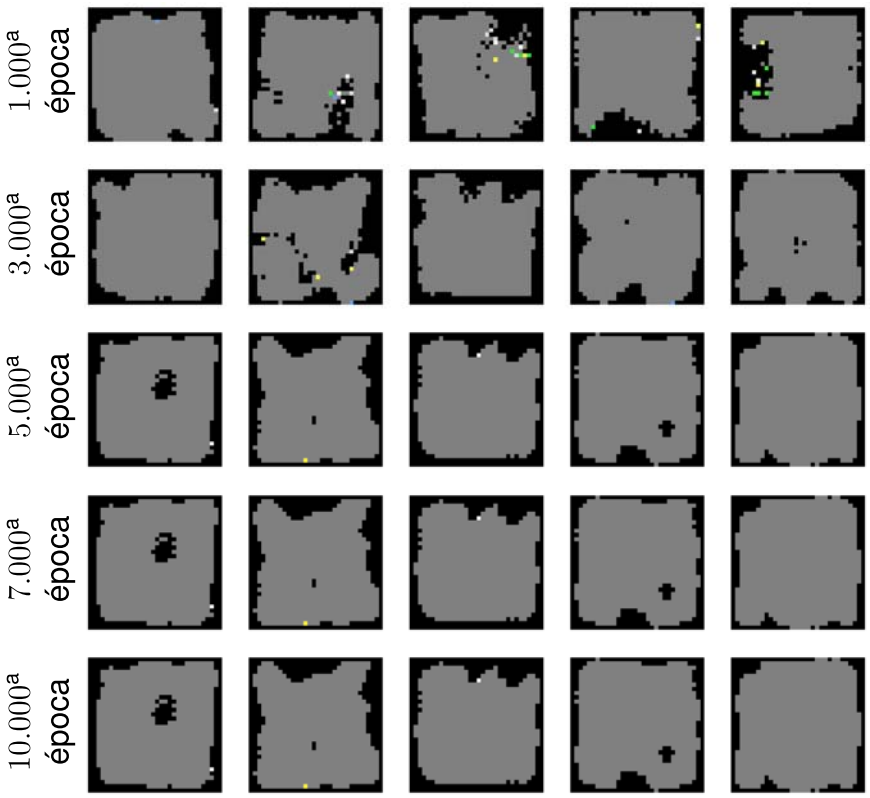
Para otimizar esse processo, a partir do Experimento 3 foi incorporada a plataforma Weights & Biases (wandb), possibilitando a realização de uma busca sistemática de hiperparâmetros utilizando a otimização Bayesiana. Essa abordagem visou automatizar e acelerar a identificação de configurações mais adequadas para cada cenário de experimentação.

4.2 Escolha da Arquitetura

Para a escolha da arquitetura, foram realizados diversos experimentos para primeiro entender como o modelo aprendia, para então definir qual modelo melhor ajustava aos dados usados.

Observou-se que a VanillaGAN apresentou desempenho insatisfatório mesmo após muitas épocas de treinamento, como pode ser visto na Figura 9. A rede rapidamente entrou em estado de estagnação, não demonstrando aprendizado significativo após 3.000 épocas.

Figura 9 – Visualização das saídas do gerador em cada época de treinamento na arquitetura VanillaGAN.



Além disso, os mapas gerados apresentaram baixa variabilidade (Figura 10), sendo compostos por grandes blocos contínuos de terreno e com elementos de jogo dispostos predominantemente próximos às bordas, o que compromete a jogabilidade. O problema de *mode collapse* foi recorrente, o que evidencia a limitação dessa arquitetura em cenários com baixo volume de dados.

Por outro lado, a arquitetura DCGAN demonstrou um desempenho mais consistente em gerar representações diversa. A Figura 11 ilustra a diversidade de terrenos gerados, com corredores bem definidos, paredes internas e melhor posicionamento dos elementos de jogo – características que favorecem a jogabilidade.

Mesmo que a taxa de níveis válidos tenha sido numericamente próxima à do modelo ajustado com SGD (com 4,7% usando RMSprop e 4,1% com Adam, conforme

Figura 10 – Nível gerado pela arquitetura VanillaGAN. Observa-se baixa variabilidade, com uma grande massa de terreno, ausência de corredores e elementos posicionados próximos às paredes.

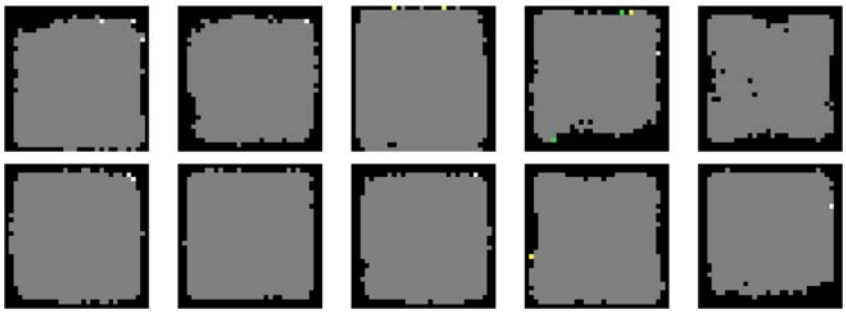


Figura 11 – Resultados válidos gerados a partir da arquitetura DCGAN, treinada por 2,000 épocas. A primeira linha apresenta resultados da rede na configuração recomendada por Radford; Metz; Chintala (2016). A segunda linha os resultados demonstrados são com base nas configurações de Ping; Dingli (2020).

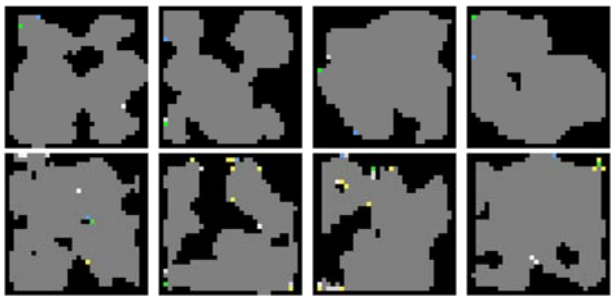
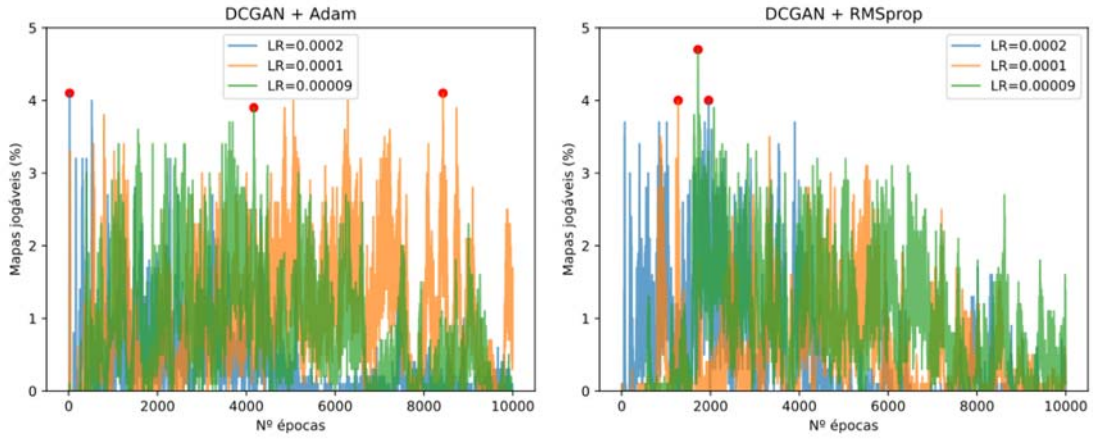


Figura 12), a qualidade estrutural e a variedade dos mapas gerados pela DCGAN foram visivelmente superiores. Além disso, a rede se mostrou mais adaptável a diferentes configurações de hiperparâmetros e apresentou menor propensão ao *mode collapse*.

Figura 12 – Comparação de diferentes configurações de *learning rate* sob os otimizadores Adam e RMSprop. As linhas em vermelho destacam o intervalo que se obteve maior percentual de mapas válidos, sendo destacado, em círculo vermelho, os picos de cada configuração.



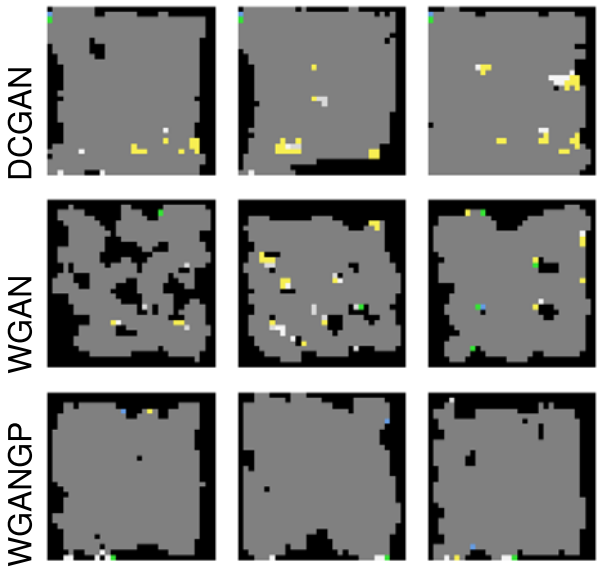
A diferença entre as arquiteturas DCGAN e WGAN/WGANGP está no tipo de função de custo, a primeira baseada na divergência de Jensen–Shannon e a última ba-

seada na distância de Wasserstein. Ademais, a WGANGP utiliza um mecanismo de restrição 1-Lipschitz para estabilizar o treinamento, mais detalhes na Seção 2.2.4. Mesmo com a base teórica para estabilidade de treinamento, a WGANGP não teve sucesso para o problema apresentado. Após investigação, observou-se que este termo de penalidade não se adapta bem a dados discretos ou categóricos. Devido às mudanças abruptas de valores em cada *tile* do nível (domínio de dados), a interpolação das entradas do discriminador resulta em valores não categóricos, fazendo com que a aproximação dos gradientes seja muito próxima de 0, o que dificulta a aplicação adequada da penalização.

O número de níveis válidos foi baixo quando consideradas as métricas de jogabilidade, ou seja, a análise do dado completo, com canais de terreno e demais elementos. No entanto, a análise isolada dos dados de terreno (*tiles* de chão e parede) revelou um bom desempenho na geração de terrenos válidos.

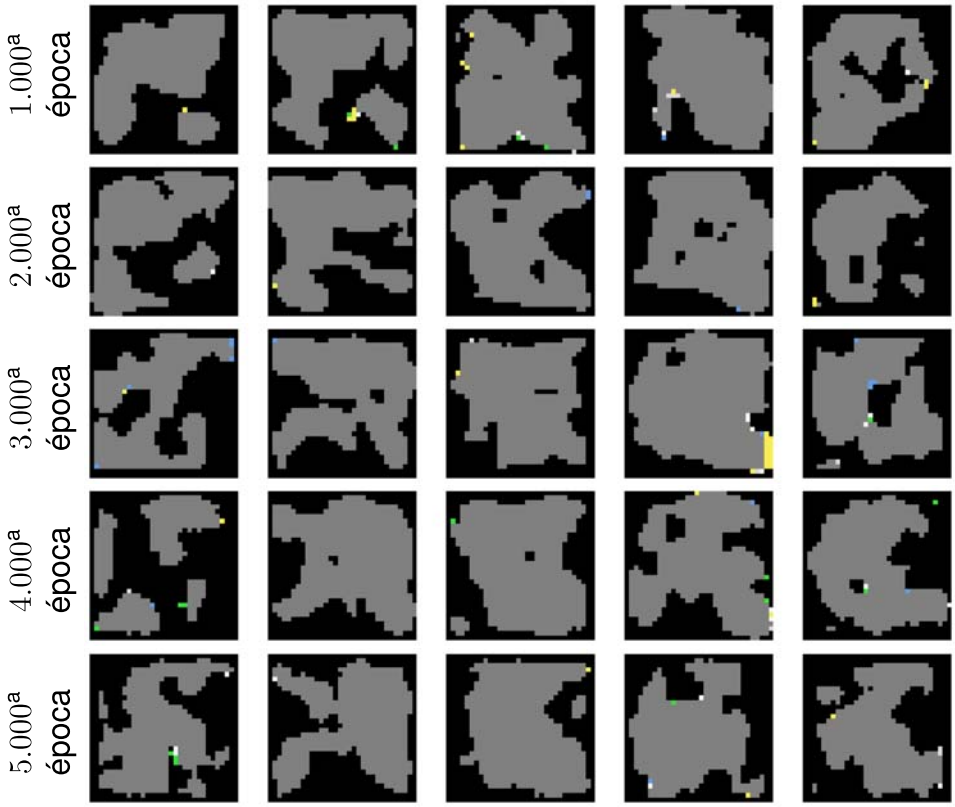
Em menos de 1.000 épocas, alcançaram valores entre 80% e 90% para terrenos válidos. Observou-se nas gerações da VanillaGAN que os terrenos tendem a ser mais largos (como mostrado na Figura 10), enquanto em arquiteturas convolucionais é possível observar algumas ocorrências de corredores (Figura 13).

Figura 13 – Nível gerado por cada arquitetura de rede convolucional. Observam-se alguns corredores nas gerações utilizando as arquiteturas DCGAN e WGAN, enquanto as gerações com a arquitetura WGANGP apresentam baixa variabilidade na disposição do terreno.



Diante dessas observações, pode-se concluir que a DCGAN é a arquitetura mais adequada para o problema proposto, por conseguir equilibrar a geração de mapas válidos e diversos, mesmo sob um conjunto de dados limitado. A evolução visual durante o treinamento (Figura 14) e os exemplos de níveis válidos (Figura 11) reforçam essa conclusão, demonstrando que a arquitetura é mais estável e capaz de capturar as regras estruturais necessárias para a jogabilidade dos níveis.

Figura 14 – Visualização da evolução das amostras durante treinamento da DCGAN. Cada coluna representa um valor diferente do espaço latente como entrada da rede. É possível observar que para uma mesma entrada o resultado converge e diverge durante o treinamento da rede, sendo observado na primeira coluna a variação da topologia do terreno.



4.3 Técnicas de Regularização

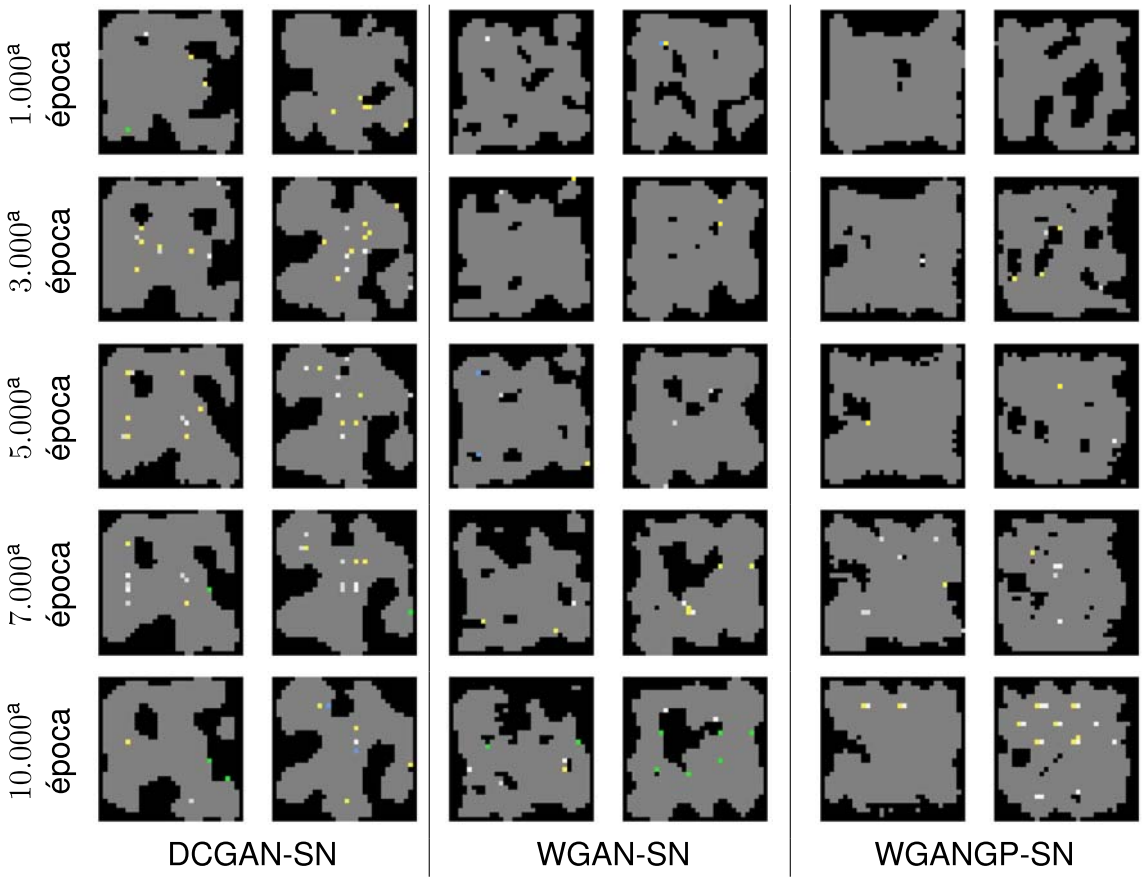
Observou-se durante os experimentos que o uso das camadas SN resultou em um atraso na convergência dos modelos, sendo a convergência ditada pelas métricas de validade do terreno e demais elementos. Por exemplo, as arquiteturas DCGAN-SN, WGAN-SN e WGANGP-SN levaram 9.407, 3.555 e 4.185 épocas, respectivamente, para alcançar os valores apresentados na Tabela 10. Mesmo com a necessidade de mais iterações durante o treinamento, comparado com as versões que utilizam BN, isso não representou necessariamente uma desvantagem. Na prática, as redes com SN demonstraram maior estabilidade ao longo do tempo e continuaram a se ajustar de forma consistente à medida que o treinamento progredia.

Em uma avaliação empírica, os níveis gerados foram semelhantes aos dados-alvo. Observou-se que as arquiteturas de GANs com SN são menos suscetíveis ao *mode-collapse* (quando comparadas ao uso de BNs), conforme mostrado na Figura 15. Outro problema encontrado na arquitetura com BN foi que os elementos tendiam a permanecer próximos às paredes. As amostras geradas a partir do treinamento das GANs com SN apresentaram os elementos mais distribuídos esparsamente pelo terreno. Acredita-se que o SN lide melhor com a distribuição dos elementos, ou seja,

Tabela 10 – Comparação da jogabilidade entre diferentes métodos de regularização para cada arquitetura com 10.000 épocas. Resultados obtidos para saídas de tamanho 32×32.

Arquitetura	Regularização	Nível válido (%)
DCGAN	BatchNormalization	*mode-collapse*
	SpectralNormalization	**13.9**
WGAN	BatchNormalization	7.4
	SpectralNormalization	**11.4**
WGANGP	BatchNormalization	*mode-collapse*
	SpectralNormalization	**2.9**

Figura 15 – Visualização das saídas do gerador em cada época de treinamento nas arquiteturas VanillaGAN-SN, DCGAN-SN, WGAN-SN e WGANGP-SN. Observam-se melhores resultados na disposição do terreno e no posicionamento dos elementos.



fazendo com que canais com menos ocorrências (jogador, portal, inimigos e moedas) tenham a mesma importância que os canais de chão e parede.

Os experimentos mostraram o desempenho das GANs treinadas com as funções de custo baseadas na divergência de Jensen–Shannon e na distância de Wasserstein, além de técnicas de regularização. Embora o treinamento com diferentes arquiteturas convolucionais e funções de perda tenha tido pouco impacto na geração de níveis, **o uso de *Spectral Normalization* não somente estabilizou o treinamento, mas também melhorou significativamente a métrica de jogabilidade dos resultados**

das GANs.

Os resultados obtidos nos experimentos indicam que a utilização da técnica de SN contribui positivamente para o treinamento do modelo, favorecendo a estabilidade e a convergência. Essa evidência está associada à capacidade da técnica de limitar a variação dos gradientes entre as camadas convolucionais do discriminador/crítico, o que sugere um comportamento mais controlado durante o processo de treinamento.

4.4 Função de Custo

Um ponto importante a ser destacado é quanto ao potencial de generalização da rede. Dispor os elementos de jogo (jogador, portal, inimigos e moedas) adequadamente é mais desafiador para o modelo gerar, do que criar a topologia do terreno nos níveis. Assume-se que a dificuldade em aprender a disposição de alguns elementos se deve a um desbalanceamento nos dados, causado pela diferença entre a quantidade de dados de terreno (chão e paredes) e de elementos (jogador, portal, inimigos e moedas). Esse desequilíbrio pôde ser minimizado, mas ainda não resolvido satisfatoriamente, por meio da normalização proposta.

Os achados anteriores indicam que o problema ainda pode ser ajustado ou que os critérios de validade precisam ser adaptados, a fim de não penalizar excessivamente a avaliação dos níveis. O que leva a considerar estratégias que forcem o treinamento a observar comportamentos específicos nos dados ou uma estratégia para aumentar a expressividade do conjunto de dados para treinamento.

Os resultados obtidos a partir do experimento com a função de custo modificada, com apoio de heurísticas, evidenciam que a GAN captura efetivamente a distribuição dos dados para os resultados do CA (observados na Figura 16) e para os resultados do DW (observados na Figura 17).

Ao comparar os dados gerados por cada configuração de treino, observa-se que as amostras se assemelham à estética dos dados originais associados a cada algoritmo. Apesar do alto número de épocas, o treinamento utilizando o conjunto de dados CA convergiu para uma alta taxa de mapas válidos em menos de 2.000 épocas. Por outro lado, o treinamento utilizando o conjunto de dados DW enfrentou mais desafios para aprender o critério de conectividade dos dados.

A abordagem proposta permite treinar um gerador que sintetiza mapas com uma boa taxa de mapas válidos em um tempo muito mais rápido do que outros métodos (Tabela 11). No experimento CA, foi possível obter 1,5$\times$ mais mapas válidos em comparação com a DCGAN tradicional (*baseline*) e 2,4$\times$ mais em comparação com o CA, utilizando amostras inválidas como mecanismo de penalização para o gerador.

Por outro lado, o potencial de variabilidade diminuiu cerca de 1,3$\times$ em comparação com o *baseline*. Para equilibrar esse *trade-off*, o valor da métrica de variabilidade foi

Figura 16 – Comparação entre as abordagens CA e DCGAN (*baseline* e abordagem proposta) para gerar mapas de masmorras. É possível observar uma grande semelhança entre elas, com algumas áreas abertas e a presença de corredores.

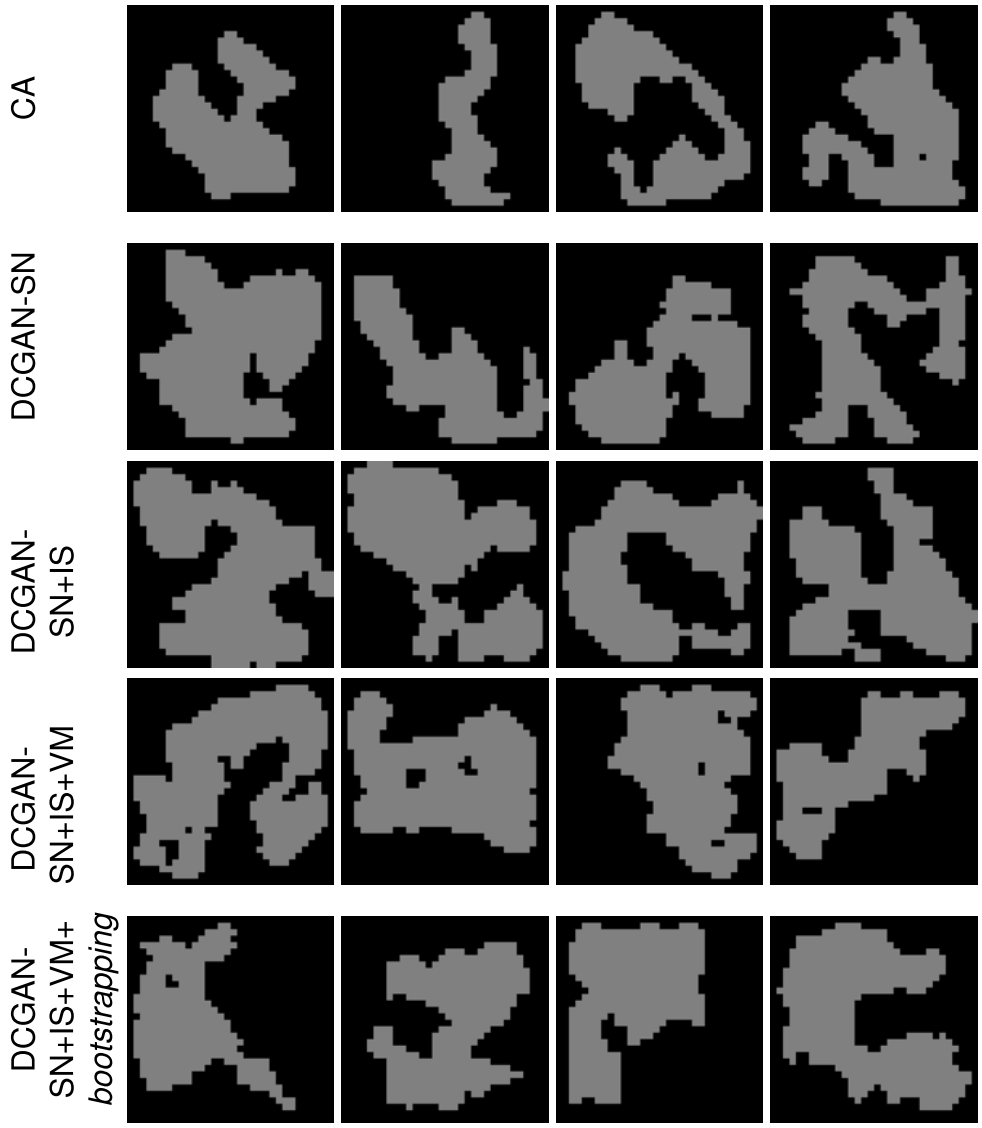
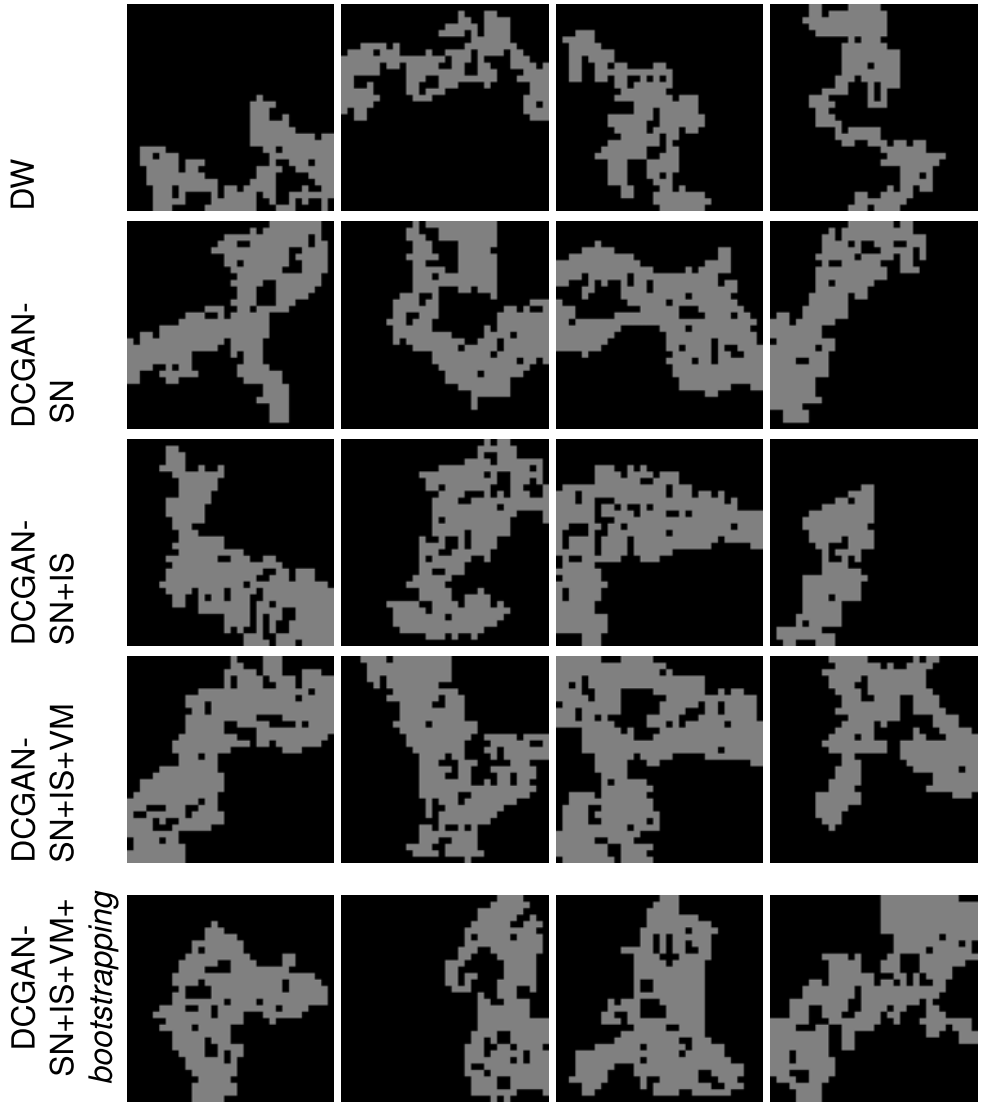


Tabela 11 – Avaliação de mapas sintéticos com resolução 32×32. Resultados do modelo DCGAN-SN treinado somente com amostras 32×32.

Algoritmo	**Validade**	**Variabilidade**	**Tempo de inferência (s)**
CA	17.6 %	0.378	$4.7 * 10^{-3}$
DCGAN-SN	26.8 %	0.304	$5.7 * 10^{-7}$
DCGAN-SN + IS (nosso)	42.8 %	0.230	$2.0 * 10^{-6}$
DCGAN-SN + IS + VM (nosso)	41.8 %	**0.395**	$7.6 * 10^{-6}$
DCGAN-SN + IS + VM (nosso) + *bootstraping*	**58.6** %	0.373	$5.0 * 10^{-6}$
DW	**57.5** %	**0.378**	$1.1 * 10^{-3}$
DCGAN-SN	11.0%	**0.376**	$1.0 * 10^{-6}$
DCGAN-SN + IS (nosso)	17.4%	0.351	$2.0 * 10^{-6}$
DCGAN-SN + IS + VM (nosso)	9.0%	0.366	$1.0 * 10^{-6}$
DCGAN-SN + IS + VM (nosso) + *bootstraping*	13.9%	0.357	$2.0 * 10^{-6}$

Figura 17 – Comparação entre as abordagens DW e DCGAN (*baseline* e abordagem proposta) para gerar mapas de masmorras. Observa-se uma grande semelhança entre elas, com algumas áreas abertas e a presença de corredores.



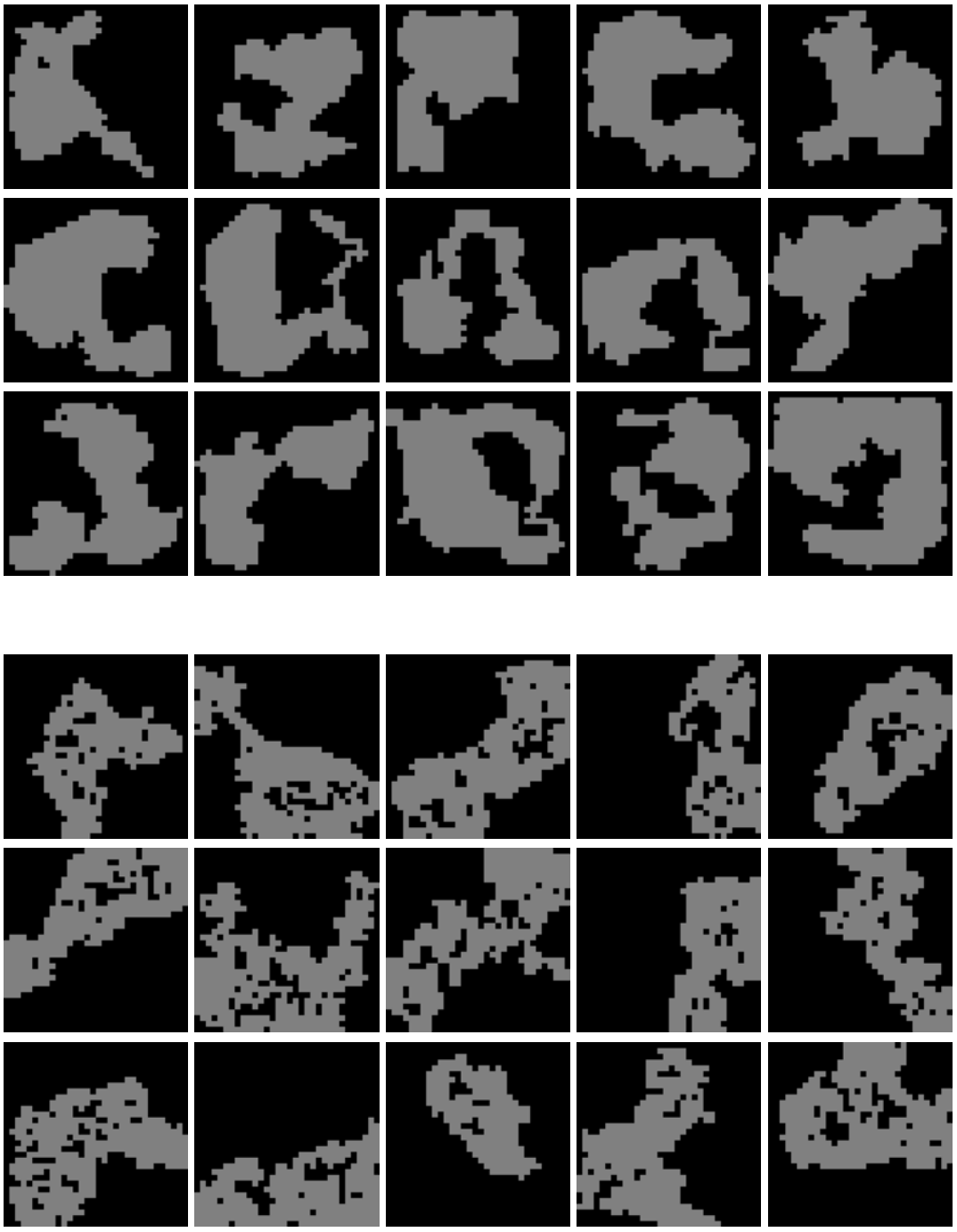
injetado na função de perda. Assim, foi obtida uma perda mínima de 1% de mapas válidos para superar a variabilidade dos métodos apresentados.

Após obter boas amostras com as abordagens anteriores, foi testada a proposta de Rodriguez Torrado et al. (2020), com a abordagem de *bootstrapping*, como um mecanismo de alimentação recorrente para obter mais dados durante o treinamento. Usando essa técnica, obtiveram-se 58,6% de mapas válidos, com uma variabilidade semelhante à do CA. Esses resultados só foram alcançados após ajustes para implementação do *bootstrapping*.

Ao treinar com o conjunto de dados DW, não foi possível alcançar um número elevado de mapas válidos. No entanto, foi possível melhorar o tempo de geração dos mapas. É importante ressaltar que há uma perda mínima de variabilidade nas gerações. Embora isso possa resultar em saídas de mapas ligeiramente menos diversas, o

trade-off é justificado pela economia significativa de tempo de geração e pela consistência (dos mapas válidos). Essa abordagem encontra um equilíbrio entre velocidade e qualidade, tornando-se uma escolha atraente para situações em que eficiência, confiabilidade na geração e escalabilidade são desejadas.

Figura 18 – Resultados do treinamento da arquitetura DCGAN-SN + IS + VM + *bootstrapping* para os conjuntos de dados CA (topo) e DW (baixo). Em ambos os casos, é possível notar que os dados gerados se assemelham à estética dos dados originais e conseguem gerar novos mapas válidos a partir deles.



A abordagem proposta alcançou uma alta taxa de mapas válidos e variados para o conjunto de dados CA (Figura 18), evitando o problema de *mode-collapse*. Além disso,

o modelo conseguiu **aprender a distribuição do domínio do mapa**, permitindo gerar uma porcentagem significativa de mapas válidos em resoluções mais altas nas quais a rede foi treinada.

É importante enfatizar que um modelo generativo deve considerar as características intrínsecas dos mapas, o que muitas vezes pode ser um desafio para as GANs convergirem. Para lidar com isso, o treinamento foi projetado de forma diferente, oferecendo amostras consideradas inválidas para a geração durante o processo de treinamento.

Mais amostras válidas permitiram bons resultados com a técnica de *bootstrapping*. No entanto, é fundamental conhecer as regras de elegibilidade para determinar se um nível é válido antes de adicioná-lo ao *corpus* de treinamento. Caso contrário, o mecanismo pode comprometer gradualmente o comportamento dos dados originais. Outro destaque da abordagem foi o uso do termo de variabilidade, que incentivou o modelo a gerar resultados mais variados.

4.5 Generalização do Modelo

Uma das vantagens de arquiteturas convolucionais é a sua flexibilidade e adaptação a diferentes formatos de entrada, permitindo que o gerador seja utilizado para criar níveis com diferentes dimensões. Isso é possível mediante o ajuste dos vetores latentes, que podem ser adaptados para gerar saídas em resoluções variadas.

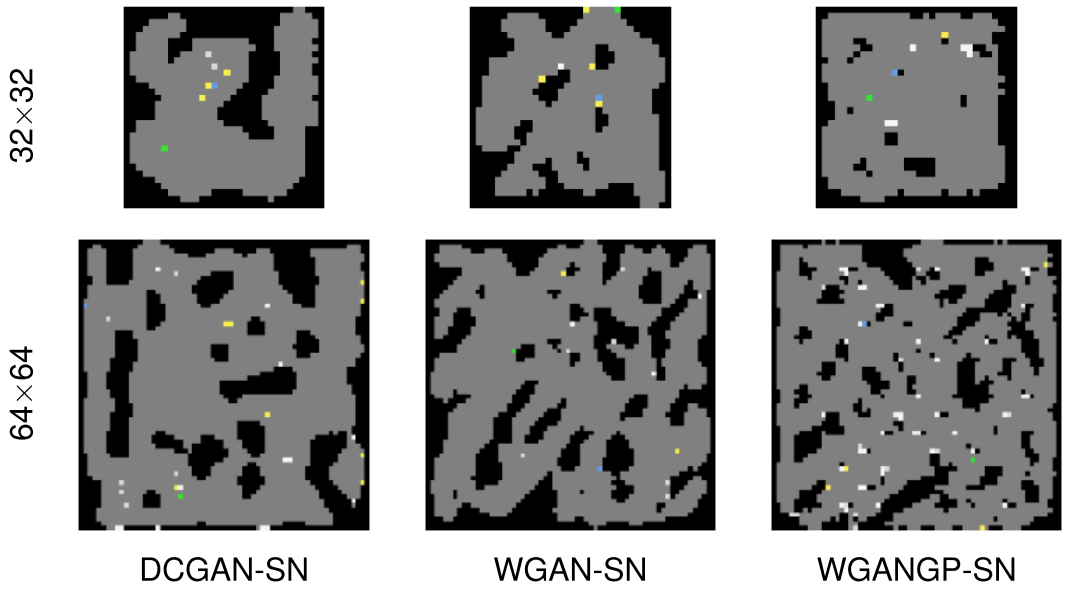
A partir do modelo treinado utilizando SN e amostras com resolução de 32×32, foi utilizado esse gerador para gerar níveis de 64×64. Embora os resultados não tenham sido ótimos, como mostra a Tabela 12, o modelo conseguiu gerar níveis com uma boa distribuição nos elementos de jogo. A Figura 19 ilustra os resultados desse experimento, comparando-os com os dados obtidos para a resolução de 32×32. Quando analisado isoladamente apenas os dados de terreno, percebe-se que os modelos aprenderam rapidamente a produzir topologias de terreno consideradas válidas, semelhante aos achados descritos na Seção 4.2.

Tabela 12 – Comparação da jogabilidade entre diferentes métodos de regularização para cada arquitetura, com 10.000 épocas de treinamento. Os resultados foram obtidos utilizando um modelo treinado com níveis de tamanho 32×32.

Arquitetura	**Níveis válidos (%)**
DCGAN-SN	0,4
WGAN-SN	0,9
WGANGP-SN	0,1

Em outro experimento, treinado a partir da abordagem proposta, com o conjunto de dados CA, o gerador treinado exclusivamente com amostras de 32×32 conseguiu

Figura 19 – Níveis gerados em diferentes resoluções por cada arquitetura de rede convolucional utilizando SN.

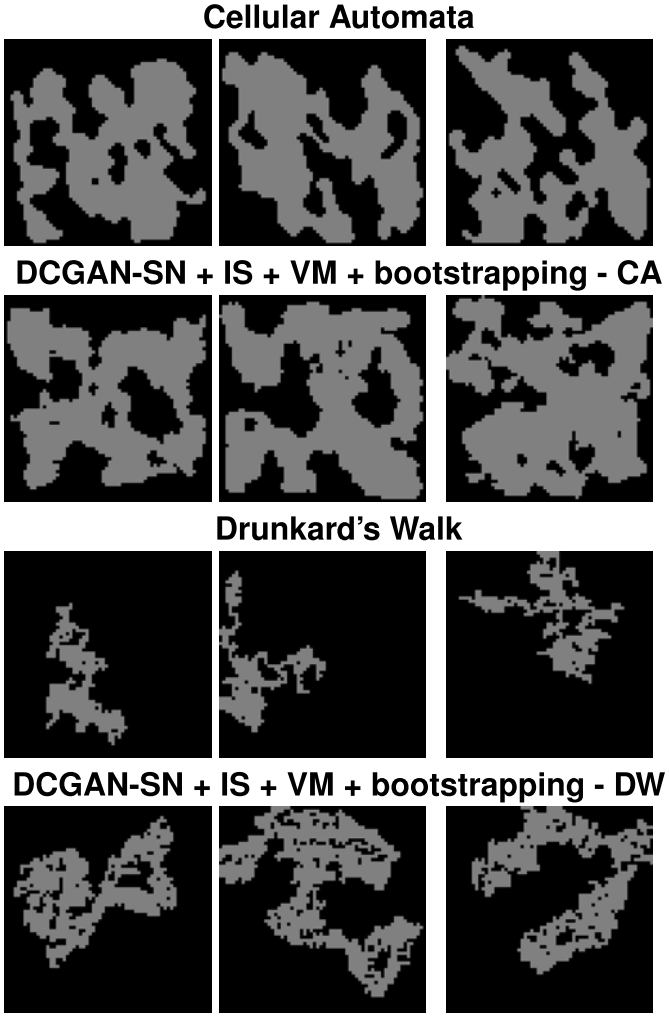


gerar 10,1% de mapas válidos na resolução 64×64, demonstrando seu potencial para generalizar os dados aprendidos. A Figura 20 mostra os resultados para esta resolução, comparados aos resultados obtidos com o conjunto de dados CA, que alcançou somente 0,7% de mapas válidos nesta resolução.

Como mencionado anteriormente, a rede enfrentou grandes desafios para aprender a restrição de conectividade dos dados DW, alcançando uma taxa muito baixa de mapas válidos quando gerados na resolução mais alta com a qual o modelo foi treinado. Para comparar a estética dos resultados gerados pela GAN com os resultados do DW, foram geradas amostras de 64×64 usando 4 agentes, o dobro do número de agentes nos dados da resolução 32×32. A abordagem foi aplicada no jogo proposto por Padilha (2022) para visualizar a topologia do mapa com o algoritmo de disposição de elementos proposto neste trabalho (Figura 6).

Esses resultados indicam que a abordagem proposta, guiada por heurísticas, apresenta boa capacidade de generalização para dados com estrutura caótica, como os encontrados no conjunto CA. No entanto, sua efetividade é limitada em cenários com alta aleatoriedade, como nos dados DW, onde a ausência de padrões consistentes dificulta o aprendizado das restrições. Isso sugere que a estratégia se adapta melhor a contextos em que há alguma regularidade topológica ou regras implícitas, mesmo que complexas, enquanto apresenta dificuldades para capturar distribuições mais ruidosas ou imprevisíveis.

Figura 20 – Comparação de mapas válidos entre CA e a abordagem proposta, onde CA alcança apenas 0,7% de mapas válidos, contra 10,1% de mapas válidos utilizando a mesma abordagem. Já o DW alcança 3,6% usando 4 agentes por 300 passos, contra 0,00008% de mapas válidos treinados com esses dados.

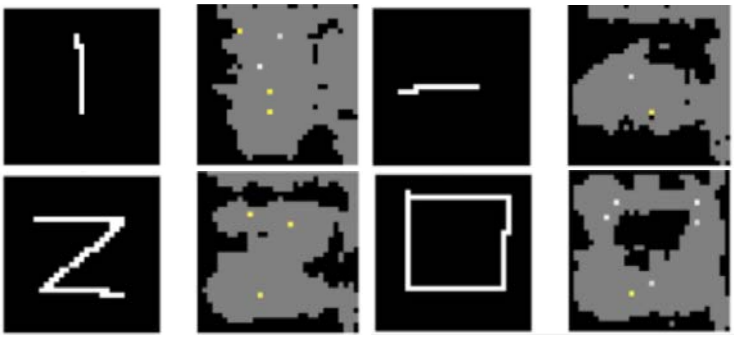


4.6 Entrada Condicional

A Figura 21 apresenta alguns resultados de amostras geradas a partir do modelo treinado. Observa-se que o uso desta entrada condicional consegue guiar o modelo para a construção de mapas, garantindo que os caminhos sigam a orientação definida pela entrada.

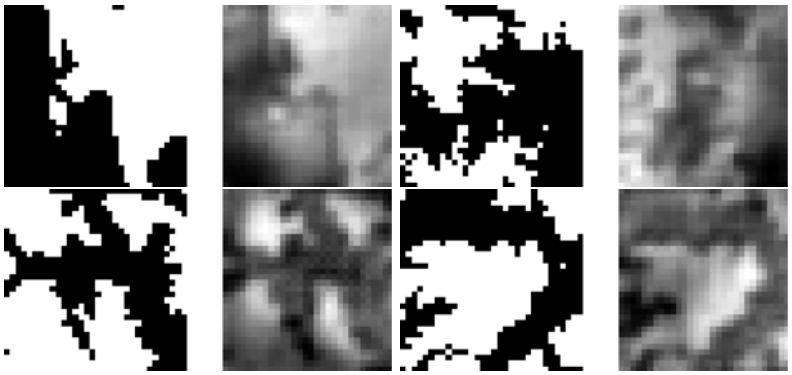
O modelo também foi treinado para gerar mapas com base em *heightmaps*, nos quais a saída representa um terreno semelhante aos dados apresentados durante o treinamento. Nesse cenário, somente a topografia é considerada, uma vez que os dados contêm exclusivamente informações sobre altura de relevo. Durante o treinamento da GAN, os *heightmaps* são utilizados como as amostras reais.

Figura 21 – Amostras geradas a partir do modelo treinado com os dados de Padilha (2022). *Sketch* usado como entrada condicional (esquerda); e amostra gerada (direita).



A Figura 22 ilustra alguns dos resultados obtidos após o treinamento. Observa-se que os mapas gerados apresentam variações atraentes de altura, porém perdem muitos detalhes finos. Quando comparados ambos os tipos de saída, observa-se que os formatos (*tiles* e *heightmap*) ajustam bem ao *sketch*. Mesmo com a baixa resolução, a rede consegue gerar amostras coerentes.

Figura 22 – Amostras geradas a partir do modelo treinado com os dados do Kaggle.*Sketch* usado como entrada condicional do modelo treinado (esquerda); amostra gerada (direita).



Para avaliar o desempenho do modelo em um cenário de jogo, as amostras geradas foram aplicadas em um ambiente virtual implementado da Unity. O modelo foi utilizado para criar níveis jogáveis, integrando os mapas gerados ao motor de jogo para simulação e interação em tempo real.

As Figuras 23 e 24 mostram o uso dos mapas gerados no ambiente virtual. Os resultados de *gameplay* demonstram que, apesar das diferenças nas representações (*tiles* e *heightmap*), os mapas gerados foram adequados para exploração no mundo virtual, mostrando a flexibilidade e a aplicabilidade do modelo em contextos de jogos.

Figura 23 – Terreno gerado proceduralmente na Unity a partir do modelo treinado com dados do Kaggle. *Sketch* fornecido pelo usuário e o respectivo resultado gerado pelo modelo (cima); simulação do terreno gerado (baixo).

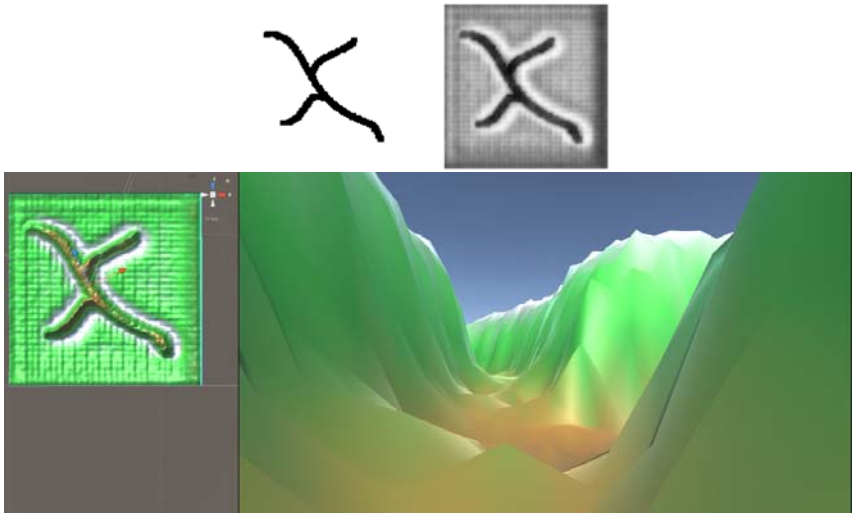
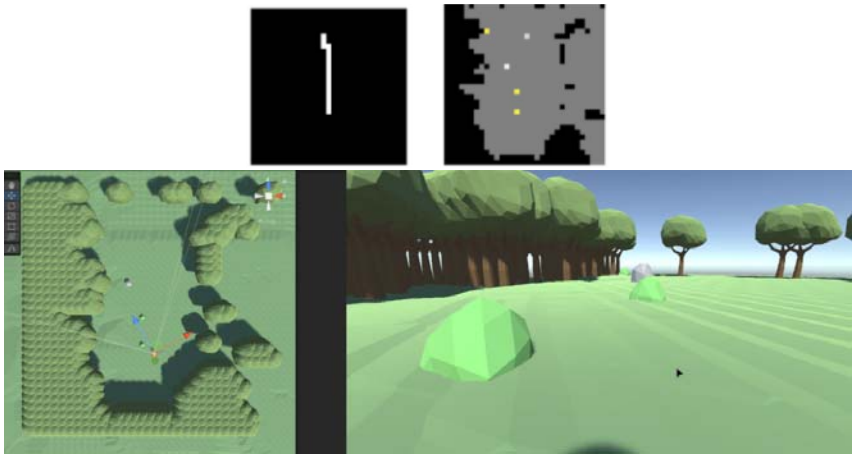


Figura 24 – Mapa gerado proceduralmente na Unity a partir do modelo treinado com dados de Padilha (2022). *Sketch* fornecido pelo usuário e o respectivo resultado gerado pelo modelo (cima); simulação do terreno gerado (baixo).



Ainda não foi possível avaliar a variabilidade nem a taxa de jogabilidade, portanto outras considerações que avaliam o potencial generativo do modelo não são aprofundadas.

5 CONCLUSÃO

Esta Tese apresentou diferentes abordagens baseadas em GANs para a geração de mapas de visão *top-down*, focando na superação de desafios clássicos associados a esse tipo de modelo, como a dificuldade de convergência, o *mode-collapse* e a instabilidade durante o treinamento. Foram realizadas experimentações com diferentes funções de perda, técnicas de regularização e ajustes nas arquiteturas, visando melhorar a qualidade e a variabilidade dos mapas gerados, especificamente em cenários com poucos dados. Embora as modificações na arquitetura tenham ajudado a minimizar alguns dos problemas identificados, como o *mode-collapse*, as abordagens ainda resultaram em uma taxa baixa relativamente de mapas válidos, evidenciando as limitações do modelo proposto.

O uso de *Spectral Normalization* se destacou como um fator positivo na estabilização do treinamento e na melhoria das métricas de jogabilidade, o que se mostrou vantajoso para a geração de mapas com maior qualidade. A flexibilidade proporcionada pelas redes convolucionais permitiu a criação de mapas em diferentes resoluções, demonstrando o potencial de generalização do modelo para diferentes tamanhos de mapas, como mapas 64×64, onde a geração de mapas válidos foi possível, embora em uma taxa ainda baixa.

A abordagem proposta conseguiu alcançar uma taxa mais alta de mapas válidos e variados para o conjunto de dados do algoritmo CA, evitando o problema de *mode-collapse* e apresentando bons resultados na aprendizagem da distribuição dos mapas. Modelos treinados com dados de CA conseguiram aprender essa distribuição, além de gerar mapas com maior taxa de validade e variabilidade. Da mesma forma, modelos treinados com DW também conseguiram reproduzir mapas dessa distribuição. Isso possibilitou a geração de mapas válidos a partir de dados com resoluções superiores àquelas utilizadas no treinamento inicial. Em ambos os casos, as arquiteturas convolucionais treinadas foram significativamente mais rápidas que os métodos CA ou DW.

Finalmente, ficou claro que um modelo generativo precisa considerar as características intrínsecas dos dados para garantir que a rede aprenda a gerar resultados

válidos e variados. A estratégia de alimentar amostras inválidas durante o treinamento, juntamente com a técnica de *bootstraping* e o uso de um termo de variabilidade, foi eficaz em melhorar a qualidade e a diversidade das gerações. Essa abordagem demonstrou ser uma alternativa interessante para a geração de mapas, tendo ganhos nos percentuais de validade, mas ainda sendo necessárias maiores experimentações para avaliar se a técnica de *bootstraping* não introduz vieses indesejados ao modelo.

5.1 Trabalhos Futuros

Como trabalhos futuros desta Tese, destaca-se o aprimoramento do modelo para considerar a experiência do jogador como um fator condicional na geração dos níveis, tornando o desenho mais dinâmico e adaptável. Esse aspecto integrado ainda é pouco explorado na literatura, onde a maioria dos trabalhos foca na estrutura e validade dos níveis, sem considerar a adaptação ao perfil e comportamento do jogador.

Diferentes perfis de jogadores podem exigir abordagens distintas na geração dos níveis (Nacke; Bateman; Mandryk, 2014). Por exemplo, jogadores colecionadores tendem a explorar os mapas em busca de itens escondidos, o que exigiria um desenho com caminhos alternativos, áreas ocultas e recompensas bem distribuídas. Já os jogadores mais voltados para velocidade e progressão preferem completar a história rapidamente, demandando níveis mais lineares, com poucos obstáculos e rotas diretas para o objetivo. Por outro lado, jogadores que valorizam combates podem se beneficiar de níveis estruturados para confrontos, com áreas de combate bem distribuídas e uma progressão que favoreça a progressão de desafios.

Além disso, outros ajustes nas arquiteturas de GANs ainda precisam ser testados para dados baseados em tile. Como observado, existem algumas dificuldades em relação ao tipo de função de custo empregada, sendo fundamental a exploração de novas estratégias que possam melhorar o percentual de amostras válidas geradas. A otimização do processo de treinamento pode ser alcançada por meio da adaptação da função de custo, incluindo penalidades que melhor refletem as características dos níveis, especialmente em termos de conectividade e o desenho da jogabilidade. Isso pode incluir, a distribuição de objetivos, a relação entre desafio e recompensa, e a organização estratégica de inimigos e itens coletáveis. Experimentações adicionais podem incluir a exploração de novas formas de *embeddings* que possam representar de maneira mais eficaz as características espaciais e estruturais dos tiles. Além disso, a utilização de *embeddings* hierárquicos ou baseados em grafos poderia capturar melhor as interdependências entre tiles e permitir a geração de níveis mais coesos.

Após o destaque das LLMs, outros modelos baseados em aprendizado profundo, como os *transformers* e modelos de difusão, vêm ganhando espaço em diversas aplicações, principalmente na geração de imagens. Os *transformers*, inicialmente desen-

volvidos para processamento de linguagem natural, vêm sendo utilizados em aplicação de visão computacional e geração de música, por exemplo. Já os modelos de difusão, onde imagens são gradualmente corrompidas por ruído e depois reconstruídas, têm sido muito utilizados em síntese de imagens. No entanto, essas técnicas geralmente exigem um volume de dados maior do que o disponível nas bases atuais de jogos. Uma estratégia promissora para viabilizar o uso desses modelos nesse contexto é adotar uma abordagem híbrida, na qual um modelo previamente treinado de GAN seja utilizado como contexto. Essa combinação pode reduzir a dependência de grandes conjuntos de dados, ao mesmo tempo, em que aprimora a qualidade das amostras geradas, aproveitando as capacidades dos modelos mais recentes sem comprometer a viabilidade do treinamento.

Outro ponto relevante é a avaliação da qualidade dos níveis gerados. Apesar dos avanços na geração de conteúdo por ML, ainda há poucos trabalhos na literatura que envolvem testes com usuários para analisar a qualidade e a jogabilidade dos níveis gerados. Algumas pesquisas utilizam abordagens automatizadas, como agentes inteligentes que simulam o comportamento de jogadores humanos, permitindo análises em larga escala. No entanto, essa abordagem, embora útil, não substitui a necessidade de avaliações subjetivas com jogadores reais. A validação experimental pode ser conduzida por meio de testes controlados com jogadores, coleta de dados quantitativos, como tempo de jogo e taxa de sucesso, além de aplicação de questionários e *feedback* qualitativo. Essas abordagens são essenciais para obter *insights* e aprimorar os critérios de geração.

REFERÊNCIAS

ABRAHAM, F.; STEPHENSON, M. Utilizing generative adversarial networks for stable structure generation in angry birds. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE AND INTERACTIVE DIGITAL ENTERTAINMENT, 2023. **Proceedings...** AAAI, 2023. v.19, p.2–12.

ARJOVSKY, M.; CHINTALA, S.; BOTTOU, L. Wasserstein GAN. **arXiv**, arXiv, Jan. 2017.

AWISZUS, M.; SCHUBERT, F.; ROSENHAHN, B. TOAD-GAN: Coherent Style Level Generation from a Single Example. In: SIXTEENTH AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE AND INTERACTIVE DIGITAL ENTERTAINMENT, 2020. **Proceedings...** AAAI Press, 2020. (AIIDE'20).

AWISZUS, M.; SCHUBERT, F.; ROSENHAHN, B. World-gan: a generative model for minecraft worlds. In: IEEE CONFERENCE ON GAMES (COG), 2021. **Proceedings...** IEEE, 2021. p.1–8.

AWISZUS, M.; SCHUBERT, F.; ROSENHAHN, B. Wor(l)d-GAN: Toward Natural-Language-Based PCG in Minecraft. **IEEE Transactions on Games**, IEEE, v.15, n.2, p.182–192, 2023.

BECKHAM, C.; PAL, C. **A step towards procedural terrain generation with GANs**. 2017. Disponível em: <https://arxiv.org/abs/1707.03383>.

BLATZ, M.; KORN, O. A Very Short History of Dynamic and Procedural Content Generation. In: KORN, O.; LEE, N. (Ed.). **Game Dynamics**: Best Practices in Procedural and Dynamic Game Content Generation. Springer: Springer International Publishing, 2017. p.1–13.

BONTRAGER, P.; TOGELIUS, J. Learning to Generate Levels From Nothing. In: IEEE CONFERENCE ON GAMES (COG), 2021. **Proceedings...** IEEE Press, 2021. p.1–8.

CAPPS, B.; SCHRUM, J. Using Multiple Generative Adversarial Networks to Build Better-Connected Levels for Mega Man. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE, 2021, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2021. p.66–74. (GECCO '21).

CHEN, Z.; LYU, D. Procedural generation of virtual pavilions via a deep convolutional generative adversarial network. **Computer Animation and Virtual Worlds**, Wiley Online Library, v.33, n.3-4, p.e2063, 2022.

GUZDIAL, M.; SNODGRASS, S.; SUMMERVILLE, A. J. **Constraint-Based PCGML Approaches**. Cham: Springer International Publishing, 2022. p.51–66.

CSIKSZENTMIHALYI, M.; CSIKZENTMIHALY, M. **Flow**: The psychology of optimal experience. New York: Harper & Row, 1990. v.1990.

DE KEGEL, B.; HAAHR, M. Procedural Puzzle Generation: A Survey. **IEEE Transactions on Games**, IEEE, v.12, n.1, p.21–40, 2020.

EARLE, S. et al. Illuminating Diverse Neural Cellular Automata for Level Generation. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE, 2022, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2022. p.68–76. (GECCO '22).

FERBER, A. et al. **GenCO**: Generating Diverse Designs with Combinatorial Constraints. 2024. Disponível em: <https://arxiv.org/abs/2310.02442>.

GIACOMELLO, E.; LANZI, P. L.; LOIACONO, D. Doom level generation using generative adversarial networks. In: IEEE GAMES, ENTERTAINMENT, MEDIA CONFERENCE (GEM), 2018. **Proceedings...** IEEE, 2018. p.316–323.

GIACOMELLO, E.; LANZI, P. L.; LOIACONO, D. Searching the latent space of a generative adversarial network to generate Doom levels. In: IEEE CONFERENCE ON GAMES (COG), 2019. **Proceedings...** IEEE, 2019. p.1–8.

GIACOMELLO, E.; LANZI, P. L.; LOIACONO, D. An analysis of DOOM level generation using Generative Adversarial Networks. **Entertainment Computing**, Elsevier, v.46, p.100549, 2023.

GISSLéN, L. et al. Adversarial Reinforcement Learning for Procedural Content Generation. In: IEEE CONFERENCE ON GAMES (COG), 2021. **Proceedings...** IEEE, 2021. p.1–8.

GOODFELLOW, I. J. et al. Generative Adversarial Nets. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS - VOLUME 2,

27., 2014, Cambridge, MA, USA. **Proceedings...** MIT Press, 2014. p.2672–2680. (NIPS'14).

GOODFELLOW, I. et al. Generative Adversarial Networks. **Commun. ACM**, New York, NY, USA, v.63, n.11, p.139–144, oct 2020.

GUÉRIN, É. et al. Interactive example-based terrain authoring with conditional generative adversarial networks. **ACM Trans. Graph.**, ACM, v.36, n.6, p.228–1, 2017.

GULRAJANI, I. et al. **Improved Training of Wasserstein GANs**. 2017. arXiv: arXiv, 2017. Disponível em: <https://arxiv.org/abs/1704.00028>.

GUTIERREZ, J.; SCHRUM, J. Generative adversarial network rooms in generative graph grammar dungeons for the legend of zelda. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION (CEC), 2020. **Proceedings...** IEEE, 2020. p.1–8.

HALD, A. et al. Procedural content generation of puzzle games using conditional generative adversarial networks. In: INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES, 15., 2020. **Proceedings...** ACM, 2020. p.1–9.

HAUSKNECHT, M. et al. Interactive fiction games: A colossal adventure. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2020. **Proceedings...** AAAI, 2020. v.34, p.7903–7910.

HENDRIKX, M. et al. Procedural content generation for games: A survey. **ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)**, [S.l.], v.9, n.1, p.1–22, 2013.

HUGHES, R. T.; ZHU, L.; BEDNARZ, T. Generative Adversarial Networks–Enabled Human–Artificial Intelligence Collaborative Applications for Creative and Design Industries: A Systematic Review of Current Approaches and Trends. **Frontiers in Artificial Intelligence**, Frontiers Media S.A, v.4, 2021.

IRFAN, A.; ZAFAR, A.; HASSAN, S. Evolving levels for general games using deep convolutional generative adversarial networks. In: COMPUTER SCIENCE AND ELECTRONIC ENGINEERING (CEEC), 11., 2019. **Proceedings...** IEEE, 2019. p.96–101.

KELVIN, L. Z.; ANAND, B. Procedural Generation of Roads with Conditional Generative Adversarial Networks. In: IEEE SIXTH INTERNATIONAL CONFERENCE ON MULTIMEDIA BIG DATA (BIGMM), 2020. **Proceedings...** IEEE, 2020. p.277–281.

KERSSEMAKERS, M. et al. A procedural procedural level generator generator. In: IEEE CONFERENCE ON COMPUTATIONAL INTELLIGENCE AND GAMES (CIG), 2012. **Proceedings...** IEEE, 2012. p.335–341.

KHALIFA, A. et al. PCGRL: Procedural Content Generation via Reinforcement Learning. **Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment**, AAAI, v.16, n.1, p.95–101, Oct. 2020.

KIM, S. W. et al. Learning to simulate dynamic environments with gamegan. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2020. **Proceedings...** IEEE/CVF, 2020. p.1231–1240.

KREITZER, M.; ASHLOCK, D.; PEREIRA, R. Automatic generation of diverse cavern maps with morphing cellular automata. In: IEEE CONFERENCE ON GAMES (COG), 2019. **Proceedings...** IEEE, 2019. p.1–8.

KUMARAN, V.; MOTT, B.; LESTER, J. Generating game levels for multiple distinct games with a common latent space. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE AND INTERACTIVE DIGITAL ENTERTAINMENT, 2019. **Proceedings...** AAAI, 2019. v.15, p.102–108.

LEHMAN, J.; STANLEY, K. O. Revising the evolutionary computation abstraction: minimal criteria novelty search. In: GENETIC AND EVOLUTIONARY COMPUTATION, 12., 2010. **Proceedings...** ACM, 2010. p.103–110.

LI, B. et al. Learning to Reconstruct Botanical Trees from Single Images. **ACM Trans. Graph.**, Association for Computing Machinery, v.40, n.6, 2021.

LIELLO, L. D. et al. Efficient generation of structured objects with constrained adversarial networks. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 34., 2020, Red Hook, NY, USA. **Proceedings...** Curran Associates Inc., 2020. (NIPS '20).

LINDEN, R.; LOPES, R.; BIDARRA, R. Designing Procedurally Generated Levels. **Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment**, AAAI Press, v.9, n.3, p.41–47, Jun. 2021.

LINDEN, R. van der; LOPES, R.; BIDARRA, R. Procedural Generation of Dungeons. **IEEE Transactions on Computational Intelligence and AI in Games**, IEEE, v.6, n.1, p.78–89, 2014.

LIU, J. et al. Deep learning for procedural content generation. **Neural Computing and Applications**, Springer, v.33, n.1, p.19–37, 2021.

MANDELBROT, B. **The Fractal Geometry of Nature**. New York: Henry Holt and Company, 1983. (Einaudi paperbacks).

MANIKTALA, M. et al. MINUET: Procedural Musical Accompaniment for Textual Narratives. In: INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES, 15., 2020, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2020. p.1–7.

MELHART, D. et al. Your gameplay says it all: Modelling motivation in tom clancy's the division. In: IEEE CONFERENCE ON GAMES (COG), 2019. **Proceedings...** IEEE, 2019. p.1–8.

MININI, P.; ASSUNCAO, J. Combining constructive procedural dungeon generation methods with wavefunctioncollapse in top-down 2D games. **Proceedings of SBGames**, SBC, 2020.

MIRGATI, N.; GUZDIAL, M. Joint Level Generation and Translation Using Gameplay Videos. In: IEEE CONFERENCE ON GAMES (COG), 2023. **Proceedings...** IEEE, 2023. p.1–10.

MIRZA, M.; OSINDERO, S. **Conditional Generative Adversarial Nets**. 2014. Disponível em: <https://arxiv.org/abs/1411.1784>.

MIYATO, T. et al. **Spectral Normalization for Generative Adversarial Networks**. 2018. Disponível em: <https://arxiv.org/abs/1802.05957>.

NACKE, L. E.; BATEMAN, C.; MANDRYK, R. L. BrainHex: A neurobiological gamer typology survey. **Entertainment Computing**, Elsevier, v.5, n.1, p.55–62, 2014.

NAM, S.; HSUEH, C.-H.; IKEDA, K. Procedural Content Generation of Super Mario Levels Considering Natural Connection. In: INTERNATIONAL JOINT CONFERENCE ON COMPUTER SCIENCE AND SOFTWARE ENGINEERING (JCSSE), 20., 2023. **Proceedings...** IEEE, 2023. p.291–296.

NEUMANN, J. V.; BURKS, A. W. **Theory of Self-Reproducing Automata**. USA: University of Illinois Press, 1966.

NUNES, V.; DIAS, J.; SANTOS, P. A. **GAN-Based Content Generation of Maps for Strategy Games**. 2023. Disponível em: <https://arxiv.org/abs/2301.02874>.

OLIVEIRA, W. et al. **Does gamification affect flow experience? A systematic literature review**. 2021. Disponível em: <https://arxiv.org/abs/2106.09942>.

PADILHA, R. F. **Analisando o engajamento de jogadores através de mapas procedurais**. 2022. 41p. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

PARK, K. et al. Generating educational game levels with multistep deep convolutional generative adversarial networks. In: IEEE CONFERENCE ON GAMES (COG), 2019. **Proceedings...** IEEE, 2019. p.1–8.

PERLIN, K. An image synthesizer. **ACM Siggraph Computer Graphics**, ACM, v.19, n.3, p.287–296, 1985.

PING, K.; DINGLI, L. Conditional convolutional generative adversarial networks based interactive procedural game map generation. In: FUTURE OF INFORMATION AND COMMUNICATION CONFERENCE (FICC), VOLUME 1, 2020. **Proceedings...** Springer, 2020. p.400–419.

PRUSINKIEWICZ, P.; LINDENMAYER, A. **The algorithmic beauty of plants**. Springer: Springer Science & Business Media, 2012.

RADFORD, A.; METZ, L.; CHINTALA, S. **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks**. 2016. Disponível em: <https://arxiv.org/abs/1511.06434>.

RAJABI, M. et al. A dynamic balanced level generator for video games based on deep convolutional generative adversarial networks. **Scientia Iranica**, Sharif University of Technology, v.28, n.Special issue on collective behavior of nonlinear dynamical networks, p.1497–1514, 2021.

RAMOS, N.; SANTOS, P.; DIAS, J. Dual Critic Conditional Wasserstein GAN for Height-Map Generation. In: INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES, 18., 2023. **Proceedings...** ACM, 2023. p.1–4.

RATLIFF, L. J.; BURDEN, S. A.; SASTRY, S. S. Characterization and computation of local Nash equilibria in continuous games. In: ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING (ALLERTON), 51., 2013. **Proceedings...** IEEE, 2013. p.917–924.

RISI, S.; TOGELIUS, J. Increasing generality in machine learning through procedural content generation. **Nature Machine Intelligence**, Nature Publishing Group UK London, v.2, n.8, p.428–436, 2020.

ROBERTS, D. A.; YAIDA, S.; HANIN, B. **The Principles of Deep Learning Theory**: An Effective Theory Approach to Understanding Neural Networks. England: Cambridge University Press, 2022.

RODRIGUEZ TORRADO, R. et al. Bootstrapping Conditional GANs for Video Game Level Generation. In: IEEE CONFERENCE ON GAMES (COG), 2020. **Proceedings...** IEEE, 2020. p.41–48.

SARKAR, A.; COOPER, S. Sequential Segment-Based Level Generation and Blending Using Variational Autoencoders. In: INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES, 15., 2020, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2020. (FDG '20).

SARKAR, A.; COOPER, S. Generating and Blending Game Levels via Quality-Diversity in the Latent Space of a Variational Autoencoder. In: INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES, 16., 2021, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2021. (FDG '21).

SARKAR, A.; COOPER, S. Dungeon and Platformer Level Blending and Generation using Conditional VAEs. In: IEEE CONFERENCE ON GAMES (COG), 2021. **Proceedings...** IEEE, 2021.

SARKAR, A.; COOPER, S. tile2tile: learning game filters for platformer style transfer. In: EIGHTEENTH AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE AND INTERACTIVE DIGITAL ENTERTAINMENT, 2022. **Proceedings...** AAAI Press, 2022. (AIIDE'22).

SARKAR, A.; COOPER, S. **Latent Combinational Game Design**. 2023. Disponível em: <https://arxiv.org/abs/2206.14203>.

SARKAR, A. et al. Procedural content generation via knowledge transformation (PCG-KT). **IEEE Transactions on Games**, IEEE, 2023.

SCHRUM, J. et al. Interactive evolution and exploration within latent level-design space of generative adversarial networks. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE, 2020, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2020. p.148–156.

SCHUBERT, F.; AWISZUS, M.; ROSENHAHN, B. Toad-gan: a flexible framework for few-shot level generation in token-based games. **IEEE Transactions on Games**, IEEE, v.14, n.2, p.284–293, 2021.

SESTINI, A.; KUHNLE, A.; BAGDANOV, A. D. Demonstration-Efficient Inverse Reinforcement Learning in Procedurally Generated Environments. In: IEEE CONFERENCE ON GAMES (COG), 2021. **Proceedings...** IEEE, 2021. p.1–8.

SHAKER, N. et al. The 2010 Mario AI championship: Level generation track. **IEEE Transactions on Computational Intelligence and AI in Games**, IEEE, v.3, n.4, p.332–347, 2011.

SHAKER, N.; TOGELIUS, J.; NELSON, M. J. **Procedural Content Generation in Games**: A Textbook and an Overview of Current Research. Switzerland: Springer, 2016.

SILVA, D. F. e et al. Dungeon level generation using generative adversarial network: an experimental study for top-down view games. In: L SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 2023. **Anais...** SBC, 2023. p.95–106.

SILVA, D. F. e.; TORCHELSEN, R.; AGUIAR, M. Procedural game level generation with GANs: potential, weaknesses, and unresolved challenges in the literature. **Multimedia Tools and Applications**, Springer, p.1–27, 01 2025.

SILVA, D. F. E.; TORCHELSEN, R. P.; DE AGUIAR, M. S. How to improve the quality of GAN-based map generators. In: BRAZILIAN SYMPOSIUM ON GAMES AND DIGITAL ENTERTAINMENT, 22., 2024, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2024. p.106–113. (SBGames '23).

SMELIK, R. M. et al. A Survey on Procedural Modelling for Virtual Worlds. **Computer Graphics Forum**, Wiley, v.33, n.6, p.31–50, 2014.

STECKEL, K.; SCHRUM, J. Illuminating the space of beatable lode runner levels produced by various generative adversarial networks. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE COMPANION, 2021, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2021. p.111–112. (GECCO '21).

SUMMERVILLE, A. et al. Procedural content generation via machine learning (PCGML). **IEEE Transactions on Games**, IEEE, v.10, n.3, p.257–270, 2018.

TOGELIUS, J. et al. Search-based procedural content generation: A taxonomy and survey. **IEEE Transactions on Computational Intelligence and AI in Games**, IEEE, v.3, n.3, p.172–186, 2011.

USMAN, F.; ANWAR, S. M.; RAUF, U. Visual Recognition for ZELDA Content Generation via Generative Adversarial Network. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (ICAI), 3., 2023. **Proceedings...** IEEE, 2023. p.76–81.

VIANA, B. M. F.; SANTOS, S. R. dos. Procedural Dungeon Generation: A Survey. **Journal on Interactive Systems**, SBC, v.12, n.1, p.83–101, Aug. 2021.

VOLZ, V. et al. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE, 2018, New York, NY, USA. **Proceedings...** Association for Computing Machinery, 2018. p.221–228. (GECCO '18).

VOLZ, V. et al. Capturing Local and Global Patterns in Procedural Content Generation via Machine Learning. In: IEEE CONFERENCE ON GAMES (COG), 2020. **Proceedings...** IEE, 2020. p.399–406.

VOULGARIS, G.; MADEMLIS, I.; PITAS, I. Procedural terrain generation using generative adversarial networks. In: EUROPEAN SIGNAL PROCESSING CONFERENCE (EUSIPCO), 29., 2021. **Proceedings...** IEEE, 2021. p.686–690.

WANG, Z.; LIU, J. Online Game Level Generation from Music. In: IEEE CONFERENCE ON GAMES (COG), 2022. **Proceedings...** IEEE, 2022. p.119–126.

WANG, Z.; LIU, J.; YANNAKAKIS, G. N. Keiki: Towards Realistic Danmaku Generation via Sequential GANs. In: IEEE CONFERENCE ON GAMES (COG), 2021. **Proceedings...** IEEE, 2021. p.01–04.

YANNAKAKIS, G. N.; TOGELIUS, J. Experience-Driven Procedural Content Generation. **IEEE Transactions on Affective Computing**, IEEE Computer Society, v.2, n.3, p.147–161, 2011.

ZHANG, H. et al. Video Game Level Repair via Mixed Integer Linear Programming. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE AND INTERACTIVE DIGITAL ENTERTAINMENT, 2020. **Proceedings...** AAAI Press, 2020. (AIIDE'20).

Apêndices

APÊNDICE A – Demonstração dos Experimentos

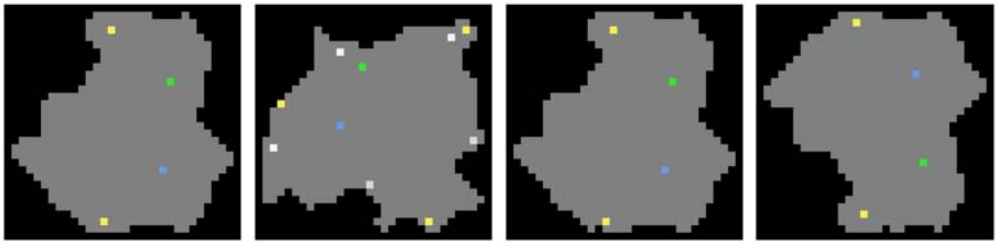
A.1 Comparação dos Otimizadores

Os experimentos apresentados foram baseados em trabalhos do estado da arte. Das arquiteturas apresentadas, foi realizado um estudo com diferentes otimizadores e modificações de hiperparâmetros para o ajuste adequado do modelo.

Com a utilização da arquitetura proposta por Goodfellow et al. (2014) foi possível observar os problemas clássicos de algoritmos de redes adversárias generativas. E com a modificação dos otimizadores foi possível obter melhores resultados da métrica de qualidade proposta. No entanto, somente com a mudança da arquitetura foi possível minimizar alguns dos problemas clássicos das GANs tais como: *mode collapse*, dificuldade de convergência e instabilidade.

O problema de *mode collapse* ocorre quando a rede aprende uma distribuição na qual a geração resulta em dados muito semelhantes. A rede pode convergir para a geração de amostras com um comportamento distante da distribuição do conjunto de dados apresentados durante o treinamento, como demonstrado anteriormente na Figura 28. O mesmo pode ocorrer para a convergência de um comportamento muito próximo ao conjunto de dados do treinamento, como demonstrado na Figura 25, referente aos resultados obtidos após treinamento da rede. Esta última foi treinada utilizando o otimizador RMSprop.

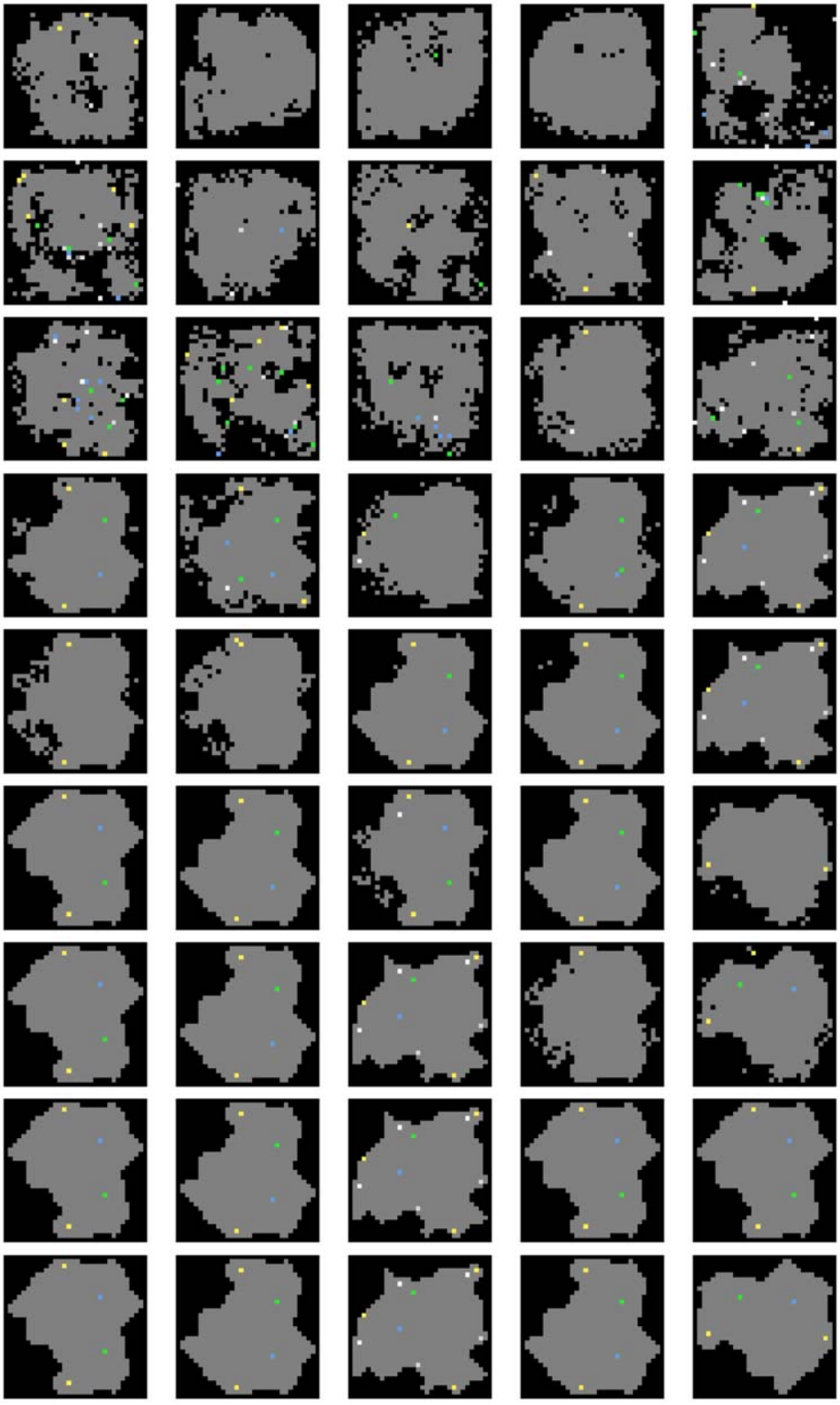
Figura 25 – Visualização dos resultados válidos após treinamento por 10.000 épocas utilizando a configuração descrita na Figura 26.



O comportamento da rede durante o treinamento com o otimizador RMSprop pode ser observado na Figura 26.

Apenas a métrica de validação de mapa jogável não é possível analisar a ocorrência de *mode collapse*. A utilização de uma métrica de diversidade pode auxiliar essa análise e servir como apoio para o ajuste da rede.

Figura 26 – Visualização dos resultados por época de treinamento. A rede foi treinada por 10.000 épocas. Configurações do experimento: arquitetura: GAN; otimizador: RMSProp; *learning rate* 0,0001.



Nas experimentações realizadas com o otimizador SGD houve uma maior dificuldade para a convergência de resultados que atendessem à métrica de qualidade pro-

posta, apresentando pequenos picos, como demonstrado na Figura 27 e na Tabela 13.

Tabela 13 – Comparação dos treinamentos utilizando a arquitetura GAN com diferentes configurações de *learning rate* e otimizador SGD. A medição do mapa é definida como jogável quando a configuração do terreno e disposição dos elementos de jogo atendem as regras do jogo.

		SGD		
Learning Rate	**Época ($*10^3$)**	**Terreno válido (%)**	**Elementos válidos (%)**	**Mapa jogável (%)**
0.005	1	39.6	0.0	0.0
	2	0.0	0.0	0.0
	3	1.7	9.8	0.0
	4	**5.4**	**10.4**	**1.3**
	5	0.0	0.0	0.0
	6	0.0	1.7	0.0
	7	0.2	0.5	0.0
	8	0.1	0.0	0.0
	9	0.0	0.0	0.0
	10	0.0	0.0	0.0
0.015	1	46.0	0.0	0.0
	2	0.0	0.0	0.0
	3	0.7	3.3	0.0
	4	0.0	2.4	0.0
	5	0.0	0.3	0.0
	6	0.0	0.0	0.0
	7	0.6	2.1	0.0
	8	0.0	5.0	0.0
	9	0.0	0.0	0.0
	10	0.0	0.0	0.0
0.025	1	25.4	0.0	0.0
	2	26.7	0.1	0.0
	3	0.1	5.1	0.0
	4	0.0	0.2	0.0
	5	0.0	0.0	0.0
	6	0.0	0.0	0.0
	7	0.9	12.0	0.9
	8	0.0	0.2	0.0
	9	0.0	30.4	0.0
	10	0.0	0.8	0.0
0.035	1	53.3	0.0	0.0
	2	0.0	0.0	0.0
	3	0.0	0.0	0.0
	4	0.0	0.0	0.0
	5	0.0	0.0	0.0
	6	0.0	0.0	0.0
	7	0.0	0.0	0.0
	8	0.0	0.0	0.0
	9	0.0	0.0	0.0
	10	0.0	0.0	0.0

Com a mudança para os otimizadores Adam e RMSprop, foi possível obter mais resultados válidos, em especial para a quantidade de terrenos válidos (Tabela 14). No entanto, para a configuração desta arquitetura, ainda estava presente o problema de *mode collapse* (Figura 26). Alterando a arquitetura para a DCGAN a porcentagem de mapas válidos diminuiu para menos de 5%, mas os resultados gerados válidos

diferiam dos dados do conjunto de treinamento. Na Tabela 15 é possível observar esse comportamento.

Tabela 14 – Comparação dos otimizadores, na arquitetura GAN, com base na validação dos mapas. O mapa é definido como jogável quando a configuração do terreno e disposição dos elementos de jogo são válidos.

		Adam			**RMSprop**		
Learning Rate	**Época**	**Terreno válido (%)**	**Elementos válidos (%)**	**Mapa jogável (%)**	**Terreno válido (%)**	**Elementos válidos (%)**	**Mapa jogável (%)**
0.01	1000	3.0	0.0	0.0	0.8	0.0	0.0
	2000	0.5	0.0	0.0	0.6	0.0	0.0
	3000	0.5	0.0	0.0	1.5	0.0	0.0
	4000	0.4	0.0	0.0	1.2	0.0	0.0
	5000	0.5	0.0	0.0	0.8	0.0	0.0
	6000	0.4	0.0	0.0	0.6	0.0	0.0
	7000	0.8	0.0	0.0	0.4	0.0	0.0
	8000	0.5	0.0	0.0	0.3	0.0	0.0
	9000	0.2	0.0	0.0	0.9	0.0	0.0
	10000	0.8	0.0	0.0	1.5	0.0	0.0
0.001	1000	0.0	0.0	0.0			
	2000	44.1	0.0	0.0	13.0	0.0	0.0
	3000	33.3	3.4	3.4	4.0	0.0	0.0
	4000	33.4	3.7	3.7	3.0	0.0	0.0
	5000	33.8	3.2	3.2	2.7	0.0	0.0
	6000	33.2	2.8	2.8	3.2	0.0	0.0
	7000	**34.6**	**4.8**	**4.8**	8.7	0.0	0.0
	8000	31.5	3.3	3.3	5.8	0.0	0.0
	9000	34.9	3.8	3.8	5.3	0.0	0.0
	10000	33.1	3.9	3.9	5.9	0.0	0.0
0.0001	1000	47.5	0.0	0.0	0.0	0.0	0.0
	2000	0.0	0.0	0.0	0.3	5.4	0.0
	3000	2.7	0.0	0.0	0.0	10.0	0.0
	4000	61.2	0.0	0.0	0.4	4.8	0.1
	5000	22.6	0.1	0.1	42.7	74.3	41.9
	6000	29.8	0.0	0.0	80.6	88.9	79.4
	7000	14.4	0.2	0.2	87.7	93.9	86.6
	8000	7.4	0.0	0.0	92.1	94.8	91.5
	9000	5.7	0.1	0.1	83.7	90.1	83.4
	10000	5.1	0.5	0.5	**94.0**	**95.9**	**93.1**

A.2 Comparação do *Learning Rate*

Também foi realizado um estudo apenas variando o *learning rate* para o treinamento das redes. Como é possível observar na Figura 27, as gerações com a configuração padrão apresentam poucos resultados válidos durante o processo de treinamento. Ajustando o valor de *learning rate* para 0,005 a rede conseguiu atingir 8,4% de mapas válidos em 3.118 épocas de treinamento.

Para a visualização do comportamento da geração da rede durante o treinamento, são demonstrados na Figura 28 os resultados a cada 1.000 épocas de treinamento, sendo cada coluna representando um valor distinto do vetor de ruído z e nas linhas o resultado variando as épocas. Observando a variação da topologia do terreno e

Tabela 15 – Comparação dos treinamentos utilizando a arquitetura DCGAN em diferentes configurações de *learning rate* com os otimizadores Adam e RMSprop. Para o otimizador Adam o coeficiente β_1 foi definido com valor igual a 0,5; conforme definido por Radford; Metz; Chintala (2016).

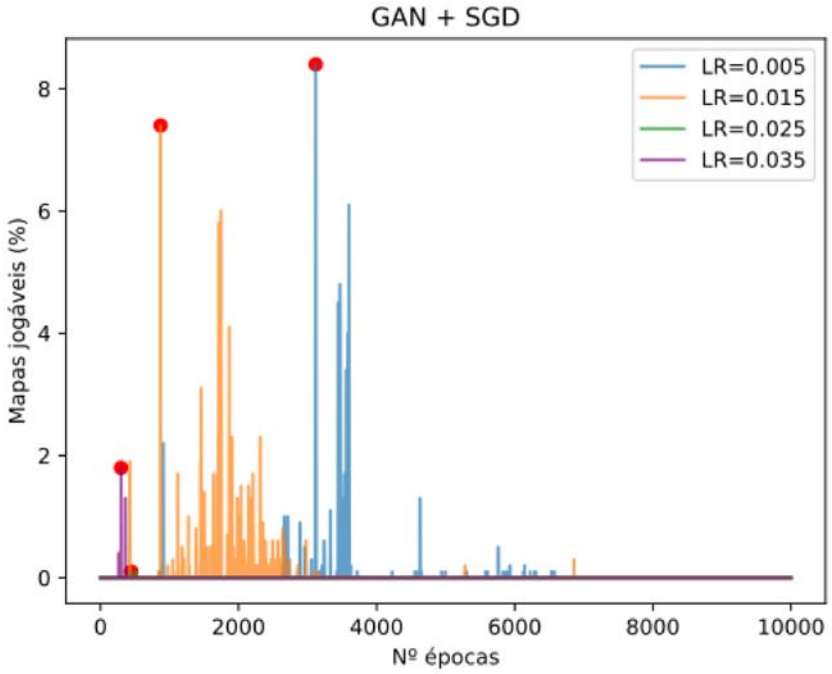
		Adam			**RMSprop**		
Learning Rate	**Época ($*10^3$)**	**Terreno válido (%)**	**Elementos válidos (%)**	**Mapa jogável (%)**	**Terreno válido (%)**	**Elementos válidos (%)**	**Mapa jogável (%)**
0.0002	1	0.2	0.0	0.0	34.2	0.0	0.0
	2	34.1	5.7	1.3	37.9	2.2	0.9
	3	20.6	7.1	1.2	19.6	6.4	1.0
	4	8.8	0.5	0.2	15.5	6.9	1.2
	5	4.4	0.1	0.0	1.6	2.0	0.0
	6	6.8	1.7	0.2	2.8	2.6	0.1
	7	19.9	2.0	0.3	0.3	7.4	0.0
	8	10.2	7.6	0.8	0.0	1.2	0.0
	9	9.7	5.9	0.6	0.2	6.6	0.0
	10	4.2	6.9	0.1	0.0	8.9	0.0
0.0001	1	0.0	0.0	0.0	58.0	0.0	0.0
	2	**49.8**	**4.7**	**2.4**	47.3	5.3	2.4
	3	47.3	3.0	1.9	32.1	2.3	1.3
	4	34.9	2.1	0.8	50.0	3.9	1.5
	5	26.6	8.0	2.3	36.6	5.5	2.3
	6	16.0	6.7	1.7	13.0	0.1	0.0
	7	25.0	7.2	2.3	1.8	7.0	0.3
	8	15.6	6.2	1.1	14.2	7.0	1.1
	9	2.8	8.1	0.2	0.1	10.3	0.0
	10	10.3	4.3	0.1	0.1	0.2	0.0
0.00009	1	15.1	0.0	0.0	21.0	0.0	0.0
	2	44.6	1.8	0.8	55.1	6.0	2.7
	3	42.0	2.7	1.3	49.2	0.5	0.4
	4	43.5	2.2	1.2	50.6	3.8	2.1
	5	34.3	5.2	1.5	**41.6**	**7.4**	**3.0**
	6	21.8	3.2	0.7	35.8	5.3	1.6
	7	17.3	0.1	0.0	14.3	7.4	1.3
	8	7.4	2.5	0.2	23.9	2.9	0.8
	9	2.6	0.1	0.0	7.4	0.6	0.0
	10	2.5	2.0	0.0	7.5	0.0	0.0

disposição dos elementos de jogos em 3.000 épocas – próximo ao pico de melhor modelo treinado para mapas válidos – a rede demonstra alguns mapas com uma área convexa do terreno e com a disposição de poucos elementos.

Esse comportamento se repete para muitos resultados válidos, destacados em azul na Figura 29, com pouca ou nenhuma variação na disposição dos elementos portal e posição do jogador. A partir de 5.000 épocas, a topologia do terreno muda um pouco, com algumas concavidades e maior quantidade de elementos dispostos no mapa, resultando em menos chances de amostras válidas nas gerações. Com base na análise dos mapas gerados durante os experimentos, foi possível verificar dois problemas descritos na literatura associados ao uso de GANs, sendo: a dificuldade de convergência e o *mode collapse*.

Também foi possível observar o desbalanceamento do equilíbrio de Nash, demonstrado na Figura 30 através do alto valor para a função de custo do gerador e do baixo valor para a função de custo do discriminador.

Figura 27 – Comparação de diferentes configurações de *learning rate* sob o otimizador SGD no treinamento da arquitetura GAN. O mapa é considerado válido quando atender aos critérios para jogabilidade e disposição dos elementos de jogo.



Um exemplo de *mode collapse* presente nos resultados pode ser observado na Figura 29. Cabe ressaltar que este problema não está associado somente aos resultados válidos *a priori*, uma vez que a rede também pode aprender a gerar um conjunto de dados inválidos muito semelhantes entre si, ou seja, há pouca diversidade nas amostras geradas pela rede.

Conforme a métrica de jogabilidade, o modelo gerou somente 4,8% de níveis válidos ao treinar a rede por 360 épocas. No entanto, a qualidade dos níveis não apresentou variabilidade, pois as amostras geradas consistiam em um grande bloco contínuo de pisos (sem paredes internas), além da disposição dos demais elementos estar predominantemente próxima às paredes, conforme ilustrado na Figura 10.

Após 3.000 épocas de treinamento, o modelo parou de aprender, estagnando na mesma geração (Figura 9). A configuração tradicional da VanillaGAN não conseguiu aprender o comportamento dos dados e ficou altamente suscetível ao problema de *mode-collapse*.

Assim, as gerações apresentadas não são satisfatórias. Esse resultado pode ser atribuído à baixa quantidade de amostras utilizadas para o treinamento, potencializando o *mode-collapse* e a baixa variabilidade. Pode-se afirmar que a VanillaGAN não se adapta satisfatoriamente ao conjunto de dados de níveis baseados em *tile* quando treinada com um volume muito pequeno de dados.

Na configuração da arquitetura convolucional, o modelo treinado conseguiu produzir terrenos com maior diversidade, como demonstrado na Figura 11.

Figura 28 – Evolução das gerações a cada época de treinamento (linhas), para 5 valores diferentes do espaço latente (colunas). Configurações da GAN: otimizador SGD, *learning rate* = 0,005; *momentum* = 0,5.

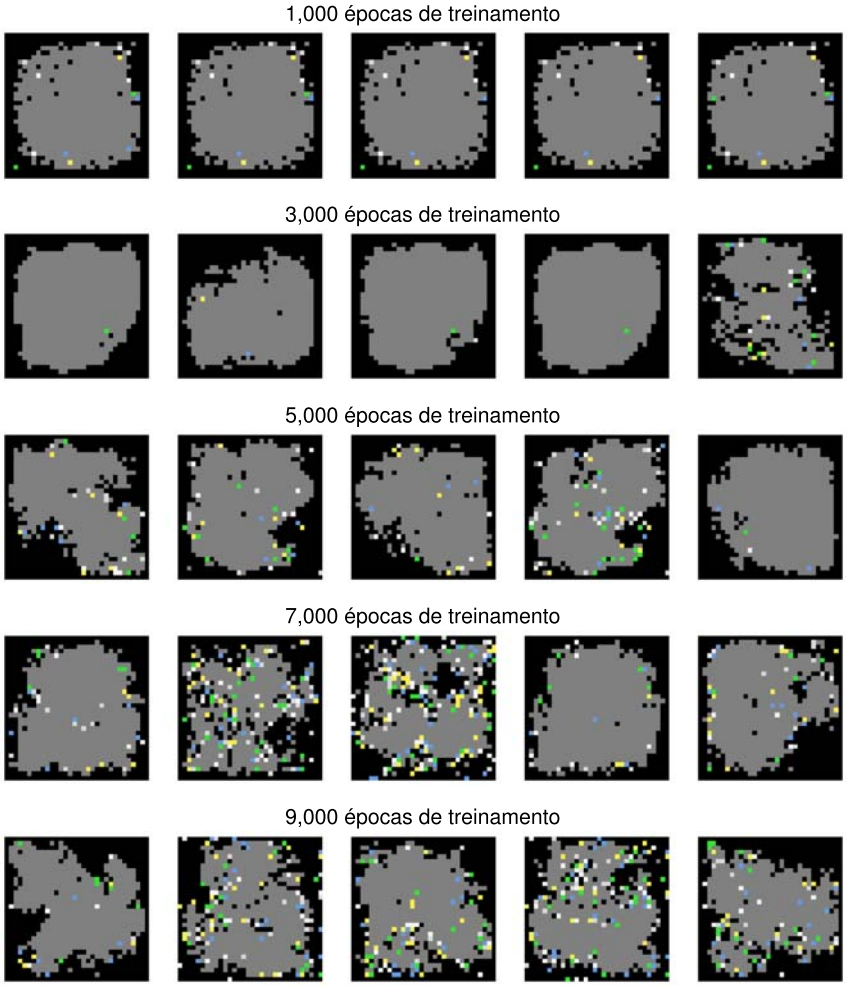


Figura 29 – Resultados do modelo treinado por 3118 épocas, com *learning rate* = 0,005 na arquitetura GAN. Contornados em azul estão os resultados ditos como válidos a partir da métrica de qualidade proposta.

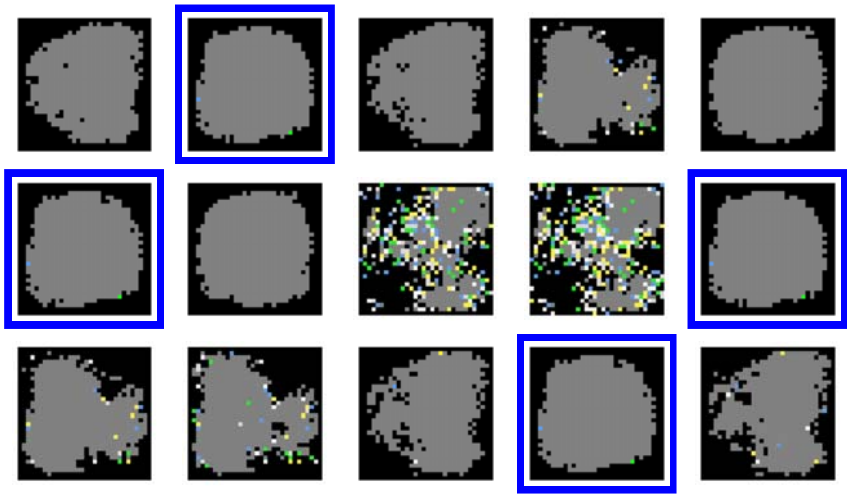
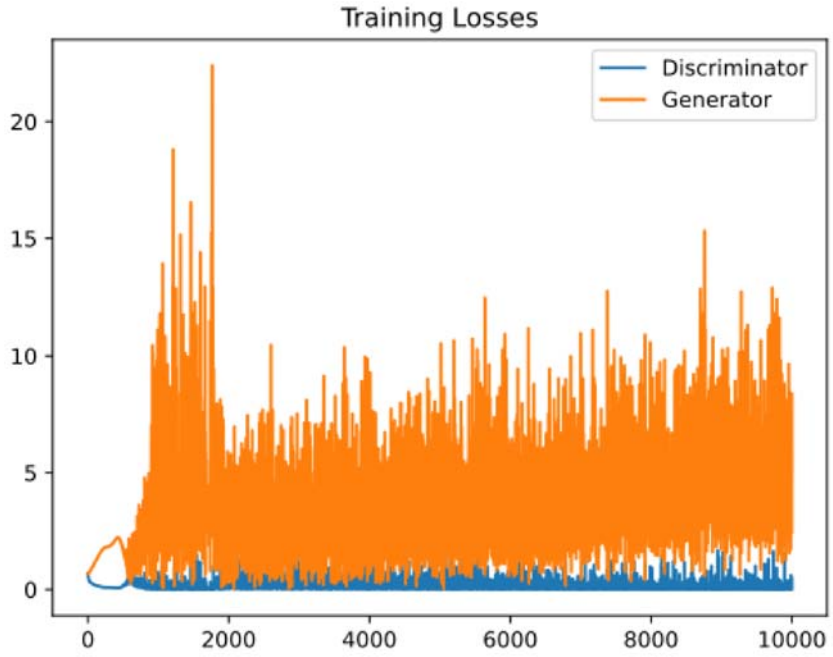


Figura 30 – Demonstração da função de custo durante o treinamento da GAN utilizando o otimizador SGD com *learning rate*=0,005.

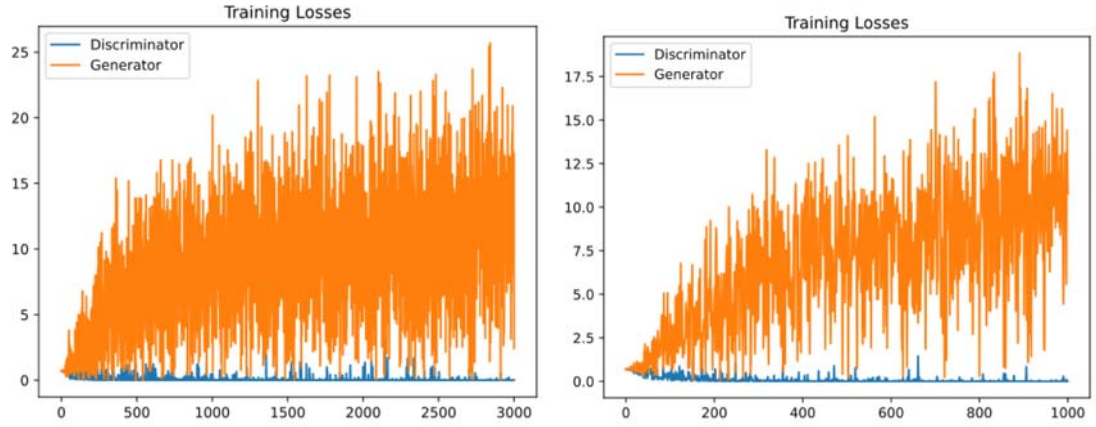


Comparado aos resultados obtidos com a VanillaGAN, os mapas gerados pela DCGAN apresentam corredores e paredes internas sem bloquear o caminho do jogador. Para otimizar o ajuste da rede, foram realizados experimentos variando o *learning rate* dos otimizadores Adam (Radford; Metz; Chintala, 2016) e RMSprop (Ping; Dingli, 2020). O objetivo desses experimentos foi comparar o desempenho da arquitetura e sua capacidade de gerar mapas válidos. Como mostrado na Figura 12, há pouca diferença no desempenho da rede na geração de mapas válidos: com o otimizador RMSprop, a rede alcançou 4,7% de mapas válidos, enquanto com o otimizador Adam, o percentual foi de 4,1%.

O comportamento de instabilidade da GAN está presente nos experimentos e pode ser observado de três formas: (1) com a variação dos valores percentuais de mapas válidos a cada época de treinamento, conforme apresentado na Figura 12; (2) com as modificações dos mapas (visualmente) a cada época, sofrendo alterações em sua topologia de terreno e disposição dos elementos de jogos, invalidando e validando o mapa, conforme demonstrado nas colunas 3 e 4 da Figura 14; e, (3) com a oscilação da função de custo do gerador, demonstrado na Figura 31.

Os resultados obtidos indicam que as GANs apresentam potencial para gerar mapas de jogos digitais mesmo com um conjunto de dados relativamente pequeno – neste caso, apenas 12 amostras –, embora com limitações observadas em relação à variabilidade e validade das saídas. Essa afirmação se sustenta pelo fato de que as arquiteturas convolucionais conseguem aprender padrões locais mesmo a partir do treinamento em um conjunto de dados reduzido. No entanto, em domínios caóticos,

Figura 31 – Demonstração das funções de custo das redes gerador e discriminador na arquitetura DCGAN. O eixo x mostra a quantidade de épocas treinadas, enquanto o eixo y mostra o valor da função de custo para cada rede. O gerador (em laranja) apresenta maior oscilação durante treinamento da rede do que o discriminador (azul).



como os gerados a partir do algoritmo CA, a representação desses padrões torna-se uma tarefa difícil. Isso se reflete nos resultados obtidos com o modelo treinado, que apresentou uma taxa de geração de amostras válidas próxima a 5%.

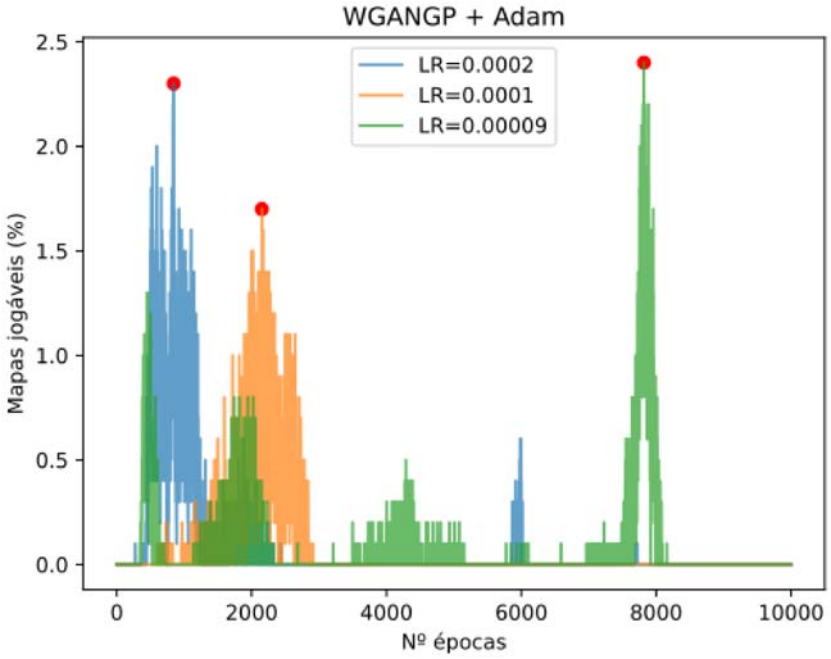
A terceira arquitetura testada foi a WGANGP, usando como base os hiperparâmetros definidos por Gulrajani et al. (2017), sendo: otimizador Adam com *learning rate* = 0,0001; β_1 = 0; β_2 = 0,9; o fator de penalidade λ = 10 e quantidade de iterações do crítico n_{critic} = 5. Para melhor ajuste de rede, esta foi treinada com variações do valor de *learning rate*.

A Figura 32 mostra menores oscilações na geração de mapas válidos quando comparado aos treinamentos realizados com a arquitetura DCGAN proposta por Gulrajani et al. (2017). No entanto, esperava-se que a quantidade de mapas válidos gerados pela arquitetura WGANGP fosse superior à da arquitetura DCGAN. O maior percentual de mapas válidos obtido durante o treinamento da WGANGP foi de 2,4% com o otimizador Adam e *learning rate* = 0.00009, enquanto a arquitetura DCGAN alcançou 4,7% de mapas válidos com o otimizador RMSprop.

Na Figura 33, estão os resultados do modelo treinado por 7817 épocas, com *learning rate* = 0,00009 na arquitetura WGANGP. Observa-se que a rede consegue gerar um número considerável de mapas que atendem à condição de chão totalmente conectado no terreno. Os mapas contornados em azul são considerados válidos, enquanto os contornados em amarelo atendem à topologia do terreno, mas são inválidos em relação à disposição dos elementos, conforme a métrica de qualidade proposta (Seção 3.2). Na Figura 34 é possível ratificar a afirmação sobre o número de mapas gerados atendendo ao critério do terreno, sendo obtido maior percentual na arquitetura DCGAN com 91.8% contra 88.4% da WGANGP.

A diferença entre as arquiteturas DCGAN e WGAN/WGANGP está no tipo de função de custo, a primeira baseada na divergência de Jensen–Shannon e a última ba-

Figura 32 – Comparação de diferentes configurações de *learning rate* sob o otimizador Adam.



seada na distância de Wasserstein. Além disso, a WGANGP utiliza um mecanismo de restrição 1-Lipschitz para estabilizar o treinamento, mais detalhes na Seção 2.2.4. Mesmo com a base teórica para estabilidade de treinamento, a WGANGP não teve sucesso para o problema apresentado. Após investigação, observou-se que este termo de penalidade não se adapta bem a dados discretos ou categóricos. Devido às mudanças abruptas de valores em cada *tile* do nível (domínio de dados), a interpolação das entradas do discriminador resulta em calores não categóricos, fazendo com que a aproximação dos gradientes seja muito próxima de 0, o que dificulta a aplicação adequada da penalização.

A.3 Comparação com a técnica de regularização

O objetivo desse experimento foi avaliar o impacto de diferentes arquiteturas e técnicas de regularização na qualidade das amostras geradas. Para isso, foram testadas as arquiteturas DCGAN, WGAN e WGANGP, todas utilizando a mesma arquitetura convolucional. A iteração do crítico (WGAN) e o parâmetro lambda (WGANGP) foram ajustados para melhorar os resultados de saída. Nos experimentos, não houve diferença significativa entre os otimizadores Adam e RMSprop. Portanto, optou-se pelo segundo devido à redução da quantidade de hiperparâmetros. Os primeiros resultados desse experimento foram obtidos utilizando camadas de *Batch Normalization* na rede Discriminador/Crítico.

Figura 33 – Resultados do modelo treinado por 7817 épocas, com *learning rate* 0.00009 na arquitetura WGANGP. Os mapas contornados em azul são resultados ditos como válidos e em amarelo os mapas válidos para a topologia do terreno e inválido para a disposição dos elementos, a partir da métrica de qualidade proposta.

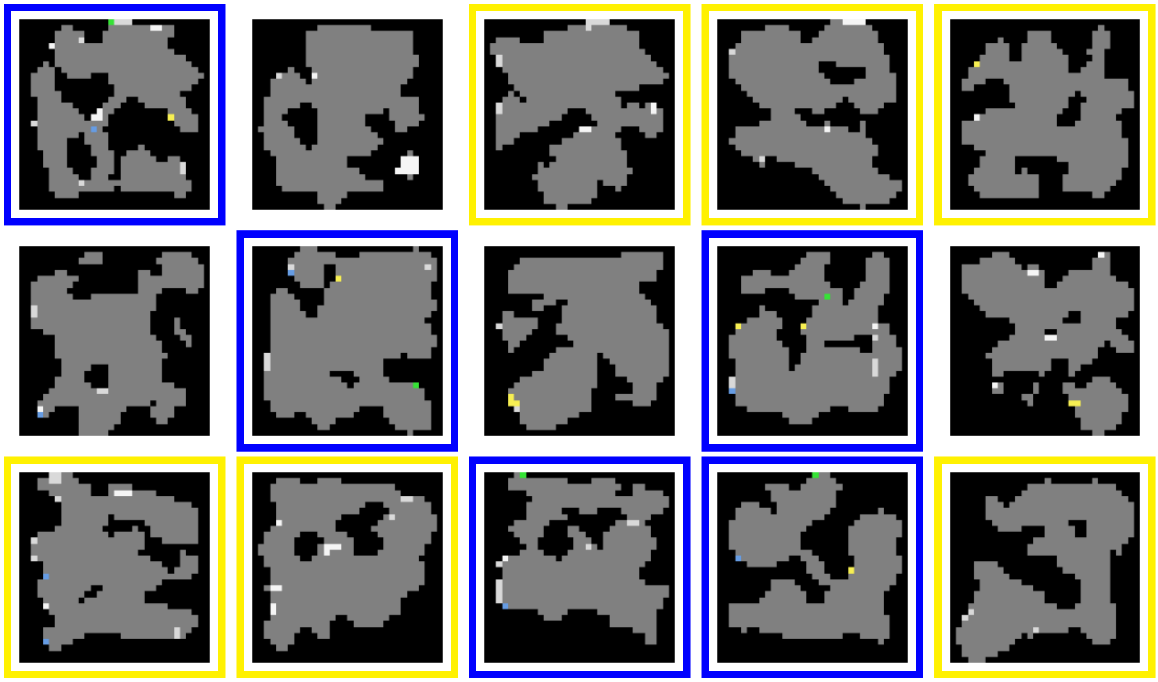
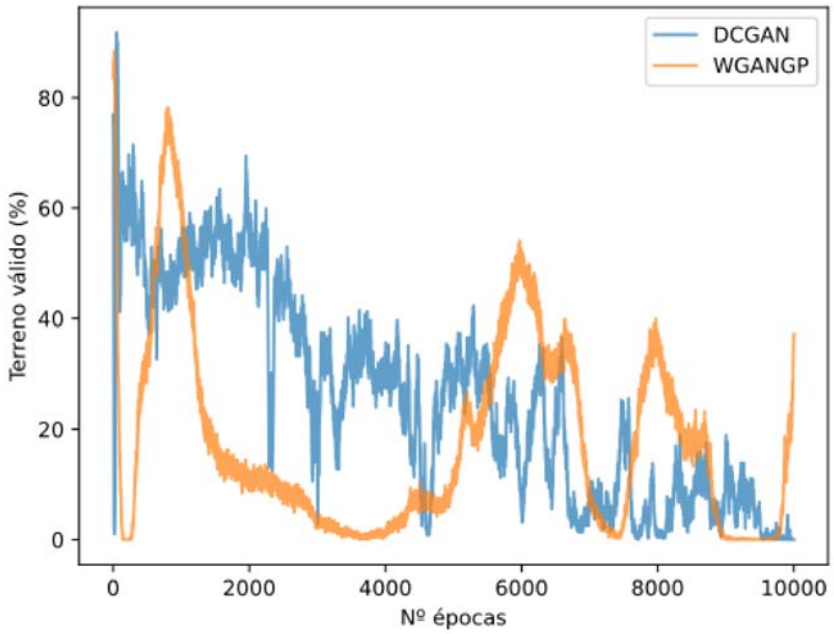


Figura 34 – Comparação das abordagens DCGAN e WGANGP com o *learning rate* 0.00009 do otimizador Adam. O eixo y refere-se ao percentual de mapas que atendem ao critério de chão totalmente conectado da métrica de qualidade proposta.



Quanto ao tempo de treinamento, observou-se que a arquitetura WGAN obteve 7,4% de níveis válidos com 7.466 épocas de treinamento e se manteve estável por mais de 10.000 épocas. No entanto, DCGAN e WGANGP apresentaram instabilidade quando treinadas por mais de 3.000 épocas. Esses modelos também sofreram *over-*

fitting para um comportamento com pouca variabilidade em 4.147 e 5.626 épocas, respectivamente.

Analisando o treinamento até 4.000 épocas, a DCGAN obteve 8,7% de níveis válidos após 594 épocas de treinamento, e o WGANGP obteve 8,6% de amostras válidas com 504 épocas de treinamento. Além disso, observou-se que o treinamento utilizando redes convolucionais conseguiu aprender melhor a distribuição do terreno, com configurações ligeiramente mais variadas de paredes internas, conforme mostrado na Figura 13.

Miyato et al. (2018) propõem uma técnica para estabilizar o processo de treinamento em GANs chamada *Spectral Normalization* (SN). Essa técnica controla a norma espectral das matrizes de pesos nas camadas convolucionais, a fim de estabilizar o treinamento. A normalização no discriminador limita o maior valor singular da matriz de pesos, evitando que ele aprenda funções excessivamente sensíveis e domine o gerador. Portanto, o segundo resultado avalia o impacto de substituir as camadas de BN por SN.

O uso das camadas SN resultou em um atraso na convergência dos modelos. Por exemplo, as arquiteturas DCGAN-SN, WGAN-SN e WGANGP-SN levaram 9.407, 3.555 e 4.185 épocas, respectivamente, para alcançar os valores apresentados na Tabela 10.

Embora seja uma abordagem menos otimizada, os resultados das gerações apresentaram melhoria visual. Em uma avaliação empírica, os níveis gerados foram semelhantes aos dados-alvo. Observou-se que as arquiteturas de GANs com SN são menos suscetíveis ao *mode-collapse* (quando comparadas ao uso de BNs), conforme mostrado na Figura 15.

Outro problema encontrado na arquitetura com BN foi que os elementos tendiam a permanecer próximos às paredes. As amostras geradas a partir do treinamento das GANs com SN apresentaram os elementos mais distribuídos esparsamente pelo terreno. Acredita-se que o SN lide melhor com a distribuição dos elementos, ou seja, fazendo com que canais com menos ocorrências (jogador, portal, inimigos e moedas) tenham a mesma importância que os canais de chão e parede.

Uma vantagem das redes convolucionais é a flexibilidade de usar o gerador para criar níveis de diferentes tamanhos, ajustando somente os vetores latentes para diferentes resoluções. Experimentos foram realizados com o mesmo modelo treinado anteriormente, com o SN e utilizando amostras de tamanho 32×32, mas testados para gerar níveis de 64×64. Embora os resultados não tenham sido ótimos, como mostra a Tabela 12, o modelo conseguiu gerar níveis com uma boa distribuição nos elementos de jogo. A Figura 19 ilustra os resultados desse experimento, comparando-os com os dados obtidos para a resolução de 32×32. Isso sugere que o problema pode ser ajustado ou que os critérios de validade precisam ser adaptados, a fim de não penalizar

excessivamente a avaliação.

Apesar do baixo valor das métricas de jogabilidade, todos os modelos avaliados para uma saída de tamanho 64×64 aprenderam rapidamente a gerar terrenos com topologias válidas (métricas de validade explicadas na Seção 3.2). Em menos de 1.000 épocas, alcançaram valores entre 80% e 90% para terrenos válidos. Empiricamente, observou-se nas gerações da VanillaGAN que os terrenos tendem a ser mais largos (Figura 10), enquanto nas arquiteturas convolucionais é possível observar algumas ocorrências de corredores (Figura 13).