UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Gabriel Costa Backes

GERAÇÃO DE MODELOS DE ELEVAÇÃO DIGITAL A PARTIR DE ESBOÇOS TOPOGRÁFICOS UTILIZANDO REDES GENERATIVAS ADVERSÁRIAS

Santa Maria, RS
2019

Gabriel Costa Backes

GERAÇÃO DE MODELOS DE ELEVAÇÃO DIGITAL A PARTIR DE ESBOÇOS TOPOGRÁFICOS UTILIZANDO REDES GENERATIVAS ADVERSÁRIAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Mestre em Ciência da Computação**.

Orientador: Prof. Dr. Cesar Tadeu Pozzer

Santa Maria, RS

2019

Backes, Gabriel Costa

Geração de Modelos de Elevação Digital a partir de Esboços Topográficos Utilizando Redes Generativas Adversárias / por Gabriel Costa Backes. – 2019.

57 f.: il.; 30 cm.

Orientador: Cesar Tadeu Pozzer

Dissertação (Mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Pós-Graduação em Ciência da Computação , RS, 2019.

1. Síntese de Terrenos por Esboço. 2. Redes Generativas Adversárias. 3. Extração Topográfica. I. Pozzer, Cesar Tadeu. II.Geração de Modelos de Elevação Digital a partir de Esboços Topográficos Utilizando Redes Generativas Adversárias .

E-mail: gbackes@inf.ufsm.br

Gabriel Costa Backes

GERAÇÃO DE MODELOS DE ELEVAÇÃO DIGITAL A PARTIR DE ESBOÇOS TOPOGRÁFICOS UTILIZANDO REDES GENERATIVAS ADVERSÁRIAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Mestre em Ciência da Computação**.

Aprovado em 17 de 07 de 2019:

Cesar Tadeu Pozzer, Dr. (UFSM)
(Presidente/Orientador)

Daniel Welfer, Dr. (UFSM)

Vinicius Da Costa de Azevedo, Dr. (ETH Zurich) (Videoconferência)

Santa Maria, RS

2019

RESUMO

GERAÇÃO DE MODELOS DE ELEVAÇÃO DIGITAL A PARTIR DE ESBOÇOS TOPOGRÁFICOS UTILIZANDO REDES GENERATIVAS ADVERSÁRIAS

AUTOR: GABRIEL COSTA BACKES
ORIENTADOR: CESAR TADEU POZZER

Apesar de amplamente explorada, a criação de terrenos virtuais realísticos em tempo real ainda é um desafio. O avanço do poder computacional por meio das unidades gráficas de processamento permitiu a ascensão das redes neurais artificiais. Este trabalho apresenta um novo método para sintetização de terrenos de forma interativa utilizando redes generativas adversárias. Modelos de elevação digital do mundo e suas respectivas topografias são utilizados para o treinamento da rede sintetizadora. A solução proposta possibilita a geração, em tempo real, de terrenos realísticos de alta resolução que seguem a natureza de esboços topográficos inseridos pelo usuário.

Palavras-chave: Síntese de Terrenos por Esboço. Redes Generativas Adversárias. Extração Topográfica.

ABSTRACT

GENERATION OF DIGITAL ELEVATION MODELS FROM TOPOGRAPHIC SKETCHES USING GENERATIVE ADVERSARIAL NETWORKS

AUTHOR: GABRIEL COSTA BACKES
ADVISOR: CESAR TADEU POZZER

Although extensively exploited, creating terrain in real time is still a challenge. The advancement of computational power through graphics processing units arose the artificial neural networks. This work presents a new method for synthesizing terrain interactively using generative adversary networks. Digital Elevation Models of the world and their respective topographies are used for the synthesizer network's training. The proposed solution enables the real-time generation of realistic high-resolution terrains that follow the nature of the topographical sketches inputs.

Keywords: Sketch Based Terrain Synthesis. Generative Adversarial Network. Topographic Extraction.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

DEM	Modelo de Elevação Digital
GPGPU	Unidade de Processamento Gráfico de Propósito Geral
RNA	Rede Neural Artificial
FNN	Rede Neural Feedfoward
RNC	Rede Neural Convolucional
GAN	Rede Generativa Adversária
cGAN	Rede Generativa Adversária Condicional

SUMÁRIO

1 INTRODUÇÃO

A modelagem de terreno virtual é um campo amplamente explorado devido à sua importância na indústria, mais especificamente em filmes, jogos e simuladores. Em aplicações atuais, é imperativo que as paisagens sejam visualmente atraentes e detalhadas, cabendo ao designer a tarefa de modelar esses cenários. Contudo, modelar um terreno manualmente sem auxílio de ferramentas que permitam a edição em alto nível é uma tarefa exaustiva, tornando-se impraticável em cenários extensos. Em vista disso, existem diversas aplicações que buscam facilitar esse processo, parametrizando possíveis características topográficas e aumentando o nível de abstração. As principais técnicas utilizadas nessas aplicações podem ser categorizadas em: procedurais, baseadas em simulação, baseadas em esboço e exemplo.

Os relevos têm, em sua natureza, fortes características fractais e, ainda que os métodos procedurais sejam eficientes para sua geração, estes trazem consigo uma limitação sobre o controle dos detalhes gerados. As abordagens baseadas em simulação geram paisagens muito similares as encontradas no mundo real. Para isso, o terreno é deformado a partir da simulação do processo de intemperismo, constituída de cálculos de erosão, decomposição ou sedimentação. Entretanto, estas técnicas exigem longos períodos de processamento, inviabilizando seu uso para aplicações interativas. Por último, temos os métodos baseados em esboço e exemplo, que oferecem ao usuário um alto nível de controle sobre os detalhes a serem esculpidos.

O progresso recente no *Deep Learning* forneceu soluções para muitos problemas complexos nas áreas de Computação Gráfica e Visão Computacional, não apenas na tarefa de classificação, mas também na síntese em um contexto geral. Denominados de modelos generativos, existem redes capazes de sintetizar amostras a partir da aprendizagem de características naturais provenientes de um conjunto de dados. Um dos mais poderosos métodos generativos é a chamado Rede Generativa Adversária (GAN) (GOODFELLOW et al., 2014). Atualmente, GANs são utilizadas para os mais diversos objetivos, desde tradução de imagens (ISOLA et al., 2017), geração de rostos foto realísticos (WANG et al., 2018), criação de textos (GUO et al., 2018), até criação de música (YANG; CHOU; YANG, 2017).

Neste trabalho, propomos o uso de uma GAN para criar um sintetizador de terreno. Por meio do esboço das características presentes em um relevo, como rios e montanhas — o que chamamos de esboço topográfico —, é possível esculpir o terreno. A saída do sintetizador está diretamente relacionada aos dados utilizados na etapa de treinamento da rede. Portanto, podem

ser reproduzidas quaisquer regiões do globo terrestre, como vales e montanhas, similares aos encontrados nos alpes suíços, ou longos precipícios, como vistos na região do *Grand Canyon*.

A convergência do sintetizador depende de inúmeras variáveis, como: a arquitetura da rede; a qualidade das imagens do banco de dados; e os valores dos diversos hiper parâmetros (taxa de aprendizado, tamanho dos *batchs*, quantidade de *epochs*, ...). O presente trabalho foi inspirado pelo trabalho de GUÉRIN et al. (2017), sendo o precursor na utilização de redes neurais para geração de terrenos. Baseamos a arquitetura do nosso sintetizador no trabalho de WANG et al. (2018), que apresentou resultados consistentes na sintetização de imagens foto realísticas de alta resolução. Nosso banco de dados é composto de dados de elevação da região dos alpes suíços, obtidos a partir do domínio público *United States Geological Survey*. O pré processamento dos dados para extração da topografia do terreno foi realizado por um algoritmo baseado no método D8 (JENSON; DOMINGUE, 1988).

1.1 CONTRIBUIÇÕES

As contribuições deste trabalho são:

- Algoritmo de extração de estruturas topológicas para execução em GPGPU;
- Sintetização de terrenos realísticos a partir de esboços;
- Definição de uma arquitetura da GAN otimizada para sintetização de terrenos;
- Edição interativa de terrenos de grande escala.

1.2 ESTRUTURA DA DISSERTAÇÃO

O Capítulo 2 apresenta uma revisão sobre terrenos e a estrutura digital de suas informações, unidades de processamento gráfico de propósito geral e redes neurais. No Capítulo 3 são apresentados alguns trabalhos nas áreas de: geração de terrenos, extração topográfica e redes neurais. O Capítulo 4 provê a metodologia utilizada neste trabalho. Os Capítulos 5 e 6 descrevem, respectivamente, a etapa de construção do banco de dados e os detalhes do sintetizador, seguidos pela discutição dos resultados obtidos no Capítulo 7. Por fim, no Capítulo 8 são apresentadas as conclusões e possíveis direções para pesquisas futuras.

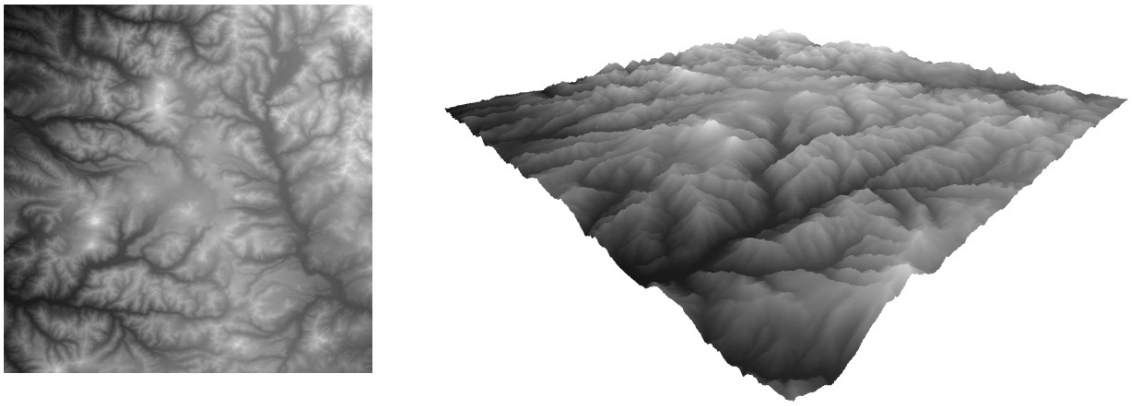
2 FUNDAMENTAÇÃO TEÓRICA

Apresentamos capítulo uma revisão sobre técnicas para representação e armazenamento de terrenos, que são fundamentais para o desenvolvimento do presente trabalho. Abordamos brevemente as Unidades de Processamento Gráfico de Propósito Geral (GPGPUs), nas quais foram implementados os métodos de extração da topografia dos terrenos e onde é executado a etapa de treinamento da GAN. Por fim, tratamos sobre as redes neurais, discorrendo seus princípios matemáticos, assim como a generalização de sua arquitetura e suas derivações.

2.1 TERRENOS

Terrenos geralmente são representados pelos dados de elevação das regiões que o compõe, em uma estrutura denominada Modelo de Elevação Digital (*Digital Elevation Model* - DEM). Os dados de elevação são armazenados em uma matriz, organizada em linhas e colunas, espaçadas em intervalos regulares, caracterizando uma estrutura denominada como *raster*. O tamanho das células da matriz representa a resolução do *raster*, quanto maior a área coberta pela célula, menor sua resolução. Em GPU, essa matriz é representada por uma textura, onde os pixels dessa textura representam as células da matriz, formando uma imagem que pode ser visualizada em tons de cinza (Figura 1), onde cores mais escuras ou mais claras correspondem, respectivamente, a menores ou maiores elevações. Neste trabalho, DEMs também são referidos como mapas de altura.

Figura 1 – Modelo de Elevação Digital: representação em formato *raster* (esquerda) e sua renderização 3D (direita).



Tendo inúmeras aplicações em cartografia, sistemas de informação geográficas, modelagem hidrológica, e na análise geomorfológica, os DEMs estão presentes em diversas áreas de pesquisa. DEMs podem ser obtidos através de bancos de dados públicos, como é o caso do *United States Geological Survey* (USGS). Dentre os diversos DEMs, a USGS disponibiliza o *Space Shuttle Radar Topography Mission* (SSRTM), um DEM desenvolvido pela *National Aeronautics and Space Administration* (NASA) no início dos anos 2000, contendo a elevação de toda superfície terrestre com uma resolução de 30 metros. Foram extraídos dessa base de dados os DEMs utilizados na etapa de treinamento da GAN.

2.2 UNIDADES DE PROCESSAMENTO GRÁFICO DE PROPÓSITO GERAL

Unidades de processamento gráfico (GPUs) emergiram ao final da década de 90, com o avanço da computação gráfica (TSUTSUI; COLLET, 2013). Inicialmente, seu foco era a implementação do pipeline de renderização 3D, onde eram executadas operações como projeção e cálculos de iluminação. Com o crescimento das necessidades dos processos de renderização, surgiram os *shaders*, que permitiram a programação de certas etapas do pipeline de renderização. As primeiras a se tornarem programáveis foram os cálculos de vértices e pixels, permitindo a customização do processo antes da rasterização.

Unidades de processamento gráfico de propósito geral (GPGPUs) foram introduzidas no início do século XXI, com a nova arquitetura da NVIDIA G80 (TSUTSUI; COLLET, 2013). GPGPUs podem oferecer uma maneira poderosa e econômica de melhorar drasticamente a velocidade de simulações computacionais. GPUs têm um grande número de núcleos e executam um número maior de tarefas. Essas tarefas são representadas por *threads*, que contém as instruções, contadores, o contexto da memória e os registradores com os quais trabalha.

A performance potencial de uma abordagem GPGPU está diretamente ligada ao entendimento do algoritmo e sua auto dependência, e os detalhes da implementação em baixo nível, como o gerenciamento de memória e tarefas computacionais. Desta forma, o aumento de performance de uma aplicação em CPU para GPU está diretamente ligada a possibilidade de paralelização do algoritmo, assim como a quantidade de computações que são realizadas em paralelo.

Uma das maneiras de implementar uma aplicação em GPGPU é através de *compute shaders*. Diferentemente dos estágios do pipeline de renderização que executam periodicamente, os *compute shaders* são baseados na demanda e despachados explicitamente pela aplicação.

As entradas e saídas de um *compute shader* são genéricas, permitindo computações de propósito geral na GPU. Os resultados do processamento de um *compute shader* podem ser tanto acessados pelos estágios padrões de renderização da GPU, quanto lidos para a CPU.

É intrínseco as redes neurais a execução de computações numéricas em profusão, que, por sua vez, podem ser acelerados através do uso de GPGPU. CUDA é uma plataforma desenvolvida pela NVIDIA destinada à computação paralela (HARRIS et al., 2007). Sobre ela foi desenvolvida a biblioteca *NVIDIA CUDA® Deep Neural Network (cuDNN)*, que contém as primitivas para as redes neurais, fornecendo implementações altamente otimizadas de estruturas como convoluções, *pooling*, normalização e funções de ativação. Atualmente, frameworks para desenvolvimento de redes neurais dependem da cuDNN para aceleração em GPU, como o TensorFlow (ABADI et al., 2016) e PyTorch (PASZKE et al., 2017).

2.3 REDES NEURAIS

A capacidade do ser humano de realizar ações, como se locomover, falar ou realizar cálculos matemáticos, está vinculada a complexa estrutura do cérebro. O cérebro humano é formado por múltiplas camadas de neurônios que interagem entre si. Esta interação ocorre por meio de uma entrada, vinda de neurônios em uma camada anterior, que é processada e enviada para neurônios em uma camada posterior, processo este conhecido como sinapse. Na literatura, essa arquitetura foi replicada em computadores e denominada Rede Neural Artificial (RNA) (BASHEER; HAJMEER, 2000) e atualmente é utilizada para tarefas como reconhecimento de fala (HINTON et al., 2012), tradução de textos (SUTSKEVER; VINYALS; LE, 2014), geração de imagens (WANG et al., 2018) e até na realização de diagnósticos médicos (AMATO et al., 2013).

Os neurônios artificiais são a base de uma RNA. Esses descrevem um modelo matemático inspirado pelos neurônios no nosso cérebro. Sua função é receber diversas entradas $x = x1, x2, ..., xn$ e computar uma soma ponderada z utilizando pesos $w = w1, w2, ..., wn$. Esta soma ponderada z é a transformação das entradas do neurônio. Além dos pesos, também é utilizado um bias b a essa soma, e o resultado passa então por uma função de ativação σ, resultando na saída final a. Esta equação é descrita por:

$$a = \sigma(\sum_{i=0}^{n} x_i w_i + b) = \sigma(z) \tag{2.1}$$

A escolha da função de ativação σ é um aspecto importante no design de uma RNA. Por meio dela é realizada a conversão de um sinal de entrada para um sinal de saída, definindo as conexões entre os neurônios artificiais. Além disso, os gradientes provenientes da mesma são utilizados na atualização dos valores de w e b, permitindo a realização da técnica conhecida como *backpropagation*. A proposta principal da função de ativação é introduzir uma não linearidade para saída do neurônio, possibilitando o aprendizado de tarefas complexas. As funções de ativação usadas mais frequentemente são:

- Sigmoid. Sendo um caso especial de função logística, a função Sigmoid tem como característica o formato em 'S'. Essa função é frequentemente utilizada em modelos de predição de probabilidade (Figura 2.a);

- Tahn. A tangente hiperbólica é semelhante a Sigmoid, entretanto, sua saída é limitada no intervalo [-1,1], centralizada em 0. Sua principal utilização é em modelos de classificação entre 2 classes (Figura 2.b);

- ReLU. A função de Unidade Linear Retificada simplesmente limita inferiormente a saída para 0. Sua grande vantagem sobre as demais funções é sua eficiência, tendo um menor custo computacional. Essa função também acelera a convergência do gradiente descendente quando comparada com as anteriores (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) (Figura 2.c);

- textitLeaky ReLU. Um dos problemas da função ReLU é a perda do gradiente ao longo do treinamento, uma vez que a mesma zera os valores negativos. Ao invés disso, a função *Leaky* ReLU apresenta um pequeno *slope* no intervalo negativo (Figura 2.d).

A Figura 3 ilustra a estrutura de um neurônio artificial. Por meio de diferentes pesos w e bias b, o neurônio pode ser treinado a aproximar uma função dado o conjunto de entrada x. A configuração desses valores é realizada na etapa de treinamento da rede neural, que utiliza métodos de aprendizagem baseados em gradientes.

A Figura 4 ilustra a generalização da arquitetura de uma rede neural. A camada a esquerda é denominada camada de entrada, e os neurônios nessa camada são denominados neurônios de entrada. A camada mais a direita representa a camada de saída. Nesse exemplo, a saída é formada por um único neurônio. As camadas internas da rede são chamadas de camadas ocultas. Em uma rede neural, podem existir números arbitrários de camadas e em cada uma

Figura 2 – Funções de ativação: fórmula numérica seguida de sua representação gráfica.

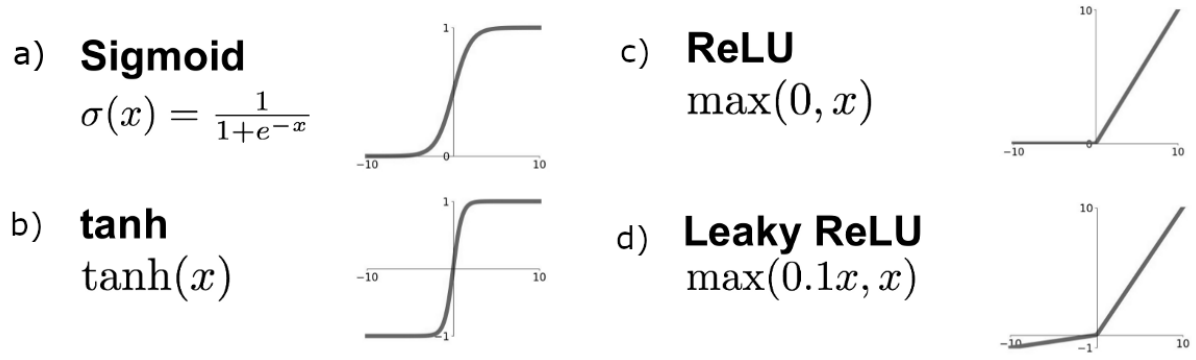
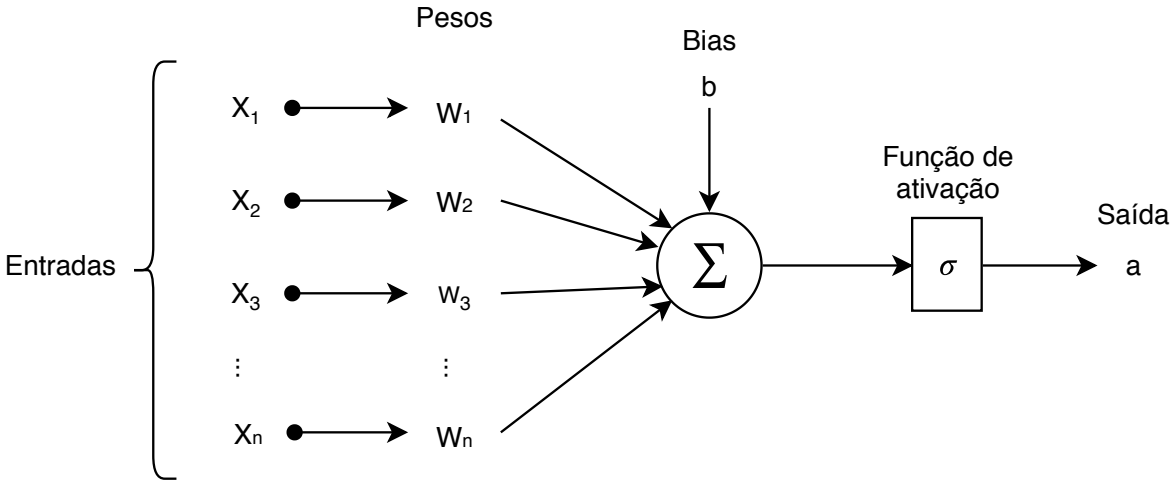


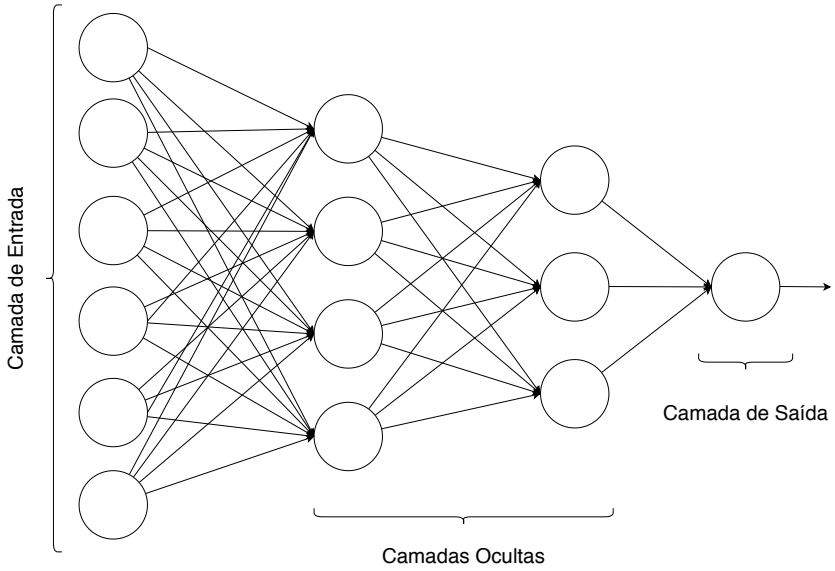
Figura 3 – Arquitetura de um neurônio artificial.



podem haver diferentes números de neurônios. A quantidade de camadas e seus respectivos tamanhos variam de acordo com o propósito da aplicação, e geralmente são definidos de maneira empírica. Redes que não contém *loops* entre as camadas, ou seja, as informações são sempre passadas para camadas posteriores, são chamadas de Redes Neurais *Feedforward* - (FNN).

A otimização de uma FNN é constituída pela busca automática dos pesos e bias para que uma determinada rede convirja para uma saída desejada. A otimização também é referida como a tarefa de minimizar uma determinada função f. Essa função é chamada de função de custo, na literatura também é denominada de função de perda, função de erro, ou função objetiva. f mede o quão errado um modelo está, referente a sua habilidade de estimar uma relação entre uma entrada x a uma saída y. A técnica utilizada para realizar a minimização de $f(x)$ é conhecida como gradiente descendente.

Figura 4 – Generalização da arquitetura de uma FNN.



2.3.1 Gradiente Descendente

Dado uma função de custo f, o objetivo do gradiente descendente é encontrar o mínimo local para $f(x)$. Enquanto para funções simples é possível computar o mínimo de forma analítica, para funções complexas com múltiplos parâmetros, como as funções das FNNs, isto se torna impraticável. Sendo assim, o gradiente descendente é uma abordagem numérica onde são escolhidos valores iniciais aleatórios e iterativamente a função segue em direção ao maior gradiente descendente. A direção do maior gradiente é dada pelo cálculo do gradiente negativo da função de custo do conjunto g de parâmetros θ na posição i:

$$- g_{\theta_i} = -\nabla_\theta f(\theta_i) = -\frac{\partial E}{\partial \theta_i} \tag{2.2}$$

Definindo um fator positivo para controlar a magnitude do gradiente descendente, chamada de taxa de aprendizagem η, a regra de atualização do gradiente descendente para a FNN de parâmetros x é definida como:

$$\theta_{i+1} = \theta_i - \eta g_{\theta_i} \tag{2.3}$$

Em uma FNN, utilizamos uma entrada x para produzir uma saída $\hat{y}$, onde a informação é passada adiante pela rede. A entrada x contém as informações que serão responsáveis por ativar os neurônios a medida que é propagado pelas camadas ocultas da rede, que por fim, produzem $\hat{y}$. Chamamos esse processo de *forward propagation*. Durante a etapa de treinamento da rede, em

cada iteração é executado o *forward propagation*, que, utilizando uma função de custo, resulta em um erro E. Para a atualização da rede, é necessário que as informações deste erro sejam propagadas de volta pela rede, permitindo o cálculo dos gradientes. Este processo é conhecido como *backpropagation*.

2.3.2 *Backpropagation*

Backpropagation é a essência do treinamento das redes neurais. Os pesos da rede são ajustados de acordo com um erro obtido a partir de uma função de custo, de forma que a rede aprenda a generalização da distribuição dos dados de entrada. Utilizando a regra da cadeia do cálculo numérico, são computados, de maneira iterativa, as derivadas parciais das funções para cada peso da rede.

Sendo x um número real, e f e g funções que mapeiam um número real para outro número real, supondo que $y = g(x)$ e $z = f(g(x)) = f(y)$, então, segundo a regra da cadeia

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}. \tag{2.4}$$

Supondo que $x \in \mathbb{R}^m$, $y = \mathbb{R}^n$, g mapeia de $\mathbb{R}^m$ para $\mathbb{R}^n$, e f mapeia de $\mathbb{R}^n$ para $\mathbb{R}$. Se $y = g(x)$ e $z = f(y)$, então

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}. \tag{2.5}$$

De forma equivalente, em notação vetorial, temos

$$\nabla_{x^z} = (\frac{\partial y}{\partial x})^T \nabla_{y^z}, \tag{2.6}$$

onde $\frac{\partial y}{\partial x}$ é a matriz Jacobiana $n \times m$ de g.

Desta forma, o gradiente de uma variável x pode ser obtido multiplicando a matriz Jacobiana $\frac{\partial y}{\partial x}$ pelo gradiente ∇_{y^z}. O algoritmo de *backpropagation* consiste em realizar o produto do gradiente Jacobiano para cada peso presente na rede (GOODFELLOW; BENGIO; COURVILLE, 2016).

Considerando um grafo computacional, onde cada nodo do grafo representa uma variável, e resulta em um escalar $u^{(n)}$. Este escalar é o valor do gradiente que queremos obter, com relação aos (n_i) nodos de entrada $u^{(1)}$ até $u^{(n_i)}$. Em outras palavras, queremos computar $\frac{\partial u^{(n)}}{\partial u^{(i)}}$

para todo $i \in 1, 2, ..., n_i$. Na aplicação do *backpropagation* para computar os gradientes, $u^{(n)}$ é o erro associado a uma iteração, enquanto $u^{(1)}$ até $u^{(n_i)}$ corresponde aos parâmetros do modelo.

Cada nodo $u^{(i)}$ está associado a uma operação $f^{(i)}$ e é computado avaliando a função

$$u^{(i)} = f(\mathbb{A}^{(i)}), \tag{2.7}$$

onde $\mathbb{A}^i$ é o conjunto de nodos antecessores e conectados a $u^{(i)}$.

Especificamos desta maneira a computação do *forward propagation*. Para realizar o *backpropagation*, computamos exatamente o procedimento reverso, para cada nodo $u^{(i)}$ computamos sua derivada associada $\frac{\partial u^{(n)}}{\partial u^{(i)}}$. Isso é feito através da regra da cadeia em relação ao escalar de saída $u^{(n)}$:

$$\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i:j \in P_a(u^{(i)})} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}} \tag{2.8}$$

O produto escalar é realizado, para cada nodo, entre o gradiente já computado com relação à $u^{(i)}$ que são filhos de $u^{(j)}$ e o vetor que contém as derivadas parciais $\frac{\partial u^{(i)}}{\partial u^{(j)}}$ para os mesmos nodos filhos $u^{(i)}$.

2.3.3 Redes Neurais Convolucionais

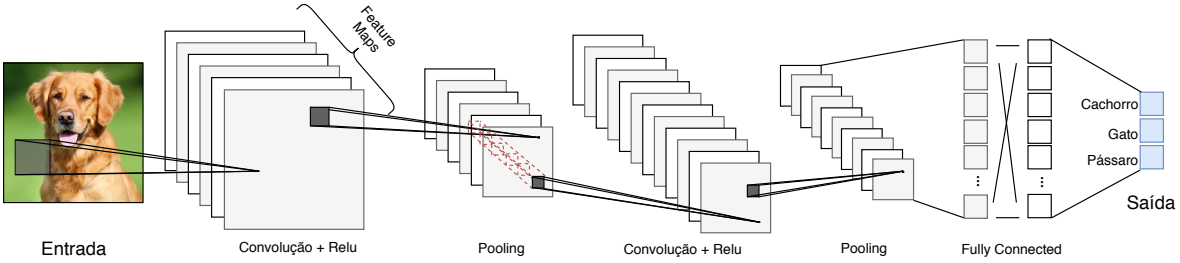
A maneira com que os neurônios são processados pelas redes neuras convencionais faz com que haja um grande custo de memória, uma vez que cada neurônio em uma determinada camada está conectado com todos os neurônios da camada subsequente, intitulada *fully-connected*. O que por sua vez inviabiliza o uso de RNA em aplicações constituídas de grandes quantidades de parâmetros. Redes Neurais Convolucionais (RNC) (LECUN et al., 1989) são uma extensão para as redes neurais convencionais, especializadas no processamento de dados representados por uma topologia em grid. Assim sendo, ao invés de realizar a conexão de todos os neurônios entre duas camadas, nas RNC são executadas convoluções. A Figura 5 ilustra uma generalização da arquitetura de uma RNC.

Para uma imagem bidimensional I, a convolução discreta é definida por:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n) \tag{2.9}$$

onde $K(m,n)$ corresponde a um *kernel* bidimensional, e a saída $S(i,j)$ se refere ao *feature map*. Essa operação desliza o *kernel* sobre a imagem I e calcula a soma ponderada do *kernel*

Figura 5 – Generalização da arquitetura de uma RNC.



com cada posição i, j. A operação de convolução é facilmente estendida para um número arbitrário de dimensões. Nesse contexto, cada *kernel* K descreve pesos aprendíveis na camada de convolução, e cada camada de convolução pode conter um número arbitrário de *kernels*, cada um resultando em seu próprio *feature map*. A convolução ocorre sempre sobre todos os canais da entrada.

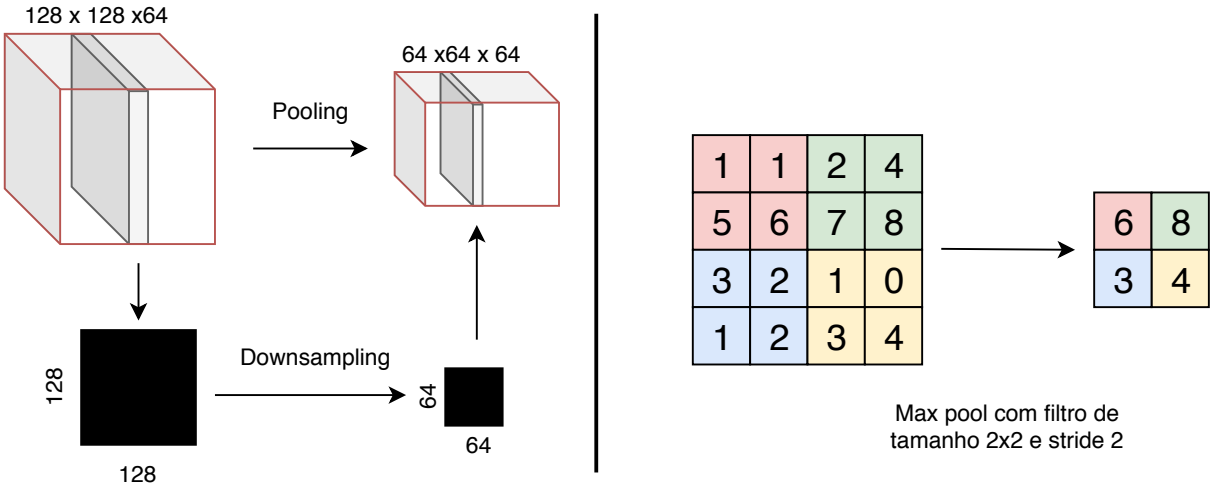
Tipicamente, o *kernel* é muito menor que a imagem em si, resultando numa conexão esparsa entre as entradas e saídas das camadas de convolução. Intuitivamente, a ideia por trás dessa conectividade esparsa nas RNC é que *kernels* pequenos são bons para detectar padrões, como bordas em imagens. Esta conectividade esparsa também define implicitamente o *receptive field* das entradas do *feature map* de saída. O *receptive field* em cada dimensão espacial é definido como o número de entradas em que cada saída respectiva do *feature map* são afetadas durante a convolução.

Para manter a baixa quantidade de parâmetros e aumentar o campo receptivo efetivo das saídas de cada camada, é feito o uso de um *downsampling* espacial, chamado *pooling*, após algumas camadas de convolução na rede. *Pooling* pode ser intuitivamente entendido como um mecanismo de deslizamento de um *kernel*, similar a convolução. A principal diferença é que o *pooling* calcula uma função fixa em suas entradas, uma das operações mais utilizadas é conhecida como *max-pooling*. A sua ideia é gerar um *feature map* contendo os neurônios mais relevantes da camada anterior, a partir do deslizamento de um *kernel* sobre a camada (Figura 6).

2.3.4 Redes Generativas Adversárias

Modelos generativos são modelos que aprendem características naturais de um conjunto de dados e geram novas amostras similares aos dados existentes. O algoritmo generativo tem como princípio aprender uma determinada distribuição de probabilidade conjunta a partir da

Figura 6 – Exemplo de *pooling*: a esquerda tem-se a representação da aplicação do *pooling* sobre uma determinada camada. A direita é ilustrada a execução do *max pooling* sobre uma *feature map*.



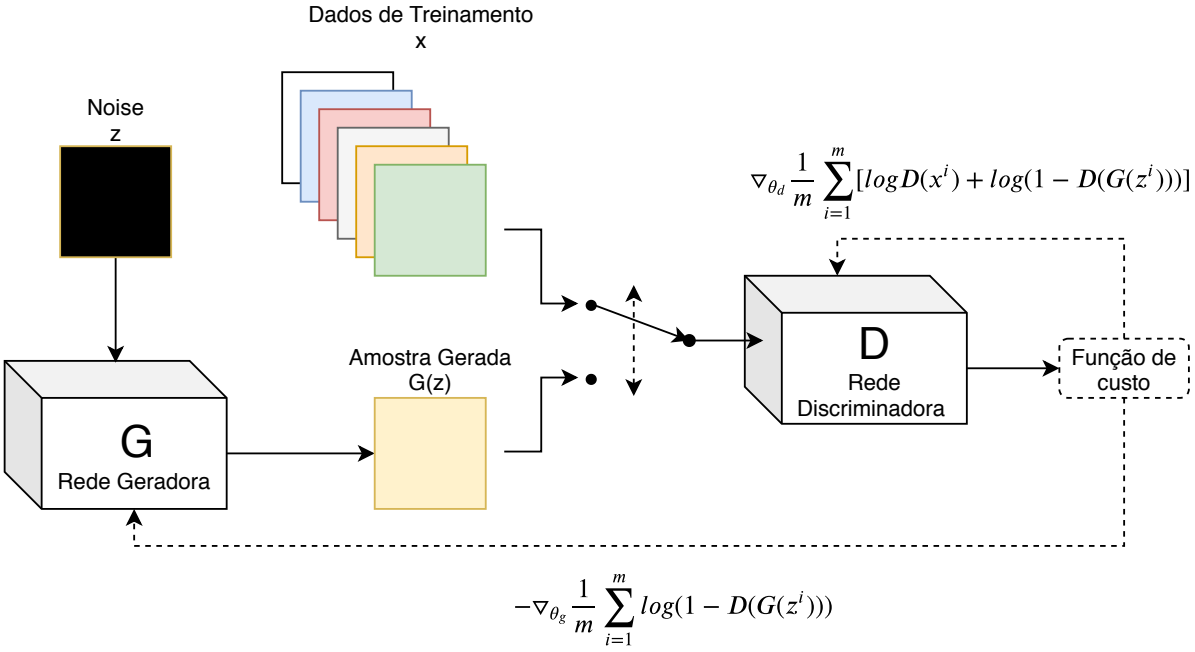
observação dos dados. Dois dos mais eficientes modelos generativos são conhecidos como: Redes Generativas Adversárias e Auto Codificadores Variacionais (Variational Auto Encoders - VAE). Neste trabalho focaremos no primeiro modelo.

GOODFELLOW et al. (2014) propuseram a Rede Generativa Adversária, na qual é um modelo de arquitetura generativa baseada numa rede geradora diferenciável, e é atualmente considerada o estado da arte para modelos generativos. GANs são utilizadas para gerar novas amostrar a partir do aprendizado de dados existentes. GANs são compostas por duas redes, uma rede geradora G e uma rede discriminadora D. A rede geradora G produz novas amostras. A rede discriminadora D é responsável por distinguir entre amostras dos dados de treinamento ou amostras geradas pela rede geradora. A Figura 7 ilustra a arquitetura do modelo.

A rede geradora G recebe como entrada um vetor de *noise* z e gera uma amostra $G(z)$. Conceitualmente, z representa as características da imagem gerada em espaço latente, como os formatos ou cores. Desta forma, não há controle sobre o significado semântico de z. Sendo assim, o objetivo de G na etapa de treinamento é encontrar um significado para o mesmo.

Entretanto, G por si só gera um noise randômico. Temos então a rede discriminadora D, com o objetivo de orientar a rede G na geração de amostras mais similares as reais. D é treinada a distinguir entre amostras reais e sintéticas. Para isso, recebe como entrada tanto as amostras geradas $G(z)$, como dos dados de treinamento x. A saída $D(x)$ representa a probabilidade de x ser um dado real, num intervalo de 0 a 1. Assim, G é direcionada para que sejam geradas

Figura 7 – Arquitetura de uma GAN: a partir de um noise de entrada z, a rede geradora G produz uma amostra $G(z)$. Alterna-se a entrada da rede discriminadora D entre os dados de treinamento x e as amostras $G(z)$. A otimização de ambas as redes é realizada por meio do gradiente resultante de uma função de custo aplicada sobre a saída $D(x)$.



imagens a partir da percepção da rede D sobre os dados reais. Logo, a eficiência da rede G em gerar imagens similares as reais está diretamente relacionada a capacidade de diferenciação de D.

Sabendo que a saída de D é um valor $D(x)$, no qual representa a probabilidade de x ser uma imagem real, temos como objetivo maximizar a chance de reconhecer imagens reais como reais ($D(x) = 1$) e minimizar a classificação de imagens sintéticas como reais ($D(G(z)) = 0$). Ao mesmo tempo, G é treinada para enganar D, gerando imagens o mais similar possível as reais, ou seja, maximizando a possibilidade de D classificar uma imagem sintética como real ($D(G(z)) = 1$). Ao invés de utilizar a probabilidade diretamente, uma maneira mais eficiente para medir o erro é por meio da *cross-entropy* ($p \log(q)$), forçando uma forte penalização sobre uma classificação incorreta.

Podemos concluir que ambas as redes competem entre si, onde a rede G aprende a produzir amostras mais reais, minimizando a probabilidade da rede D identificar as amostras geradas $G(z)$ como sintéticas, enquanto a rede D aprende a distinguir as amostras geradas de dados reais. D e G jogam um *minmax* com a função de custo $V(G, D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]. \quad (2.10)$$

Com os objetivos definidos, o treinamento das redes é feito a partir da aplicação do gradiente descendente alternadamente. Primeiramente é otimizado os parâmetros da rede geradora e realizado uma iteração do gradiente descendente na rede discriminadora usando as imagens reais e sintetizadas. Seguido da otimização dos parâmetros da rede discriminadora e execução de uma iteração da rede geradora. Estas etapas são alternadas até o momento em que a rede geradora produza imagens satisfatórias.

Os dados gerados pelas GANs são decorrentes, em sua versão original, da entrada composta somente pelo espaço latente z. Desta forma, não há controle sobre a saída da rede, já que z é proveniente de uma distribuição randômica. *Conditional GANs* (CGANs) (MIRZA; OSINDERO, 2014) utilizam, de forma complementar a z, uma camada adicional y, que pode ser entendido como um auxiliador para as redes. Incluindo y na função de custo da GAN, temos:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)))]. \quad (2.11)$$

Para que y torne a rede mais eficiente, seu conteúdo deve ter relação com o objetivo da mesma. Por exemplo, caso o objetivo de uma GAN seja a geração de imagens de dígitos no intervalo de 0 a 9, poderíamos representar a entrada condicional y na forma um vetor de 10 posição, que por sua vez, corresponderia a um digito. Para representar o valor 2, o vetor seria definido como (0,1,0,0,0,0,0,0,0,0). y direciona G a gerar um digito específico, ao mesmo tempo que guia D a saber qual elemento está a distinguir.

3 TRABALHOS RELACIONADOS

Neste capítulo são abordados os trabalhos presentes na literatura na área de geração de terrenos virtuais. Discorremos as técnicas desenvolvidas para a extração da topografia de terrenos a partir de DEMs, devido a sua utilização para a construção do banco de dados. Por fim, são apresentadas algumas aplicações e extensões desenvolvidas sobre as GANs.

3.1 GERAÇÃO DE TERRENO

Inúmeras pesquisas na área da geração de terrenos virtuais foram desenvolvidas ao longo das últimas três décadas. Porém, a essência das diversas aplicações que fazem o uso de terrenos virtuais é distinta. Na indústria cinematográfica é fundamental que os terrenos tenham um alto grau de realismo. Sendo assim, desenvolveu-se um ramo na área denominado geração baseada em simulação. Em aplicações como simuladores ou em uma parcela da indústria de jogos, fez-se necessário o controle do artista sobre o cenário de forma detalhada e interativa, originando o ramo conhecido como geração baseada em esboço e exemplo. Outra parcela da indústria de jogos tem como particularidade a geração do terreno em tempo real, onde, por muitas vezes, o mesmo se estende por áreas massivas. Nesses casos, não é necessário que o artista tenha um controle minucioso sobre o terreno, mas é plausível o controle em escala global, caracterizando o terceiro ramo, denominado geração procedural.

3.1.1 Geração Procedural

Os primeiros métodos procedurais foram baseados em subdivisões iterativas de uma malha. FOURNIER; FUSSELL; CARPENTER (1982) apresentaram um método que ficou conhecido como deslocamento do ponto médio, onde o nível de detalhes do terreno é gerado ao se deslocar um novo vértice pela altura média do nível anterior, combinado por um valor aleatório (Figura 8).

Abordagens baseadas em ruído, como o *Perlin Noise* (PERLIN, 1985), foram promissoras devido a seus baixos custos computacionais. Os detalhes na malha são inseridos pela soma de várias oitavas de ruído em diferentes frequências e amplitudes escalonadas. A Figura 9 mostra um exemplo de um terreno gerado a partir de um *Perlin Noise*. De maneira oposta ao método apresentado por FOURNIER; FUSSELL; CARPENTER (1982), os pontos da malha

Figura 8 – Técnica de deslocamento do ponto médio. A malha é refinada ao inserir, subsequentemente, novos vértices entre os já existentes.

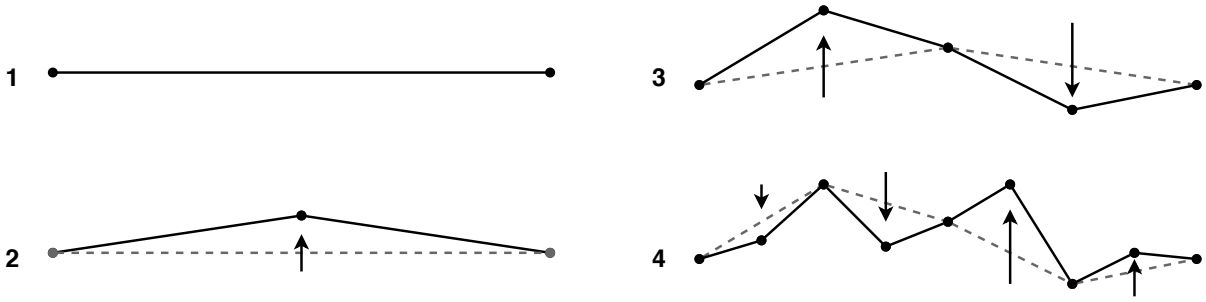
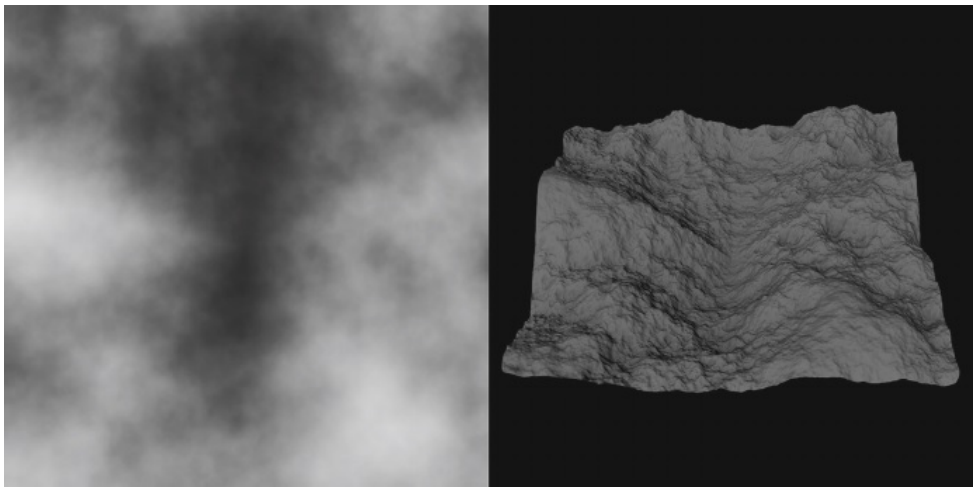


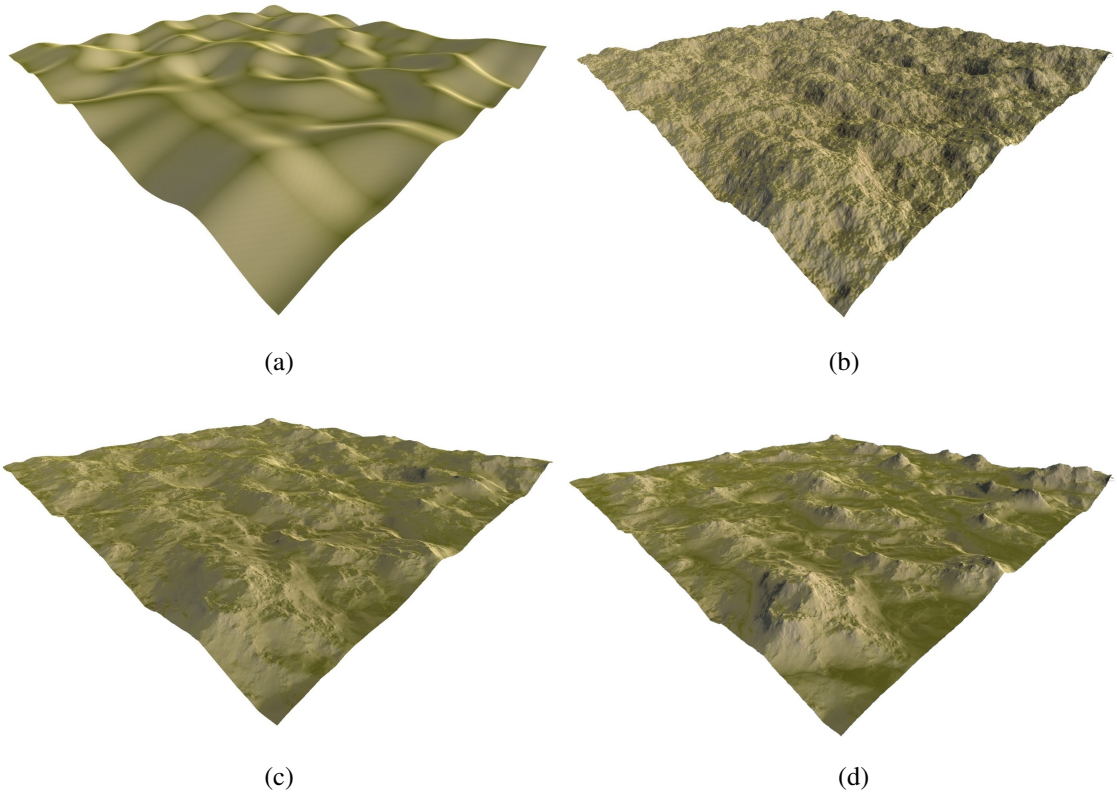
Figura 9 – Terreno gerado a partir de um *Perlin Noise*. O *Perlin Noise* é representado por uma textura com valores normalizados no intervalo [0, 1] (esquerda), juntamente com sua visualização 3D (direita) (ECK; LAMERS, 2015).



são localmente independentes, uma vez que obtidos por meio da interpolação de uma função contínua.

Apesar de produzir características particulares, a distribuição do ruído é regular sobre toda superfície. BACKES; ENGEL; POZZER (2018) propuseram um conjunto de parâmetros, nomeado conjunto de erosão procedural, dando mais controle sobre os detalhes durante a geração. Nele são definidos parâmetros como taxa de erosão por altitude, taxa de erosão em inclinações e em concavidades. Apesar de não ser fisicamente correto, esse conjunto tem o propósito de imitar os efeitos da erosão. A Figura 10 mostra o pipeline de criação de um terreno usando o conjunto de erosão procedural juntamente com o *Perlin Noise*. Apesar de manter o desempenho dos algoritmos procedurais, essa abordagem ainda apresenta a limitação do controle dos detalhes em escala local.

Figura 10 – Pipeline de geração do terreno com o conjunto de erosão procedural: (a) Primeiramente, o terreno base é gerado utilizando um *Perlin Noise* 2D; (b) Os detalhes são introduzidos usando *fractional Brownian motion (fBm)*; (c) As características do terreno são esculpidas usando o conjunto de erosão procedural, proporcionando um aspecto mais realístico; (d) Por último é aplicado um realce sobre as características do terreno.



(a) (b)

(c) (d)

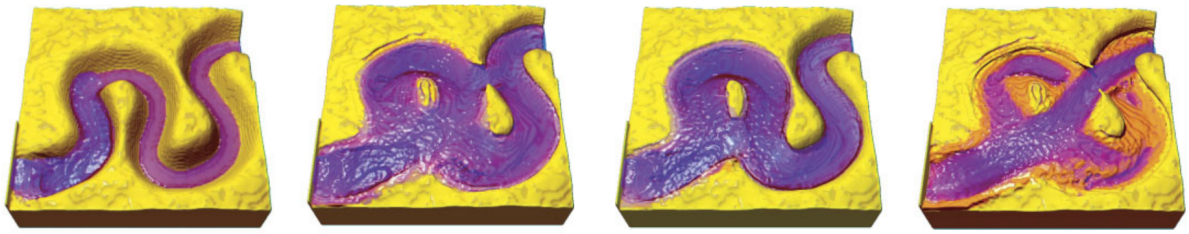
Fonte: (BACKES; ENGEL; POZZER, 2018).

3.1.2 Geração baseada em simulação

Abordagens baseadas em simulação introduziram aspectos mais naturais aos terrenos. Um dos primeiros trabalhos na área foi apresentado por MUSGRAVE; KOLB; MACE (1989). Sua proposta contava com a utilização de algoritmos para a simulação da erosão hidráulica e do intemperismo térmico. Essa abordagem foi estendida em diversas direções, apresentando algoritmos de erosão mais eficientes e corretos (CHIBA; MURAOKA; FUJITA, 1998), (BENEŠ; FORSBACH, 2002), (BENEŠ et al., 2006). A Figura 11 mostra a simulação do processo de erosão hidráulica sobre um terreno. Uma das principais desvantagens encontradas nos métodos baseados em simulação é o alto custo computacional, uma vez que requerem uma grande quantidade de cálculos sobre inúmeras iterações.

VANEK et al. (2011) tentou amenizar os problemas de escalabilidade das abordagens anteriores levando para as GPUs os cálculos matemáticos. A solução encontrada para lidar com

Figura 11 – Simulação da erosão hidráulica.



Fonte: (BENEŠ et al., 2006).

terrenos de grande escala foi dividir o terreno em blocos de diferentes resoluções. Sendo assim, são utilizados diferentes níveis de detalhes para acelerar os cálculos da simulação baseada em física. Apesar de apresentar resultados efetivos, a proposta se mostrou ineficiente em cenas que continham uma grande quantidade de dinâmicas de fluido ou terrenos com variações abruptas de altura.

Mesmo com a inserção das GPUs, as técnicas que se baseiam em simulações físicas ainda sofrem com o problema de escalabilidade (VANEK et al., 2011). Podemos relacionar isto ao fato de que os cálculos trazem em si a dependência local entre as alturas de um terreno. Desta forma, o poder computacional oferecido pelas GPUs não pode ser explorado por completo, devido à impossibilidade do paralelismo.

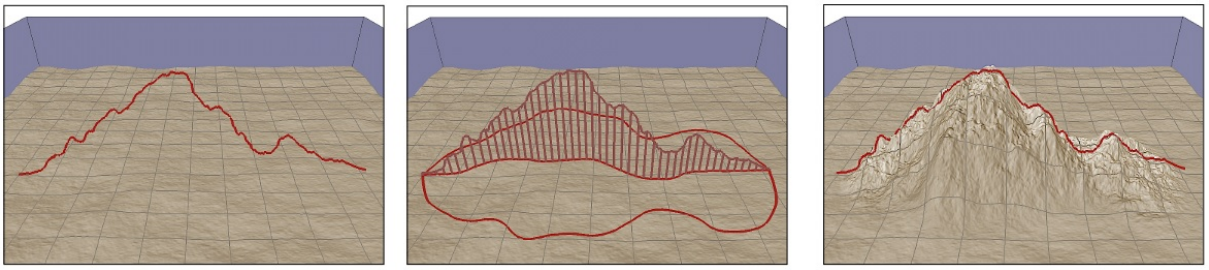
3.1.3 Geração baseada em exemplo e esboço

Métodos baseados em exemplo e esboço conseguem lidar com uma dos principais limitações encontradas na geração procedural: a baixa controlabilidade. Nessa técnica, a geração do terreno é controlada pela intuição do usuário, que insere por meio de uma interface as diferentes características encontradas no terreno. RUSNELL; MOULD; ERAMIAN (2009) descrevem um método de síntese de terreno baseado em distâncias em um grafo ponderado, criado a partir de nós geradores especificados pelo usuário. Esses nós são denominados *strokes*, e representam diferentes primitivas, como montanhas, colinas e crateras.

GAIN; MARAIS; STRASSER (2009) apresentam uma interface para geração procedural do terreno de maneira interativa a partir de desenhos do usuário. Um pouco diferente da interação apresentada no trabalho de (RUSNELL; MOULD; ERAMIAN, 2009), nesse são inseridas pelo usuário as silhuetas e curvas delimitando colinas, montanhas, rios e cânions (Figura 12). O nível de detalhamento do terreno é aumentado por meio da inserção de um noise,

que é interpolado entre as novas formas e o terreno base. Na mesma direção, HNAIDI et al. (2010) propôs um método baseado na difusão de restrições, em que o usuário controla as formas de terreno por meio de curvas de feições, nas quais definem cumes, leito de rios, penhascos e colinas.

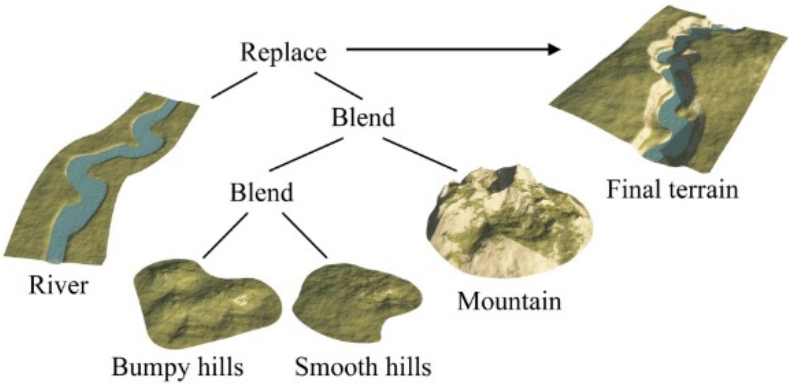
Figura 12 – Modelagem de terrenos por meio de esboço.



Fonte: (GAIN; MARAIS; STRASSER, 2009).

Recentemente, (GÉNEVAUX et al., 2015) apresentou um modelo procedural hierárquico que se baseia em uma árvore de construção. As folhas dessa árvore representam as características do terreno, como montanhas, cumes, vales, rios e lagos. Os nodos internos representam os operadores, que combinam as folhas esculpindo, misturando ou distorcendo as mesmas (Figura 13). Esse trabalho foi estendido por (GUÉRIN et al., 2016), no qual incorporou essa estrutura e propôs uma modelagem procedural inversa genérica para aprender e gerar automaticamente um terreno de grande escala em vez de ser necessário gerar o terreno interativamente a mão.

Figura 13 – Árvore de construção simplificada: esculpimento de um rio em meio a uma colina.

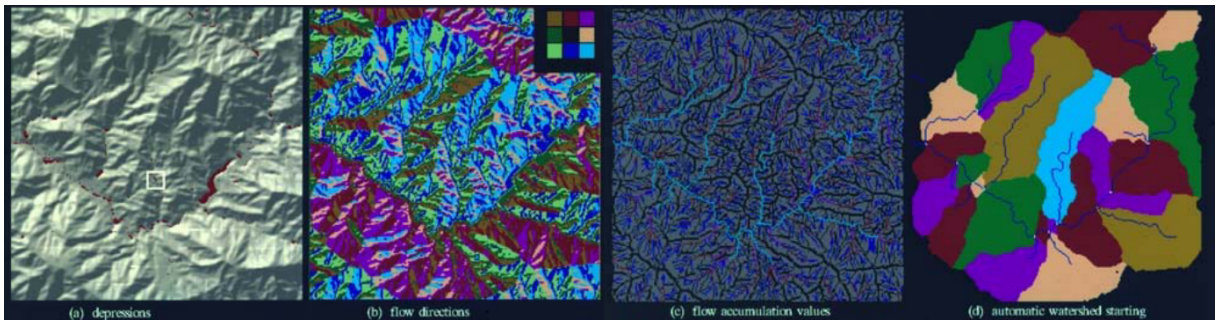


Fonte: (GÉNEVAUX et al., 2015).

3.2 PROCESSAMENTO DE MODELOS DE ELEVAÇÃO DIGITAL

Modelos de elevação digital podem ser utilizados para obter uma grande variedade de informações sobre uma determinada superfície. Bacias hidrográficas e redes fluviais estão diretamente relacionadas a sua inclinação e elevação. JENSON; DOMINGUE (1988) propuseram um algoritmo para a extração dessas características topográficas por meio de uma técnica de vizinhança. Essa técnica tem uma fase inicial condicionante, na qual resulta em três conjuntos de dados que são utilizados nas fases subsequentes. Esses conjuntos de dados são formados por: (a) um DEM com suas depressões preenchidas; (b) uma estrutura contendo a direção de fluxo de cada célula; e (c) uma estrutura contendo os dados de acumulação de fluxo para cada célula. Com esses dados é possível extrair as redes fluviais e bacias hidrográficas (Figura 14).

Figura 14 – Extração de redes fluviais: (a) Áreas destacadas em vermelho representam as depressões a serem preenchidas; (b) Direções de fluxo de cada célula do terreno; (c) Dados de acumulo de fluxo para diferentes intervalos colorizados; (d) Resultado da extração das bacias hidrográficas.



Fonte: (JENSON; DOMINGUE, 1988).

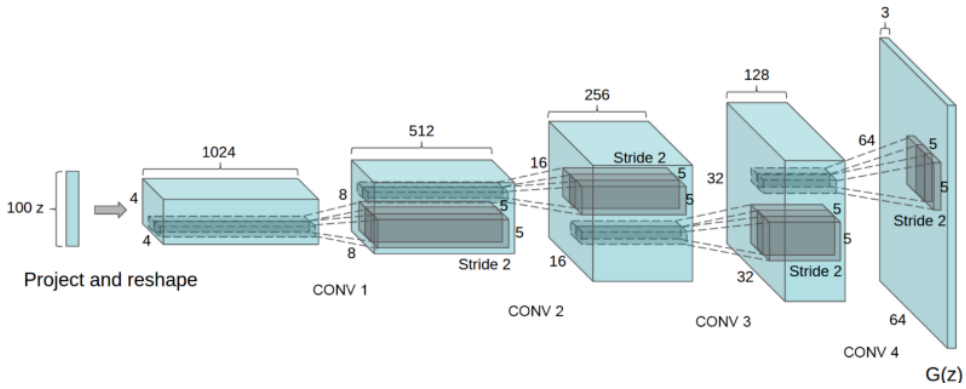
Apesar da técnica de JENSON; DOMINGUE (1988) processar a hidrologia para qualquer DEM de entrada, o algoritmo extrai incorretamente as redes em regiões de superfícies planas. GARBRECHT; MARTZ (1997) apresentam um novo método para o cálculo de direções de drenagem das redes fluviais em superfícies planas. O algoritmo utiliza dois gradientes, um afastando os pontos de regiões mais altas e outro direcionando-os para regiões mais baixas. Essa técnica foi estendida por BARNES; LEHMAN; MULLA (2014), que propuseram otimizações para o algoritmo, como a utilização de pesos para os gradientes, de modo a evitar a ambiguidade na etapa de definição do fluxo.

3.3 REDES NEURAIS GENERATIVAS

Em sua versão original, GANs apresentaram resultados com qualidade igual ou superior quando comparados com os melhores modelos generativos existentes. Atualmente, por serem consideradas o estado da arte em modelos generativos, as GANs vem sendo estendidas em diversos sentidos (JOHNSON; ALAHI; FEI-FEI, 2016) (ISOLA et al., 2017) (RADFORD; METZ; CHINTALA, 2015) (GUÉRIN et al., 2017). Entretanto, um dos principais problemas presentes nas GANs está relacionado com sua estabilidade durante a etapa de treinamento.

Uma das arquiteturas mais bem sucedidas para as GAN foi proposta por RADFORD; METZ; CHINTALA (2015), denominada *deep convolutional* GANs (DCGANs). Nesse trabalho foi definido um conjunto de restrições na topologia da rede tornando-a mais estável na etapa de treinamento. Essas restrições passam pela substituição das camadas de pooling, e vão até à definição das funções de ativação. A figura 15 ilustra a generalização da arquitetura para a rede geradora.

Figura 15 – Arquitetura da rede geradora utilizada em DCGAN.

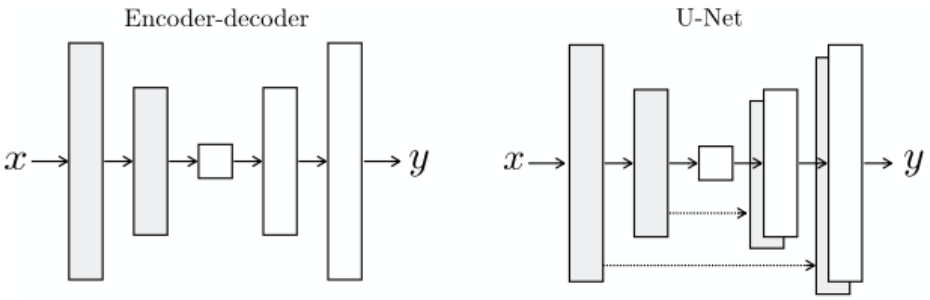


Fonte: (RADFORD; METZ; CHINTALA, 2015).

Seguindo os fundamentos da DCGAN, ISOLA et al. (2017) utilizaram uma GAN condicional (cGAN) como solução de propósito geral para o mapeamento de diferentes representações de um mesmo objeto. Algumas das situações exploradas pelos autores foram a tradução de imagens aéreas para mapas, a colorização de imagens preto e branco, e a transformação de imagens diurnas em noturnas. A arquitetura adotada para a rede geradora foi baseada na U-Net (RONNEBERGER; FISCHER; BROX, 2015), composta por um encoder-decoder que apresenta conexões entre as camadas anteriores e posteriores (Figura 16). Para a rede discriminadora, os autores propuseram uma arquitetura denominada PatchGAN, que, dada uma imagem de entrada, tem como objetivo realizar a discriminação de pedaços da mesma, ao invés dela como

um todo.

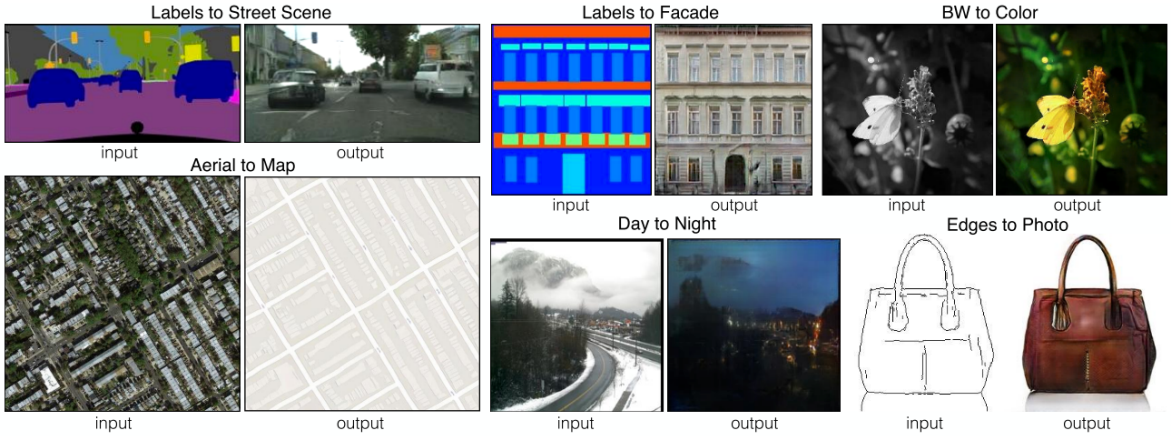
Figura 16 – Arquiteturas de redes generativas.



Fonte: (ISOLA et al., 2017).

A entrada da rede geradora é composta de uma imagem em um determinado contexto, e o seu treinamento é feito a partir da rede discriminadora. A rede discriminadora, por sua vez, recebe como entrada um par de imagens, representado a imagem em seu domínio original e sua transformação. Os autores optaram por não utilizar como entrada adicional a camada de *noise* z, uma vez que perceberam, através de experimentos, que a rede geradora simplesmente aprendia a ignora-la. Para evitar a geração de resultados determinísticos, foram aplicadas camadas de *dropout*, que realizam a desativação de maneira randômica de uma porcentagem das ligações entre duas camadas. A Figura 17 mostra os resultados obtidos na tradução de imagens em diferentes domínios, demonstrando o enorme potencial das cGANs em tarefas de tradução de imagens.

Figura 17 – Resultados para tradução de imagens em diversos domínios utilizando cGAN.



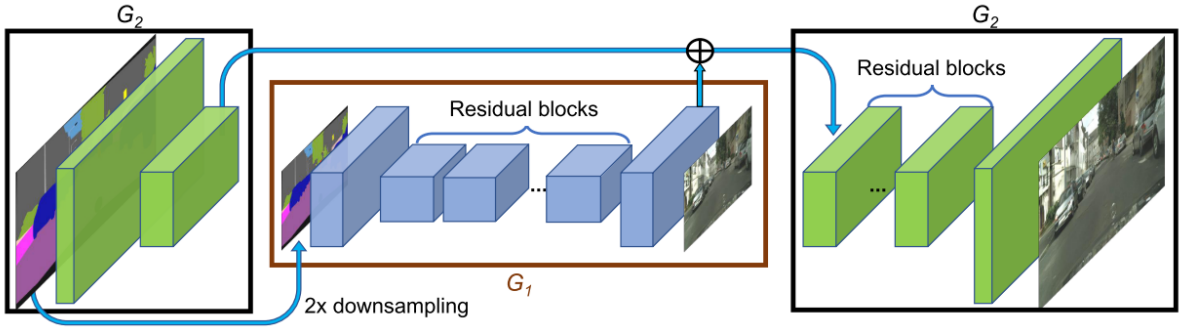
Fonte: (ISOLA et al., 2017).

Apesar de apresentarem resultados surpreendentes, a convergência da rede está limitada para imagens de baixa resolução. Recentemente, CHEN; KOLTUN (2017) avaliaram a

dificuldade na geração de imagens de alta resolução devido a grande instabilidade das redes e propensão a não convergência. WANG et al. (2018) apresentam um novo método para sintetização de imagens foto-realísticas de alta resolução. Esse método é composto de uma nova função de custo, juntamente com redes geradora e discriminadora que trabalham em multirresolução.

A rede geradora é composta por duas sub redes: $G1$ e $G2$. Denominada rede geradora global, $G1$ opera com a imagem na metade da resolução original. $G2$, chamada de rede realçadora local, é responsável por gerar as imagens na resolução final. A arquitetura de ambas as redes é baseada na implementação de JOHNSON; ALAHI; FEI-FEI (2016), consistindo de blocos de convoluções, seguidos por blocos residuais (HE et al., 2016) e convoluções transpostas (Figura 18). Entretanto, diferentemente da rede $G1$, a entrada para os blocos residuais de $G2$ é composta pela soma de duas feature-maps: a feature-map proveniente da saída das camadas convolucionais iniciais de $G2$ e a saída das camadas de convolução transposta de $G1$.

Figura 18 – Arquitetura da rede geradora *coarse-to-fine*.



Fonte: (WANG et al., 2018).

A utilização de imagens de alta resolução torna mais complexo o objetivo da rede discriminadora. Desta forma, WANG et al. (2018) estruturaram uma rede discriminadora capaz de manipular esses dados. A rede discriminadora D é composta de 3 sub redes equivalentes. Entretanto, cada rede trabalha com a mesma entrada em diferentes resoluções. Podemos descrever o novo objetivo da rede D da seguinte maneira:

$$\min_{G} \max_{D1,D2,D3} \sum_{k=1,2,3} \mathcal{L}_{GAN}(G, D_k). \tag{3.1}$$

Com a finalidade de estabilizar o treinamento, WANG et al. (2018) propuseram uma função de custo baseada na similaridade das características entre as imagens de entrada e sintetizada. Para a extração dessas características são utilizadas as camadas intermediárias da rede discriminadora D. Logo, o treinamento é guiado de forma que as camadas intermediárias de

uma imagem sintetizada sejam similares as camadas correspondentes de uma imagem real. Essa função de custo $\mathcal{L}_{FM}$ é definida como:

$$\mathcal{L}_{FM}(G, D_k) = \mathbb{E}_{(s,x)} \sum_{i=1}^{T} \frac{1}{N_i} [||D_k^{(i)}(s, x) - D_k^{(i)}(s, G(s))||_1] \tag{3.2}$$

onde T representa o número total de camadas e N_i denota o número de elementos em cada camada. Podemos estabelecer uma relação entre $\mathcal{L}_{FM}$ e a chamada função de custo perceptiva $\mathcal{L}_{perc}$ (JOHNSON; ALAHI; FEI-FEI, 2016), já que ambas seguem a mesma lógica, diferindo no fato de que a segunda é utilizada sobre uma rede pré treinada. WANG et al. (2018) demonstram em experimentos que a utilização de $\mathcal{L}_{perc}$ em conjunto com $\mathcal{L}_{FM}$ proporciona uma melhora de performance. Ao final, o objetivo que engloba a função de custo $\mathcal{L}_{GAN}$ e $\mathcal{L}_{FM}$ para essa nova arquitetura é:

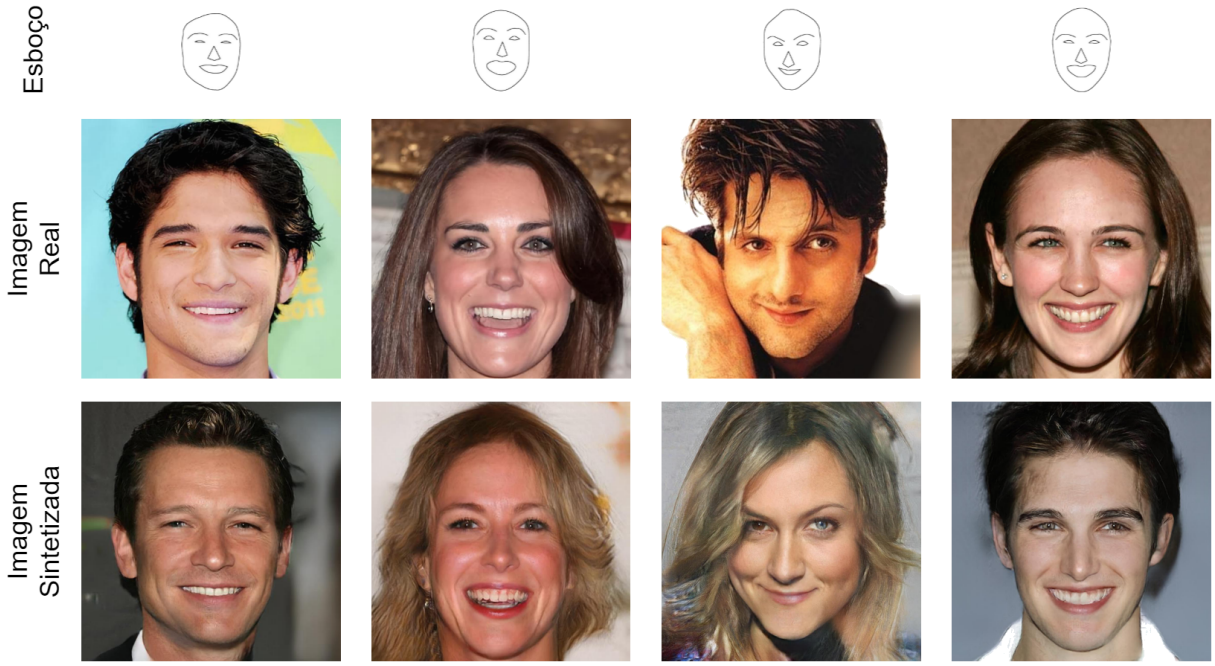
$$\min_G (\max_{D1,D2,D3} \sum_{k=1,2,3} \mathcal{L}_{GAN}(G, D_k) + \lambda \sum_{k=1,2,3} \mathcal{L}_{FM}(G, D_k)). \tag{3.3}$$

O treinamento da rede com o banco de dados de CelebA-HQ (KARRAS et al., 2017) possibilitou a sintetização de faces em alta resolução a partir de esboços (Figura 19). Contudo, a complexidade da arquitetura proposta pelos autores exige uma grande quantidade de memória de video, impossibilitando a execução do treinamento da rede na grande maioria dos computadores atuais. Para obter os resultados apresentados, foi utilizado uma NVIDIA QUADRO M6000 com 24GB de memória, ou seja, um hardware que para muitos ainda é inacessível.

Sendo uma das inspirações para o presente trabalho, GUÉRIN et al. (2017) propuseram a criação de terrenos de forma interativa utilizando uma cGAN como um sintetizador. A entrada da rede é formada por esboços correspondentes as estruturas de rios e montanhas. Porém, a arquitetura utilizada para criar o sintetizador foi baseada em (ISOLA et al., 2017), e assim, a mesma apresenta limitações quanto a resolução das imagens.

Assim como em (ISOLA et al., 2017), GUÉRIN et al. (2017) optaram por não utilizar na entrada da rede geradora o noise z. Entretanto, z somente torna-se descartável quando a entrada condicional contém dados em sua completude, como é o caso de (ISOLA et al., 2017). Ao utilizar esboços como entrada, (GUÉRIN et al., 2017) acabam fornecendo informações esparsas, levando a falha da rede na geração de terrenos em diversos casos (Figura 20). Além disso, os resultados apresentados por GUÉRIN et al. (2017) não correspondem a resultados idealmente esperados (Figura 20).

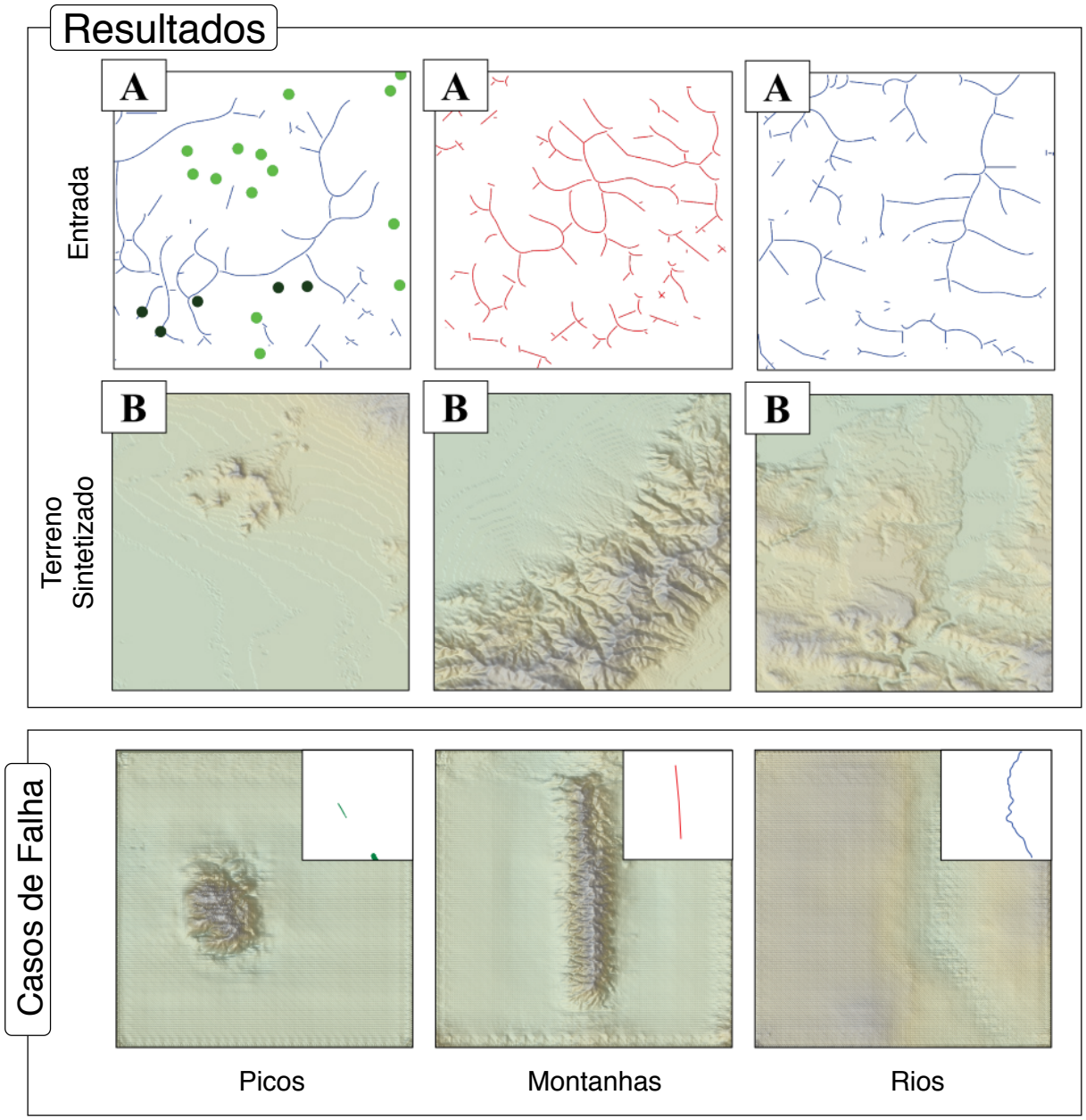
Figura 19 – Resultados para geração de rostos a partir de esboços: esboço seguido da imagem original e imagem sintetizada.



Fonte: (WANG et al., 2018).

A sintetização de terrenos é uma área que vem sendo estudada há muitos anos, porém, a utilização de GANs para esse propósito foi uma ideia inovadora por parte de GUÉRIN et al. (2017). GANs tem a capacidade de gerar resultados com qualidade muito similar aos dados de entrada. Sendo assim, explorou-se melhor a capacidade das GANs na sintetização de terrenos por meio de esboços. Devido aos resultados obtidos por WANG et al. (2018) na geração de rostos em alta resolução, fez-se adaptações em sua arquitetura para manipulação de DEMs e suas respectivas topografias, trazendo-a para o contexto da geração de terrenos.

Figura 20 – Terrenos gerados por meio de esboços. Na primeira linha são apresentados os resultados para diferentes entradas. A segunda linha mostra casos em que o sintetizador falha em produzir terrenos realísticos.



Fonte: (GUÉRIN et al., 2017).

4 METODOLOGIA

A proposta deste trabalho é permitir a geração de terrenos realísticos de alta resolução a partir de esboços da topografia de um terreno por meio de um sintetizador. A tarefa do sintetizador pode ser interpretada como aprender alguma maneira de mapear uma entrada *X* para uma saída *Y*, a partir de um banco de dados de entrada. Nosso sintetizador é representado por uma rede generativa adversária condicional treinada para este objetivo.

Primeiramente, são coletados os mapas de altura de uma região do globo terrestre. Então, esses mapas de altura passam por uma etapa de pré processamento para extração de suas topografias. Por sua vez, esse conjunto de dados é utilizado na etapa de treinamento do sintetizador (Figura 21). Desta forma, os esboços topográficos são utilizados como entrada da rede e os mapas de altura correspondem as saídas desejadas.

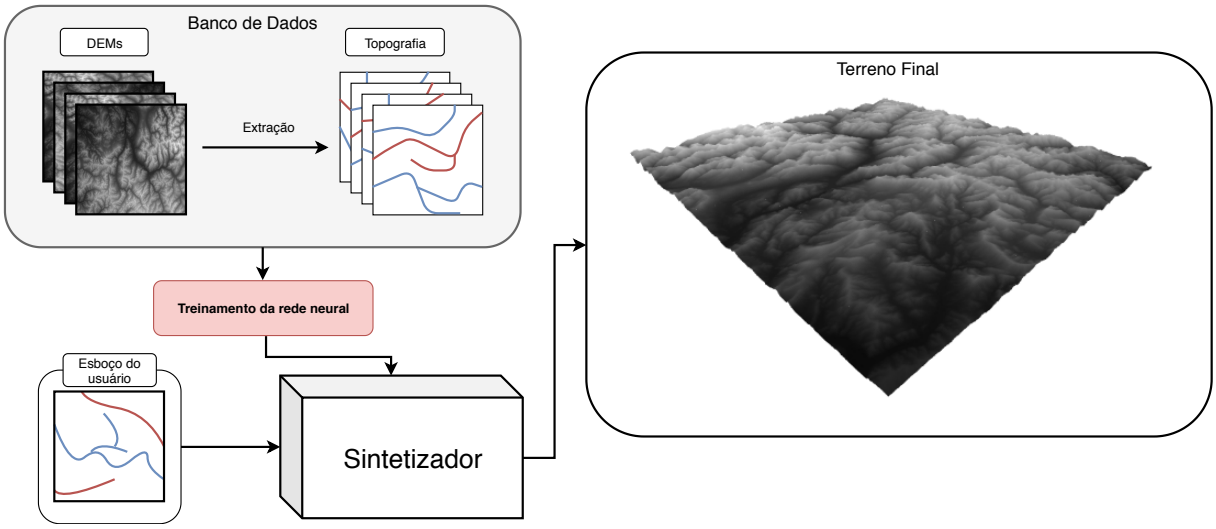
No decorrer deste projeto, foram experimentadas uma grande quantidade de arquiteturas. Como resultado desses experimentos, foram identificadas 3 características importantes na sintetização de terrenos realísticos em alta resolução:

- Estruturas Topográficas Consistentes. Estruturas topográficas são essenciais para a coordenação da geração dos terrenos. Entretanto, as estruturas utilizadas devem ter fortes correlações com o terreno original. Ou seja, ao utilizar redes fluviais e montanhas sobre um banco de dados da região dos alpes suíços, a rede é capaz de entender que regiões de rios tendem a ter uma menor elevação, enquanto regiões próximas aos esboços de montanhas têm maiores elevações.

- *Noise* Adicional. A não utilização de z, como proposto por GUÉRIN et al. (2017), faz com que sejam geradas alturas somente para as regiões próximas aos esboços topográficos. Contudo, a entrada adicional z permite que a rede seja capaz de sintetizar terrenos completos mesmo não havendo nenhum esboço topográfico.

- Memória. Uma alta complexidade para o modelo é essencial para a assimilação das informações contidas nos dados de treinamento. Ao mesmo tempo, essa complexidade necessita uma grande quantidade de memória. De forma que nosso modelo possa ser executado na ampla maioria dos hardwares atuais, foi proposto um design modular capaz de sintetizar terrenos similares aos originais. Todavia, é consistente o aumento da qualidade dos terrenos sintetizados com o aumento da capacidade do modelo.

Figura 21 – O pipeline de execução da aplicação proposta consiste em três etapas: pré processamento; treinamento e sintetização. No pré processamento são extraídas as informações topográficas dos mapas de elevação coletados. Durante a etapa de treinamento, a rede aprende a correlação entre os terrenos e a topografia. Como resultado, na etapa de sintetização, terrenos podem ser esculpidos de maneira interativa.



A etapa de pré processamento dos DEMs coletados é composta por um algoritmo de cálculo de direção de fluxo, que resulta na extração das redes fluviais contidas no mesmo. Esse algoritmo também é utilizado para a extração da cadeia de montanhas, para isso, o arquivo tem suas elevações invertidas. Essa etapa é descrita em detalhes no Capítulo 5. A arquitetura utilizada para a rede sintetizadora, assim como a função de custo para a otimização e a abordagem adotada para o treinamento, são tratados no Capítulo 6.

5 GERAÇÃO DO BANCO DE DADOS

O algoritmo proposto por JENSON; DOMINGUE (1988) e as otimizações propostas por GARBRECHT; MARTZ (1997) permitem a extração das redes fluviais contidas em DEMs, ao mesmo tempo que corrigem as falhas presentes nesses arquivos. Para tornar o algoritmo mais eficiente, neste trabalho foram realizadas adaptações das estruturas de dados utilizadas para execução do mesmo por meio de *compute shaders*. Ao invés das listas utilizadas nos algoritmos originais, optou-se por texturas auxiliares para armazenar as informações necessárias em cada etapa do processo. O processo de extração das redes fluviais é composto pelas etapas de: 1) Cálculo das direções de fluxo; 2) Processamento do fluxo sobre planícies; 3) Construção das bacias hidrográficas; 4) Cálculo da taxa de acumulo de fluxo (Figura 22).

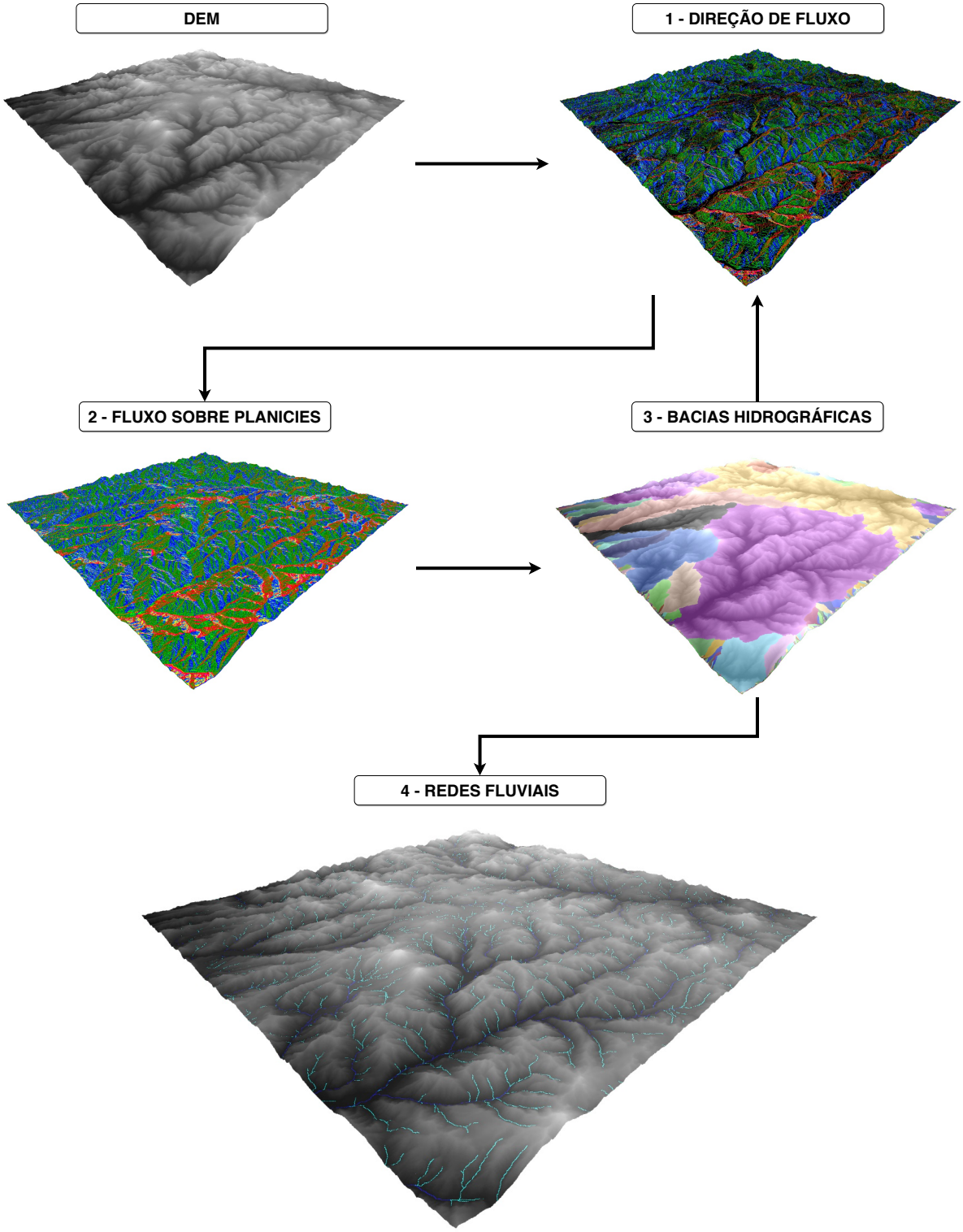
Em geral, DEMs contém depressões que podem ocultar os fluxos de drenagem. Essas regiões são caracterizadas por serem cercadas de pontos com elevações superiores, impossibilitando a definição do sentido de drenagem por meio das direções de fluxo. Para resolver esse problema, executa-se de maneira iterativa as etapas 1, 2 e 3, elevando a altura das células do terreno até o momento em que todas as bacias hidrográficas tenham um ponto de escoamento para uma de suas extremidades. Analogamente, esse processo equivale a inundar as depressões até que as mesmas transbordem.

5.1 DIREÇÃO DE FLUXO

O fluxo de uma célula do terreno é determinado pela direção em que a água é drenada para fora da mesma. Esta direção pode ser definida como uma orientação para uma das 8 células adjacentes à célula atual. A Figura 22.1 mostra as direções de fluxo de cada célula, onde cada direção é representada por uma cor. Células pretas representam regiões sem fluxo definido, casos como esse correspondem a regiões planas. Existem três condições no momento da avaliação dos 8 vizinhos adjacentes.

- Condição 1. Todos os vizinhos têm uma elevação maior do que a elevação da célula atual. Quando ocorre, o sentido do fluxo é definido como nulo, ou seja, (0,0);
- Condição 2. Todos os vizinhos têm uma elevação maior e pelo menos um tem uma elevação igual à elevação da célula atual. Desta forma, nesta etapa ainda não é possível determinar o sentido do fluxo, e seu valor é definido também como nulo (0,0);

Figura 22 – Processo de extração das redes fluviais de um DEM: Dado um DEM, (1) o fluxo de cada célula corresponde a direção para célula vizinha com menor elevação, (2) caso não exista, a célula está localizada sobre uma região plana, e um segundo algoritmo é executado para extração do fluxo dessas células. (3) Com todo o fluxo calculado, são extraídas as bacias hidrográficas. Caso exista alguma bacia que não escoe para uma extremidade, essa é inundada e o algoritmo é executado novamente a partir do passo (1). Por fim, (4) as redes são extraídas a partir do acúmulo de fluxo de cada célula.



- Condição 3. Existem um ou mais vizinhos com elevação menor do que a elevação da célula atual. Nesse caso, o sentido do fluxo é definido pelo vizinho que tiver o maior valor na relação entre diferença de altura e distância entre as células, denominado δ. Células adjacentes horizontal e verticalmente têm distância 1, já células diagonais têm distância $\sqrt{2}$. Caso exista dois vizinhos com o maior valor, um deles é escolhido arbitrariamente.

5.2 FLUXO SOBRE PLANÍCIES

Para cada célula sem fluxo definido (Condição 2), são avaliadas as direções dos fluxos de células adjacentes. Caso alguma célula adjacente contenha seu fluxo já definido, o fluxo da célula atual será em direção ao vizinho com maior valor δ. Entretanto, as redes fluviais em regiões planas de grandes extensões tendem a seguir uma das extremidades do canal, onde o correto seria percorrer o estreito do mesmo.

Como no algoritmo apresentado por JENSON; DOMINGUE (1988), utiliza-se dois gradientes, um responsável por afastar as células de regiões mais altas, e o outro por direcionar para regiões mais baixas. Para obter ambos os gradientes, são utilizados dois mapas de incrementos, que são calculados em paralelo por meio de dois *compute shaders* e armazenados em uma textura T_G de dois canais inteiros.

O primeiro gradiente é obtido analisando todas as células que não contém uma direção de fluxo e são adjacentes a células com maiores elevações. Iterativamente, incrementa-se o contador de cada célula que tenha uma célula adjacente de mesma altura e tenha seu contador inicializado. O processo termina no momento em que todas as células sem direção de fluxo tenham seu contador inicializado (Figura 23.1). O valor final v_f da célula é determinado por

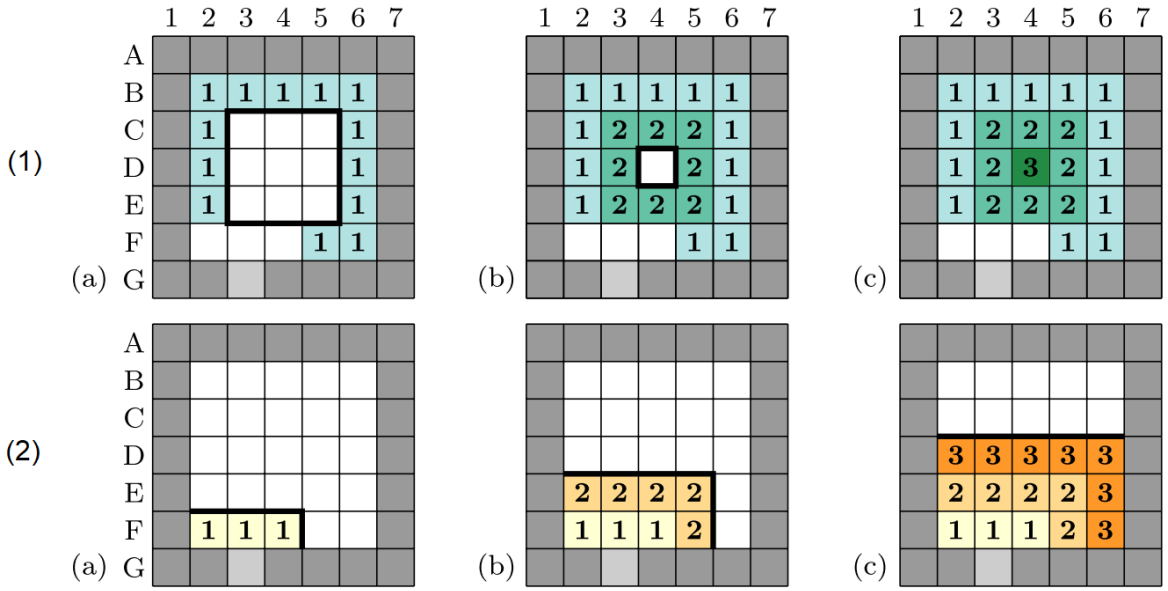
$$v_f = v_{max} - v_c$$

onde v_{max} é o contador com maior valor, e v_c é o contador de uma respectiva célula. v_f é gravado no primeiro canal da textura T_G.

O segundo gradiente é obtido por meio da análise de células que, ao mesmo tempo, tenham: (a) a direção de fluxo definida; e (b) uma célula adjacente de mesma altura que não tenha direção de fluxo. E, assim como no gradiente anterior, um contador para as células sem fluxo é incrementado iterativamente, até que todas as células tenham seu contador inicializado (Figura 23.2). O contador de cada célula é gravado no segundo canal da textura T_G.

Após o cálculo dos gradientes, a direção de fluxo de cada célula é definida como a

Figura 23 – Gradientes para o cálculo de fluxo sobre planícies: Células brancas, cinza escuro e cinza claro representam, respectivamente, regiões planas, mais altas e mais baixas. As colunas a, b e c, representam as 3 primeiras iterações dos algoritmos de cálculo dos gradientes. (1) e (2) correspondem ao gradiente de afastamento de regiões mais altas e gradiente de direcionamento para regiões mais baixas. Adaptada de (BARNES; LEHMAN; MULLA, 2014).

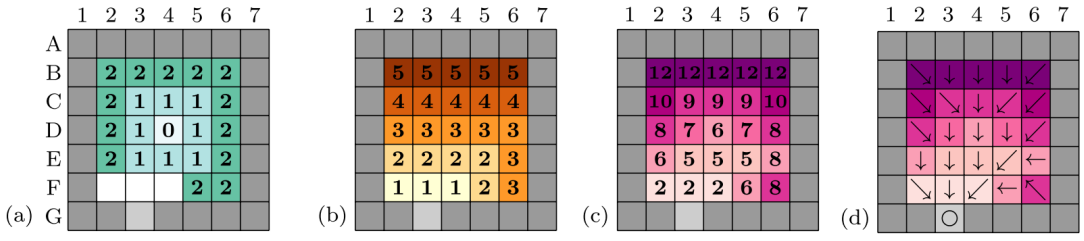


direção para o vizinho que tenha o menor valor G_i (Figura 24). G_i é definido por:

$$G_i = Ga_i + 2 * Gb_i \tag{5.1}$$

onde G_i é o valor resultante, Ga_i é o valor do contador no gradiente para longe de áreas mais altas e Gb_i é o valor do contador em direção para áreas mais baixas.

Figura 24 – Direção de fluxo obtida a partir dos gradientes: (a) e (b) Resultado dos gradientes Ga e Gb. (c) Valor resultante de G. (d) Direções de fluxo de cada célula. Adaptada de (BARNES; LEHMAN; MULLA, 2014).



5.3 BACIAS HIDROGRÁFICAS

Uma bacia hidrográfica é composta por todas as células que têm seu fluxo convergido para um mesmo local (Figura 22.4). Primeiramente, percorre-se a direção de fluxo de cada célula, identificando seu ponto de escoamento final. Todos os pontos que convirjam para uma mesma planície pertencerão a mesma bacia. Cada bacia hidrográfica é representada por um valor inteiro único.

Após a definição de todas as bacias hidrográficas, caso haja alguma que não escoe para alguma extremidade do terreno, neutralizando o fluxo de todas suas células, é realizado o processo de inundação. Para isso, primeiramente são encontradas as células denominadas células de mínimo local. Para cada célula da extremidade de uma bacia é selecionada a célula adjacente com a menor altura. O passo seguinte é calcular a célula de mínimo global, ou seja, dentre todas as células selecionadas como células de mínimo local, é selecionada a célula de menor altura. Então, todas as células pertencentes a essa bacia, menores que esse valor, são elevadas até o mesmo.

5.4 ACÚMULO DE FLUXO

A etapa final é o cálculo do mapa de acúmulo de fluxo. O valor de acúmulo de uma célula corresponde a quantidade de células que durante o percurso de drenagem, passam sobre a mesma. Para cada célula do DEM são inicializados iteradores que seguem o fluxo da célula atual e incrementam seu contador até que o mesmo chegue a uma extremidade do DEM.

As redes fluviais são extraídas a partir do mapa de acúmulo de fluxo. Determinado um ponto de corte PC, valores superiores são considerados pertencentes à rede. Desta forma, quanto menor o valor de PC, mais densa será a rede. Da mesma maneira, é possível caracterizar diferentes tipos de canais. Por exemplo, canais acima de um valor X pode ser determinado como canal principal, e valores Y onde $X > Y > PC$, são ditos como canais secundários (Figura 22.4).

6 REDE SINTETIZADORA

A GAN que constitui o sintetizador é formada por uma rede geradora G e uma rede discriminadora D. As estruturas de ambas as redes foram projetadas para a manipulação de imagens em alta resolução, sendo similares as utilizadas por WANG et al. (2018). Além disso, foram utilizadas outras duas funções de custo complementares à GAN, *feature matching Loss* (WANG et al., 2018) e *perceptual loss* (JOHNSON; ALAHI; FEI-FEI, 2016), tornando o objetivo da rede mais robusto.

6.1 ARQUITETURA

Assim como em WANG et al. (2018), a rede geradora é composta pelas sub redes geradoras $G1$ e $G2$. A arquitetura de ambas as redes é similar, tendo sua estrutura dividida em três camadas:

- Camada de entrada. A primeira delas é composta por 3 blocos subsequentes de: (1) convolução; (2) *instance normalization* (ULYANOV; VEDALDI; LEMPITSKY, 2016); e (3) *relu* (MAAS; HANNUN; NG, 2013) (Figura 25.a). As convoluções utilizam um *kernel* de tamanho 3x3, *stride* 2 e *padding* 1. Os filtros utilizados são duplicados a cada camada subsequente. Camadas de *pooling* não são utilizadas, uma vez que, desta maneira, a convolução por si acaba reduzindo a dimensionalidade da entrada;

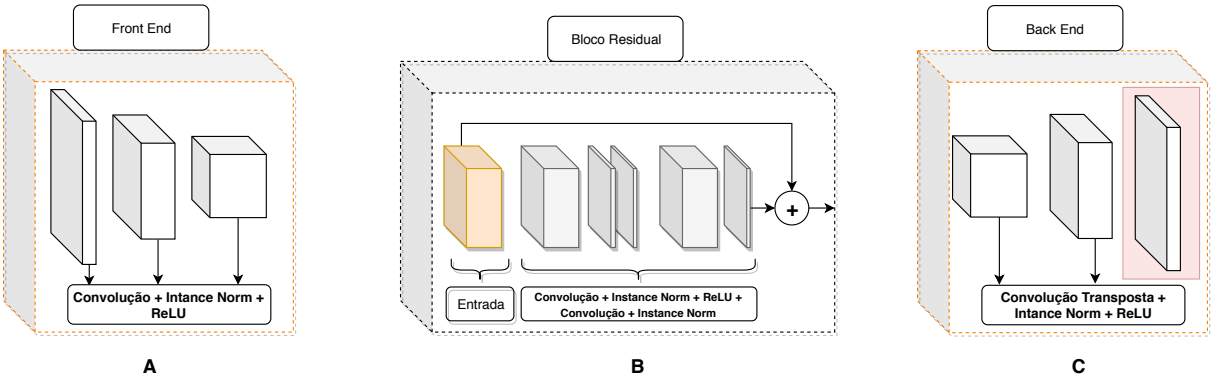
- Camada intermediária. A segunda camada é composta por um conjunto de blocos residuais (HE et al., 2016). A Figura 25.b ilustra a estrutura de um bloco residual;

- Camada de saída. Por último, um conjunto de 3 blocos subsequentes de convolução transposta, *instance normalization* e *relu* formam a camada de saída (Figura 25.c).

Há duas principais diferenças entre as sub redes: a quantidade de blocos residuais utilizados por cada uma e as resoluções de suas entradas.

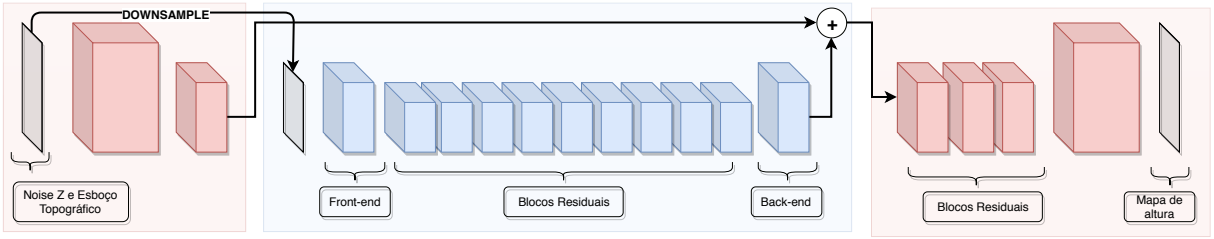
A Figura 26 ilustra a arquitetura da rede geradora. $G1$ é denominada rede geradora global e trabalha com a imagem na metade da resolução final, sendo assim, a sua entrada é composta pela entrada original escalada para metade de suas dimensões. $G2$ é denominada rede refinadora local, tendo sua entrada composta de uma imagem representando o esboço topográfico do terreno na resolução de 1024x1024. Diferentemente da rede $G1$, a entrada para os

Figura 25 – Estrutura que constituem as redes. Ambas as sub redes utilizam a mesma estrutura para suas camadas de entrada (a) e saída (c). Note que no último bloco da camada de saída (destacado em vermelho) não são aplicadas as mesmas operações dos demais. Nesse, é executada uma convolução transposta seguida da função de ativação *Tanh*. Blocos residuais (b) são aplicados entre as camadas de entrada e saída das sub redes, e o número de unidades é diferente para cada uma.



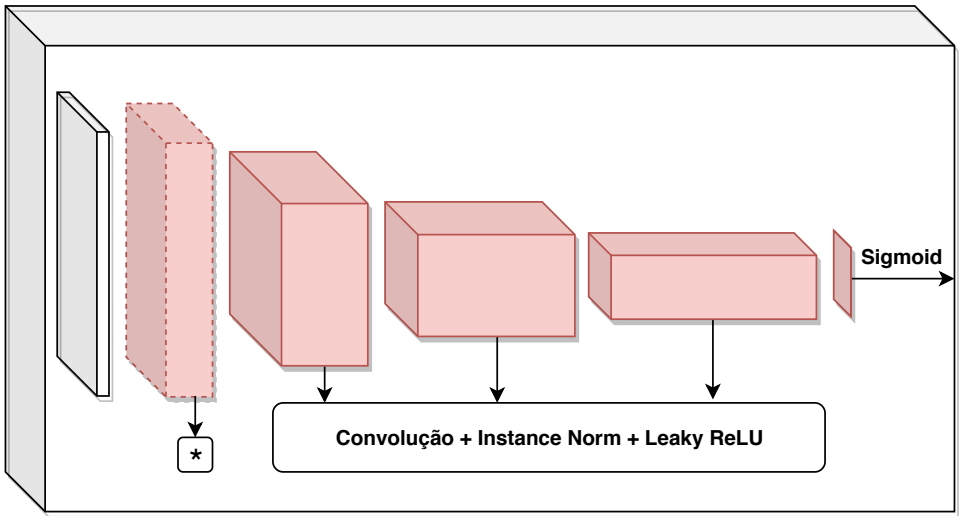
blocos residuais de $G2$ é formado pela concatenação dos *feature maps* vindo da sua camada de entrada com os *feature maps* da camada de saída da rede $G1$.

Figura 26 – Arquitetura proposta da rede geradora. As redes $G1$ e $G2$ são representadas, respectivamente, pelas cores azul e vermelha. A rede $G1$ é treinada com imagens de menor resolução. Sua saída é concatenada a saída da camada de entrada da rede $G2$. A rede $G2$ é treinada com imagens de alta resolução.



Duas redes, com arquiteturas equivalentes, compõe a rede discriminadora, $D1$ e $D2$. Baseada na proposta de ISOLA et al. (2017), a arquitetura é composta de 5 camadas subsequentes de: convolução, *instance normalization* e *leaky relu*. Na última camada é aplicada uma convolução para uma saída unidimensional, seguida de uma função *sigmoid* (Figura 27). Apesar de terem arquiteturas idênticas, as redes trabalham em resoluções diferentes, fazendo com que os *receptive field* operem em diferentes escalas. Ou seja, enquanto a entrada para rede $D1$ tem dimensões de 1024x1024, a entrada da rede $D2$ é de 512x512. Assim, uma das redes guia a rede geradora a produzir imagens consistentes em escala global, enquanto a outra auxilia na produção de detalhes mais específicos.

Figura 27 – Arquitetura de uma rede discriminadora: É formada por blocos subsequentes de convolução, *instance normalization* e *leaky ReLU*. Note que diferente dos demais, no primeiro bloco da rede (pontilhado) somente são aplicadas funções de convolução e leakyReLU.



6.2 FUNÇÃO DE CUSTO

Um dos desafios para o treinamento da rede geradora G é construir uma função de custo que consiga direcionar automaticamente a geração para saídas de alta qualidade. GUÉRIN et al. (2017) utilizou, juntamente com a função de custo $\mathcal{L}_{gan}$, uma função de custo que utiliza como base a distância euclidiana entre a saída da rede geradora $\hat{y}$ e a imagem original y, denominada $\mathcal{L}_{L1}$. Na prática, essa função de custo faz com que a rede geradora seja guiada de forma que os pixeis de $\hat{y}$ correspondam exatamente aos pixeis da imagem original y.

Ao invés disso, foi utilizada uma função de custo conhecida como função de custo perceptual $\mathcal{L}_{perc}$, que trabalha no contexto de características, direcionando a rede geradora a produzir $\hat{y}$ com características semelhantes a y. $\mathcal{L}_{perc}$ é descrita por:

$$\mathcal{L}_{perc}(\hat{y}, y) = \sum_{l \in L_\phi} \sum_{i=1}^{N_l} ||F_i^l(\hat{y})) - F_i^l(y)||_2^2 \tag{6.1}$$

onde F_i^l representa a saída de uma determinada camada convolucional $l \in L_\phi$. L_ϕ corresponde aos índices das camadas selecionadas da rede ϕ. N_l é o número de feature maps na respectiva camada l. A rede utilizada para extrair as camadas foi uma rede VGG19 (SIMONYAN; ZISSERMAN, 2014) pré treinada para classificação sobre o banco de dados ImageNet (DENG et al., 2009).

Outro problema característico das GANs é sua dificuldade de estabilização na etapa de treinamento, resultando no colapso ou na não convergência da rede. Uma função de custo deno-

minada *feature matching loss* $\mathcal{L}_{FM}$ (WANG et al., 2018) foi utilizada de maneira a estabilizar o treinamento a medida em que a rede geradora aprende a produzir características naturais em diferentes escalas. $\mathcal{L}_{FM}$ e $\mathcal{L}_{perc}$ são similares, porém, $\mathcal{L}_{FM}$ utiliza as informações presentes nas camadas internas das próprias redes discriminadoras para guiar o aprendizado dessas representações intermediárias, ao invés de utilizar uma rede pré treinada. O objetivo final resulta da combinação dos três custos, e é descrito por:

$$\min_{G}(\max_{D1,D2} \sum_{k=1,2} \mathcal{L}_{GAN}(G, D_k) + \lambda_1 \sum_{k=1,2} \mathcal{L}_{FM}(G, D_k) + \lambda_2 \mathcal{L}_{perc}(G, y)). \tag{6.2}$$

onde k representa as redes discriminadoras e λ é um parâmetro que controla a influência de cada função de custo.

6.3 TREINAMENTO

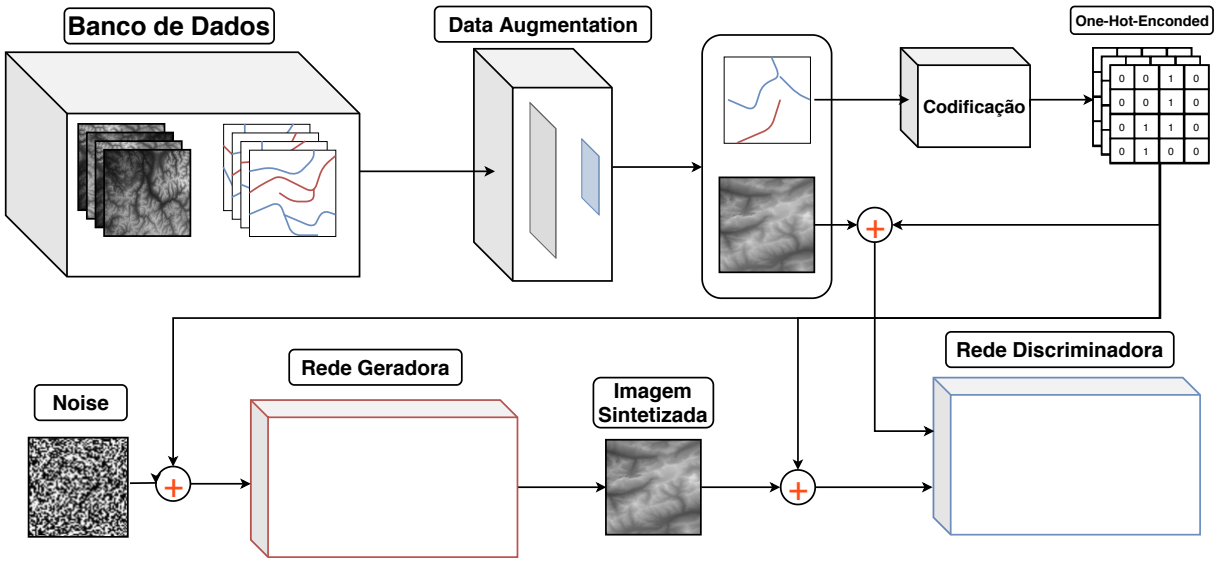
Na etapa de treinamento é realizada a otimização da rede seguindo a abordagem padrão adotada em (GOODFELLOW et al., 2014): alterna-se uma iteração do gradiente descendente em D e uma iteração em G. Foi utilizado gradiente descendente estocástico aplicando o Adam solver (KINGMA; BA, 2014), com a taxa de aprendizado 0.0002 e *momentum* β de 0.999, e o tamanho do *batch* de 1. O valor de λ aplicado para ambas funções de perda foi de 10.

A entrada da rede geradora é composta por uma imagem referente a topografia e um vetor z. Ao ser lida, a imagem topográfica é convertida para uma estrutura *one-hot-enconde*. Considerando o layout semântico $L \in 0, 1^{m \times n \times c}$, onde $m \times n$ representa a resolução em pixels da imagem e c corresponde ao número de classes semânticas, que por sua vez equivale ao número de elementos topográficos. A entrada da rede discriminadora é formada pelos pares de mapas de altura e suas respectivas topografias, obtidos tanto do banco de dados, quanto da saída da rede geradora (Figura 28).

O banco de dados é composto por DEMs das regiões dos Alpes Suíços extraídos da *USGS Earth Explorer*. Foi utilizado um total de 10 *patches* cobrindo uma região de um grau quadrado na precisão de um arco minuto. Cada *patch* consiste em um grid de resolução 3,600 x 3,600, cada célula do grid representa aproximadamente uma área de 30 metros quadrados.

Ao utilizar uma baixa quantidade de amostras durante o treinamento, limita-se a assimilação da rede a um número escasso de características. Mesmo produzindo resultados muito bons para o conjunto de dados de treinamento, a mesma torna-se ineficaz para prever novas

Figura 28 – Pipeline de execução para o treinamento da rede sintetizadora.



entradas. Essa falha é conhecida como *overfitting*. Evita-se que isso ocorra aplicando técnicas de *data augmentation*. Uma vez que cada *patch* tem 3600x3600 pixels, são aplicadas operações de rotação horizontal e vertical, e recortes de uma área de 1024x1024 pixels em uma posição aleatória da imagem. Desta forma, são aumentadas efetivamente a quantidade de amostras.

Em geral, técnicas que manipulam imagens aplicam uma grande quantidade de filtros entre as camadas das redes, como visto em (WANG et al., 2018), (ISOLA et al., 2017) e (CHEN; KOLTUN, 2017). Esses filtros são responsáveis pela assimilação das características presentes nas imagens. Desta forma, quanto mais complexa uma imagem, maior a quantidade de filtros necessários para processar a mesma. Ao mesmo tempo, maior é o consumo de memória para o treinamento da rede. Apesar de estarmos lidando com imagens, a representação dos mapas de altura é feita com apenas 1 canal, diferente de imagens coloridas, que utilizam 3 canais. Consequentemente, a complexidade dos dados de entrada da nossa aplicação é muito menor, possibilitando a redução do número de filtros e, assim, diminuindo o uso da memória e permitindo o treinamento da rede em alta resolução, mesmo em computadores com hardwares muito inferiores aos utilizados por WANG et al. (2018). Nossa arquitetura utiliza a quantidade inicial de 16 filtros, ao invés de 64 como apresentado no trabalho original.

7 RESULTADOS

Primeiramente, é provida uma análise paramétrica da solução, comparando o *trade-off* entre a capacidade do modelo e a qualidade dos resultados. Posteriormente, é investigado, em contraposição do trabalho de GUÉRIN et al. (2017), o uso de z como entrada adicional. Por fim, exploramos a influência dos esboços topográficos utilizados durante a etapa de inferência do sintetizador.

Tanto o algoritmo para extração topográfica dos DEMs, quanto a aplicação para edição interativa e visualização 3D do terreno, foram implementados na Unity. Já a parte da rede neural foi implementada utilizando o PyTorch. O treinamento da rede foi realizado em uma placa de video NVidia GTX 1070 com 8GB de memória.

7.1 ANALISE PARAMÉTRICA

Uma das limitações das redes neurais na geração de imagens de alta resolução é a quantidade de memória necessária para o treinamento. Desta forma, não é possível manipular imagens de 1024x1024 pixels utilizando inicialmente 64 *feature maps*, já que o mesmo requer mais de 16GB de memória. De maneira a analisar a influência do número de *feature maps*, treinou-se duas redes sintetizadoras, que usam a quantidade de 16 e 64 na camada inicial, para geração de terreno de 256x256 pixels.

A Tabela 1 mostra a quantidade de memória necessária para o treinamento de ambas as redes. A arquitetura da rede geradora para as aplicações que trabalham com a resolução de 256x256 é formada somente pela rede geradora global. O custo de memória para a rede que utiliza 64 *feature maps* iniciais é, aproximadamente, 5 vezes maior que a mesma rede com 16 *feature maps* iniciais. Desta forma, a quantidade de 16 *feature maps* viabiliza o treinamento de uma rede com entradas de resolução 1024x1024 pixels em boa parte dos hardwares encontrados no mercado. O tempo de inferência para todas as redes é baixo, o que torna seu uso propício em aplicações de edição interativa.

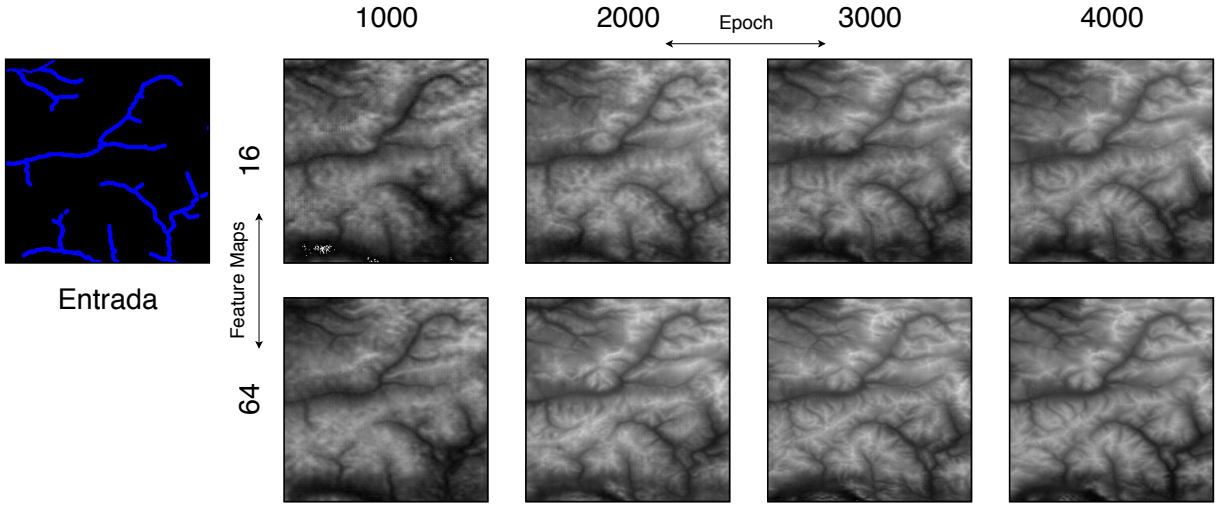
A Figura 29 mostra os resultados obtidos para uma mesma entrada em diferentes *epochs*. Constata-se que, mesmo com um número muito menor de *feature maps*, é possível reproduzir as características naturais dos dados de entrada. Ou seja, as informações topográficas são quase plenamente capturadas por um número pequeno e distinto de *feature maps*. Portanto, o aumento

Tabela 1 – Comparação para o tempo de inferência, tempo de treinamento e consumo de memória entre três diferentes redes.

Resolução (pixels)	Feature Maps (un)	Arquitetura de G	Inferência (s)	Iteração do Treinamento (s)	Memória (GB)
256x256	16	Global	0.014	0.21	1.0
256x256	64	Global	0.015	0.65	5.0
1024x1024	16	Completa	0.041	1.15	5.6

deste número pode ocasionar a redundância, onde diferentes *feature maps* acabam mapeando as mesmas características, o que implica na estabilização da qualidade dos resultados.

Figura 29 – Resultados de redes com diferentes números de *feature maps* e *epochs*.



A convergência da rede ao longo do treinamento é ilustrada na Figura 30. A maior evolução da rede é vista nos 1000 primeiros *epochs*, onde são assimiladas as características globais, como depressões e picos nas regiões de rios e montanhas. Detalhes mais específicos, como superfícies erodidas, começam a aparecer a partir do 2000 *epoch*. Os resultados alcançam a qualidade visual dos dados de entrada no *epoch* 3000.

Os DEMs utilizados para o treinamento contém inúmeras falhas de elevação (Figura 31), produzidas durante o escaneamento da superfície na criação do arquivo. Entretanto, por estarem contidos em uma pequena porção dos dados, a rede, inicialmente, é capaz de abstrair essas falhas. Nota-se na Figura 31.c que a rede reproduz um terreno com boa qualidade, sem a existência desses artefatos. Sendo assim, este trabalho também pode ser aplicado para correção de DEMs.

Figura 30 – Resultados ao longo do treinamento da rede. Os números abaixo de cada terreno representam a quantidade de *epochs* executada.

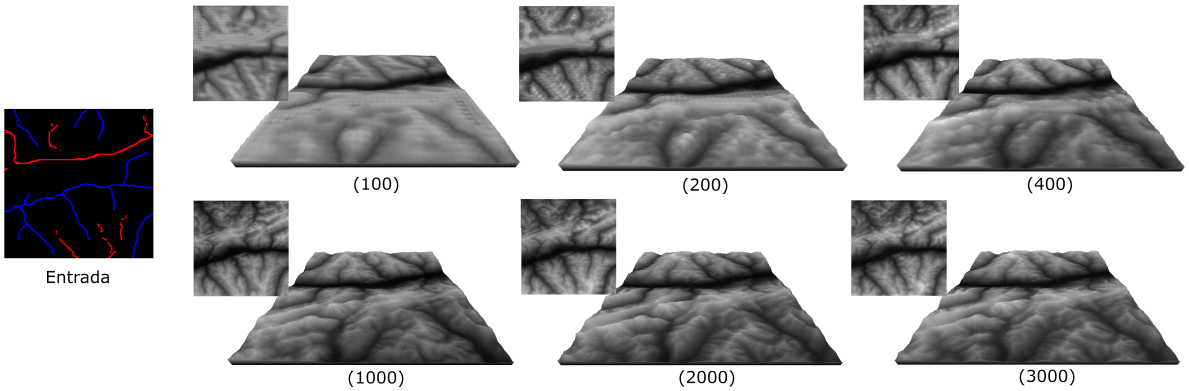
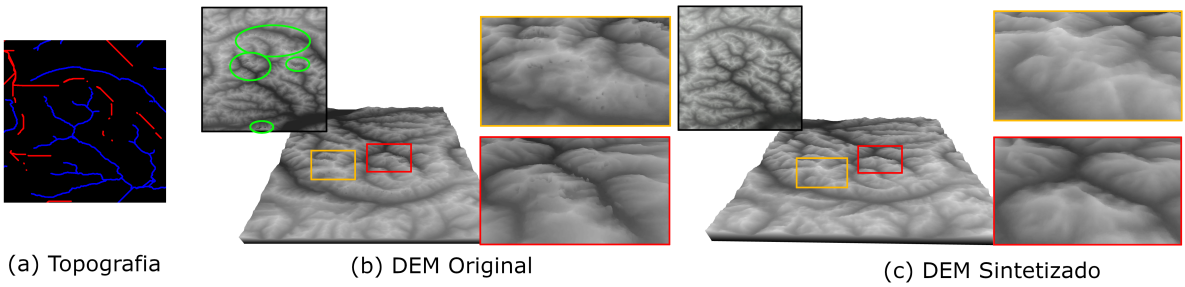


Figura 31 – Correção de falhas em DEMs: (a) representa a topografia extraída do DEM apresentado em (b). Note que são encontradas diversas falhas de elevação, provocando a inconsistência da renderização do terreno. A mesma topografia foi utilizada como entrada da rede sintetizadora já treinada e o terreno sintetizado é visto em (c). Percebe-se que a mesma é capaz de reproduzir o DEM original de forma muito similar, corrigindo suas falhas.

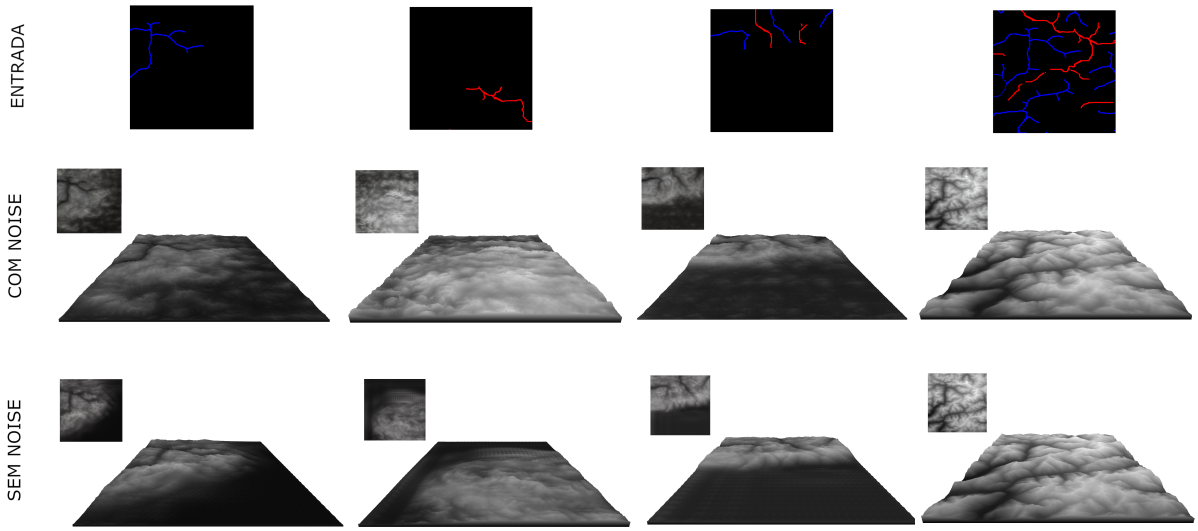


7.2 NOISE COMO ENTRADA ADICIONAL

Uma das limitações apresentadas por GUÉRIN et al. (2017) é a falha na produção de resultados realísticos para esboços esparsos. Como discutido anteriormente, isso deve-se a falta de informações nos dados de entrada. A utilização de um *noise* z como entrada adicional soluciona essa falha. Treinou-se duas redes equivalentes, a primeira somente com os esboços topográficos e a segunda com o *noise* adicional.

A Figura 32 mostra os resultados para diferentes esboços de entrada. A primeira rede é capaz de gerar as elevações somente em regiões próximas aos esboços. Já a rede que utiliza a entrada adicional z produz elevações para o terreno em sua totalidade. Regiões distantes dos esboços são preenchidas com elevações que tendem a reproduzir a natureza das características dos dados de treinamento.

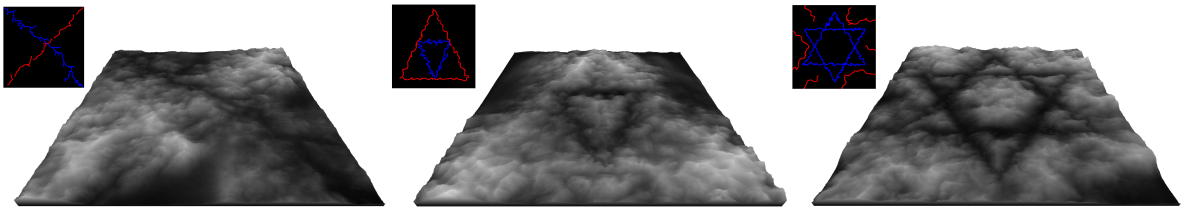
Figura 32 – Influência do *noise* sobre os resultados. A utilização de z torna a rede capaz de sintetizar detalhes mesmo com esboços escaços. Para uma mesma entrada, a rede que não utiliza z falha, produzindo padrões em regiões sem esboço.



7.3 INFLUÊNCIA DOS ESBOÇOS TOPOGRÁFICOS

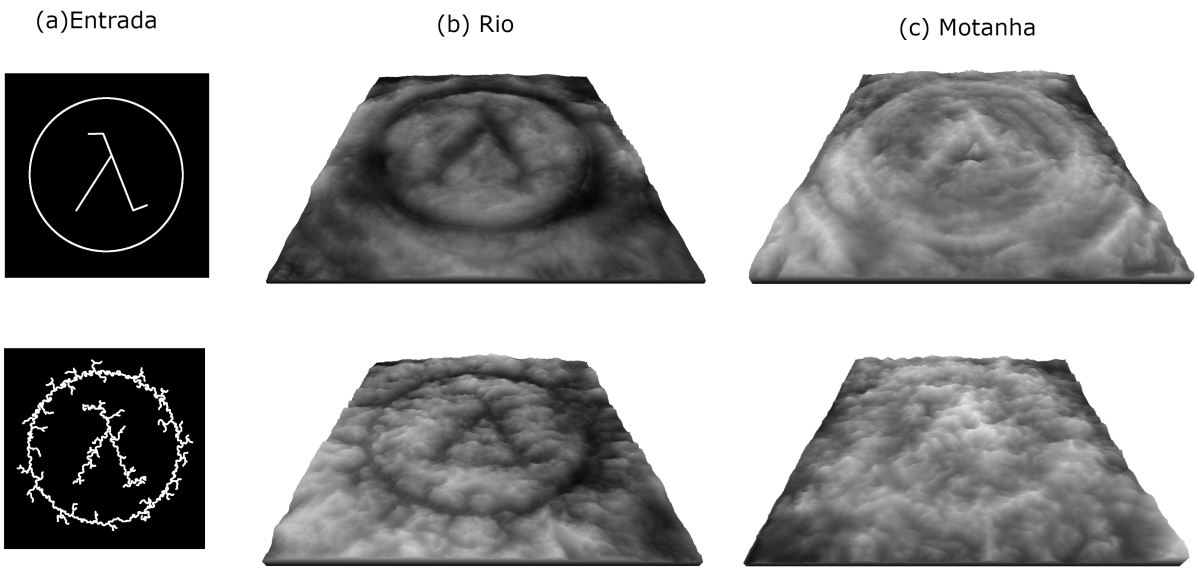
Os resultados demonstram que a produção de terrenos plausíveis está associada aos detalhes, em escala local, presentes nos esboços. Esboços de entrada que se assemelham localmente as topografias de treinamento estabelecem formas naturais aos terrenos, mesmo que em escala global não sejam correlatas. A Figura 33 ilustra topografias que não são encontradas no mundo real - como montanhas que passam sobre rios - mas que originaram terrenos verossímeis.

Figura 33 – Terrenos sintetizados para topografias fictícias.



Tanto as redes fluviais, quanto as cadeias de montanhas utilizadas no treinamento da rede sintetizadora, não passaram sob nenhuma etapa de pré processamento, mantendo todas as deformações de seus percursos. Desta forma, durante a inferência, ao utilizar entradas com esboços simples (geometrias retas) são gerados resultados não realísticos (Figura 34). Uma possível correção seria a suavização da topografia antes da etapa de treinamento.

Figura 34 – Terrenos sintetizados para uma entrada com diferente nível de detalhamento. Ao utilizar esboços fora dos padrões utilizados durante o treinamento, a rede falha ao produzir características realistas.



8 CONCLUSÃO

Algumas abordagens existentes inspiraram o presente trabalho de maneira a propor uma nova abordagem para a geração de terrenos utilizando redes generativas adversárias. As técnicas utilizadas nos últimos anos, apesar de apresentarem bons resultados, contêm, em seus respectivos campos, limitações. As redes neurais na última década tiveram um crescimento exponencial, e GUÉRIN et al. (2017) observaram que essas, uma vez que eram utilizadas com diversos propósitos na área de visão computacional e computação gráfica, poderiam abranger, também, a geração de terrenos.

Motivado pelos resultados de GUÉRIN et al. (2017), foi introduzida uma nova arquitetura otimizada, baseada em WANG et al. (2018), para sintetização de terrenos de alta resolução por meio de esboços topográficos. Além disso, como segundo objetivo, foi estruturada uma rede que viabiliza seu treinamento na grande maioria dos dispositivos atuais. Os resultados obtidos indicam a superioridade dessa solução, na sintetização de terrenos realísticos, sobre as demais técnicas desta área. A rede proposta é capaz de reproduzir detalhes específicos dos terrenos presentes nas bases de treinamento, como erosão hidráulica ou intemperismo térmico, sem a necessidade da aplicação de simulações físicas.

As superfícies terrestres sofrem inúmeros efeitos morfológicos. Entretanto, regiões são expostas a determinados efeitos com base em sua localização, resultando em diferentes topografias. Como explicado neste trabalho, para que as camadas condicionais sejam efetivas durante o treinamento de uma rede, essas devem conter informações que estejam relacionadas com as entradas do banco de dados. Sendo assim, ao utilizar a região dos alpes, foram extraídas suas características predominantes, rios e montanhas. Desta forma, este trabalho pode ser generalizado para o controle da geração de terrenos com outros elementos topográficos. Tomando como exemplo a região do *Grand Canyon*, onde estão presentes longas planícies e imensos precipícios, podemos inferir que a utilização das montanhas nesse caso não teria a mesma influência encontrada sobre os alpes.

Por fim, a rede sintetizadora proposta neste trabalho é capaz de lidar com inúmeros elementos topográficos sem que sejam realizadas alterações em sua arquitetura ou implementação. Vemos neste ponto um potencial caminho a ser explorado, uma vez que uma maior quantidade de elementos topográficos possibilitaria um melhor controle sobre o terreno final.

REFERÊNCIAS

ABADI, M. et al. Tensorflow: a system for large-scale machine learning. In: USENIX} SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION ({OSDI} 16), 12. **Anais...** [S.l.: s.n.], 2016. p.265–283.

AMATO, F. et al. **Artificial neural networks in medical diagnosis**. [S.l.]: Elsevier, 2013.

BACKES, G. C.; ENGEL, T. A.; POZZER, C. T. Real-Time Massive Terrain Generation using Procedural Erosion on the GPU. , [S.l.], 2018.

BARNES, R.; LEHMAN, C.; MULLA, D. An efficient assignment of drainage direction over flat surfaces in raster digital elevation models. **Computers & Geosciences**, [S.l.], v.62, p.128–135, 2014.

BASHEER, I. A.; HAJMEER, M. Artificial neural networks: fundamentals, computing, design, and application. **Journal of microbiological methods**, [S.l.], v.43, n.1, p.3–31, 2000.

BENEŠ, B. et al. Hydraulic erosion. **Computer Animation and Virtual Worlds**, [S.l.], v.17, n.2, p.99–108, 2006.

BENEŠ, B.; FORSBACH, R. Visual simulation of hydraulic erosion. , [S.l.], 2002.

CHEN, Q.; KOLTUN, V. Photographic image synthesis with cascaded refinement networks. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION. **Proceedings...** [S.l.: s.n.], 2017. p.1511–1520.

CHIBA, N.; MURAOKA, K.; FUJITA, K. An erosion model based on velocity fields for the visual simulation of mountain scenery. **The Journal of Visualization and Computer Animation**, [S.l.], v.9, n.4, p.185–194, 1998.

DENG, J. et al. Imagenet: a large-scale hierarchical image database. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2009. **Anais...** [S.l.: s.n.], 2009. p.248–255.

ECK, W. van; LAMERS, M. H. Biological content generation: evolving game terrains through living organisms. In: INTERNATIONAL CONFERENCE ON EVOLUTIONARY AND BIOLOGICALLY INSPIRED MUSIC AND ART. **Anais...** [S.l.: s.n.], 2015. p.224–235.

FOURNIER, A.; FUSSELL, D.; CARPENTER, L. Computer rendering of stochastic models. **Communications of the ACM**, [S.l.], v.25, n.6, p.371–384, 1982.

GAIN, J.; MARAIS, P.; STRASSER, W. Terrain sketching. In: INTERACTIVE 3D GRAPHICS AND GAMES, 2009. **Proceedings...** [S.l.: s.n.], 2009. p.31–38.

GARBRECHT, J.; MARTZ, L. W. The assignment of drainage direction over flat surfaces in raster digital elevation models. **Journal of hydrology**, [S.l.], v.193, n.1-4, p.204–213, 1997.

GÉNEVAUX, J.-D. et al. Terrain modelling from feature primitives. In: COMPUTER GRAPHICS FORUM. **Anais...** [S.l.: s.n.], 2015. v.34, n.6, p.198–210.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org.

GOODFELLOW, I. et al. Generative adversarial nets. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS. **Anais...** [S.l.: s.n.], 2014. p.2672–2680.

GUÉRIN, E. et al. Sparse representation of terrains for procedural modeling. In: COMPUTER GRAPHICS FORUM. **Anais...** [S.l.: s.n.], 2016. v.35, n.2, p.177–187.

GUÉRIN, E. et al. Interactive example-based terrain authoring with conditional generative adversarial networks. **Acm Transactions on Graphics (TOG)**, [S.l.], v.36, n.6, p.228, 2017.

GUO, J. et al. Long text generation via adversarial training with leaked information. In: THIRTY-SECOND AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE. **Anais...** [S.l.: s.n.], 2018.

HARRIS, M. et al. Optimizing parallel reduction in CUDA. **Nvidia developer technology**, [S.l.], v.2, n.4, p.70, 2007.

HE, K. et al. Deep residual learning for image recognition. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. **Proceedings...** [S.l.: s.n.], 2016. p.770–778.

HINTON, G. et al. Deep neural networks for acoustic modeling in speech recognition. **IEEE Signal processing magazine**, [S.l.], v.29, 2012.

HNAIDI, H. et al. Feature based terrain generation using diffusion equation. In: COMPUTER GRAPHICS FORUM. **Anais...** [S.l.: s.n.], 2010. v.29, n.7, p.2179–2186.

ISOLA, P. et al. Image-to-image translation with conditional adversarial networks. **arXiv preprint**, [S.l.], 2017.

JENSON, S. K.; DOMINGUE, J. O. Extracting topographic structure from digital elevation data for geographic information system analysis. **Photogrammetric engineering and remote sensing**, [S.l.], v.54, n.11, p.1593–1600, 1988.

JOHNSON, J.; ALAHI, A.; FEI-FEI, L. Perceptual losses for real-time style transfer and super-resolution. In: EUROPEAN CONFERENCE ON COMPUTER VISION. **Anais...** [S.l.: s.n.], 2016. p.694–711.

KARRAS, T. et al. Progressive growing of gans for improved quality, stability, and variation. **arXiv preprint arXiv:1710.10196**, [S.l.], 2017.

KINGMA, D. P.; BA, J. Adam: a method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, [S.l.], 2014.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS. **Anais...** [S.l.: s.n.], 2012. p.1097–1105.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. **Neural computation**, [S.l.], v.1, n.4, p.541–551, 1989.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: ICML. **Proceedings...** [S.l.: s.n.], 2013. v.30, n.1, p.3.

MIRZA, M.; OSINDERO, S. Conditional generative adversarial nets. **arXiv preprint arXiv:1411.1784**, [S.l.], 2014.

MUSGRAVE, F. K.; KOLB, C. E.; MACE, R. S. The synthesis and rendering of eroded fractal terrains. In: ACM SIGGRAPH COMPUTER GRAPHICS. **Anais...** [S.l.: s.n.], 1989. v.23, n.3, p.41–50.

PASZKE, A. et al. Automatic differentiation in PyTorch. , [S.l.], 2017.

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

RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. **arXiv preprint arXiv:1511.06434**, [S.l.], 2015.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: convolutional networks for biomedical image segmentation. In: INTERNATIONAL CONFERENCE ON MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION. **Anais...** [S.l.: s.n.], 2015. p.234–241.

RUSNELL, B.; MOULD, D.; ERAMIAN, M. Feature-rich distance-based terrain synthesis. **The Visual Computer**, [S.l.], v.25, n.5-7, p.573–579, 2009.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, [S.l.], 2014.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS. **Anais...** [S.l.: s.n.], 2014. p.3104–3112.

TSUTSUI, S.; COLLET, P. **Massively parallel evolutionary computation on GPGPUs**. [S.l.]: Springer, 2013.

ULYANOV, D.; VEDALDI, A.; LEMPITSKY, V. Instance normalization: the missing ingredient for fast stylization. **arXiv preprint arXiv:1607.08022**, [S.l.], 2016.

VANEK, J. et al. Large-scale physics-based terrain editing using adaptive tiles on the GPU. **IEEE Computer Graphics and Applications**, [S.l.], v.31, n.6, p.35–44, 2011.

WANG, T.-C. et al. High-resolution image synthesis and semantic manipulation with conditional gans. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. **Proceedings...** [S.l.: s.n.], 2018. p.8798–8807.

YANG, L.-C.; CHOU, S.-Y.; YANG, Y.-H. MidiNet: a convolutional generative adversarial network for symbolic-domain music generation. **arXiv preprint arXiv:1703.10847**, [S.l.], 2017.