

Lucas da Silva dos Santos
Matheus Zanivan Andrade
Rafael Nascimento Lourenço

Geração procedural de mapas de ilhas 2d com biomas através de técnicas de segmentação de imagem

São Paulo - Brasil

2023

Lucas da Silva dos Santos
Matheus Zanivan Andrade
Rafael Nascimento Lourenço

Geração procedural de mapas de ilhas 2d com biomas através de técnicas de segmentação de imagem

Monografia apresentada na disciplina Trabalho de Conclusão de Curso, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.

Centro Universitário Senac - Santo Amaro
Bacharelado em Ciência da Computação

Orientador: Ainda não definido

São Paulo - Brasil
2023

Obtenha a ficha catalográfica junto a
biblioteca.
Substitua o arquivo ficha.pdf pela versão
obtida lá.

Lucas da Silva dos Santos
Matheus Zanivan Andrade
Rafael Nascimento Lourenço

Geração procedural de mapas de ilhas 2d com biomas através de técnicas de segmentação de imagem

Monografia apresentada na disciplina Trabalho de Conclusão de Curso, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.

Ainda não definido
Orientador

Professor
Convidado 1

Professor
Convidado 2

São Paulo - Brasil
2023

*Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.*

Resumo

Esta monografia descreve o desenvolvimento de uma ferramenta para jogos que oferece uma nova funcionalidade. A ferramenta começa com a seleção de uma foto, que é processada por um modelo de inteligência artificial especializado em segmentação panóptica. Isso permite a segmentação da imagem, incluindo a separação de objetos da mesma classe, como pessoas e carros. Após o modelo gerar uma saída, é possível selecionar um contorno detectado e, a partir disso, gerar um mapa 2D de ilha de forma procedural, combinado com o diagrama de Voronoi para criar os biomas do mapa.

Palavras-chaves: segmentação panóptica, geração procedural, diagrama de Voronoi, mapas, jogos.

Abstract

This monograph describes the development of a tool for games that offers new functionality. The tool starts with the selection of a photo, which is processed by an artificial intelligence model specialized in panoptic segmentation. This allows the segmentation of the image, including the separation of objects of the same class, such as people and cars. After the model generates an output, it is possible to select a detected contour and, from that, generate a 2D island map procedurally, combined with the Voronoi diagram to create the biomes of the map.

Key-words: panoptic segmentation, procedural generation, voronoi diagram, maps, games.

Listas de ilustrações

| | |
|---|----|
| Figura 1 – Mapa de tesouro do jogo Sea of Thieves | 19 |
| Figura 2 – Diagrama de Voronoi. | 24 |
| Figura 3 – Diagrama de Voronoi separado em solo e mar | 25 |
| Figura 4 – Diagrama de Voronoi separado em solo e mar com os cantos dos polígonos indicando a direção para o litoral | 26 |
| Figura 5 – Diagrama de Whittaker | 27 |
| Figura 6 – Resultado final da geração do mapa | 27 |
| Figura 7 – Algoritmos de detecção facial e de roupas/cabelos por cor localizam e reconhecem pessoas nesta imagem (Szeliski, 2022) | 28 |
| Figura 8 – Segmentação de instâncias de objetos pode delinear cada pessoa e objeto em uma cena complexa. (GMBH, 2023) | 28 |
| Figura 9 – Diagrama de dados de pixels. À esquerda, nossa imagem de Lincoln; no centro, os pixels rotulados com números de 0 a 255, representando sua luminosidade; e à direita, apenas esses números (Babich, 2020). | 29 |
| Figura 10 – Diagrama de aprendizado de máquina | 30 |
| Figura 11 – Diagrama de Venn sobre relação entre os tópicos de inteligência artificial | 30 |
| Figura 12 – Diagrama de Venn com aprendizado de máquina | 31 |
| Figura 13 – Ilustração da relação entre os principais tópicos de aprendizado de máquina | 31 |
| Figura 14 – Modelo de um neurônio não-linear. | 32 |
| Figura 15 – Gráfico da função <i>Sigmoid</i> | 33 |
| Figura 16 – Gráfico da função <i>ReLU</i> | 34 |
| Figura 17 – Gráfico da função <i>ReLU</i> | 34 |
| Figura 18 – Gráfico da função <i>Softmax</i> | 35 |
| Figura 19 – Gráficos mostrando subajuste, balanceado e sobreajuste respectivamente | 37 |
| Figura 20 – Comparação de uma rede neural convencional com uma rede neural profunda. | 38 |
| Figura 21 – Camadas principais de uma rede neural convolucional | 38 |
| Figura 22 – Tipos de segmentação em redes neurais convolucionais | 42 |
| Figura 23 – Exemplo de arquitetura de rede totalmente convolucional | 42 |
| Figura 24 – Arquitetura codificador-decodificador UNet | 43 |
| Figura 25 – Exemplo da classificação dos conjuntos usados nas métricas de segmentação | 45 |
| Figura 26 – Arquitetura geral do EfficientPS | 47 |
| Figura 27 – Imagem de entrada para rede neural. | 51 |
| Figura 28 – Imagem saída de um modelo de segmentação panóptica de segmentação panóptica. | 52 |

| | |
|---|----|
| Figura 29 – Ilha gerada a partir da segmentação panóptica e aplicando um filtro com o diagrama de Voronoi, azul representa oceano, verde floresta, cinza montanhas. | 52 |
| Figura 30 – Tela de início, botões de carregar imagem e carregar projeto, menu de contexto arquivos com 3 botões, carregar imagem, carregar projeto e salvar. | 53 |
| Figura 31 – Tela de processamento da segmentação | 54 |
| Figura 32 – Tela de seleção de segmentação da imagem. | 54 |
| Figura 33 – Tela de processamento para geração do mapa com a seleção do segmento. | 55 |
| Figura 34 – Tela de resultado com o mapa gerado após processamento. | 55 |

Lista de tabelas

| | |
|---|----|
| Tabela 1 – Top 15 modelos que melhor classificam pessoas com métrica P em segmentação panóptica | 46 |
|---|----|

Lista de abreviaturas e siglas

IA Inteligência Artificial

Sumário

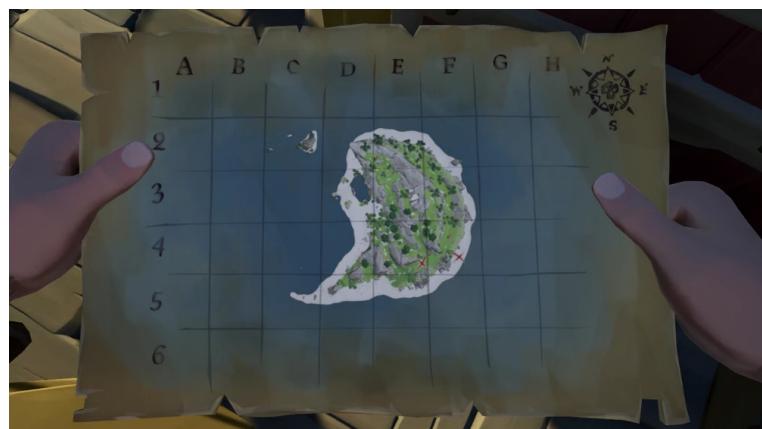
| | | |
|------------|--|-----------|
| 1 | INTRODUÇÃO | 19 |
| 1.1 | Objetivos | 20 |
| 2 | FUNDAMENTAÇÃO TEÓRICA | 23 |
| 2.1 | Geração procedural de conteúdo | 23 |
| 2.1.1 | Diagrama de Voronoi | 23 |
| 2.1.2 | Geração de biomas no diagrama de Voronoi | 24 |
| 2.2 | Visão computacional | 28 |
| 2.3 | Inteligência Artificial | 29 |
| 2.3.1 | Aprendizado de Máquina | 31 |
| 2.3.2 | Rede neural artificial | 32 |
| 2.3.3 | Aprendizado profundo | 37 |
| 2.3.4 | Redes neurais convolucionais | 38 |
| 2.3.5 | Segmentação | 41 |
| 2.3.6 | EfficientPS | 46 |
| 2.4 | Trabalhos relacionados | 48 |
| 3 | DESENVOLVIMENTO | 51 |
| 3.1 | Proposta | 51 |
| 3.2 | Cronograma | 51 |
| | REFERÊNCIAS | 57 |

1 Introdução

A indústria de jogos digitais cresce cada vez mais a cada dia, de acordo com a consultoria Newzoo ([SANTANA, 2022](#)), essa indústria tende a ultrapassar em 2023, os US\$ 200 bilhões (aproximadamente, R\$ 1 trilhão). Novos jogos são produzidos e publicados diariamente, e somente na plataforma digital Steam, foram 10.963 novos títulos em 2022 ([CLEMENT, 2023](#)).

Os mapas desempenham um papel fundamental nos jogos, fornecendo orientação aos jogadores e criando a sensação de escala em uma área. Por exemplo o jogo de aventura pirata chamado Sea of Thieves, os mapas revelam locais de interesse, como tesouros escondidos, missões e áreas perigosas. Na Figura 1 pode-se observar um exemplo de mapa dentro do jogo. Eles ajudam os jogadores a planejar suas estratégias, explorar o mundo virtual e tomar decisões com base em informações espaciais. Além disso, os mapas podem transmitir a sensação de escala e proporção, dando aos jogadores uma compreensão visual da extensão do mundo do jogo. Portanto os mapas enriquecem a experiência geral do jogo, mas cria-los pode ser um desafio, especialmente levando em consideração o orçamento disponível, uma vez que um jogo para chegar no consumidor final, passa por diversos processos de criação, o que demanda muito tempo e dinheiro ([W!N, 2023; FOFFANO, 2020](#)).

Figura 1 – Mapa de tesouro do jogo Sea of Thieves



Fonte: [Fandom \(2021\)](#)

Ademais, o mercado de jogos no Brasil teve um aumento de 2,5% em 2022, como apontado por uma pesquisa sobre o crescimento da demanda ([GIANNOTTI, 2022](#)). O custo de produção de jogos varia bastante, dependendo do tamanho e da complexidade do projeto, *e.g.*, a empresa Rockstar Games revelou que o jogo *Grand Theft Auto V* custou cerca de 265 milhões de dólares para ser desenvolvido e comercializado ([BAIRD, 2021](#)).

Apesar do rápido crescimento da indústria, existe uma carência de ferramentas que possam auxiliar os designers e artistas durante o processo de produção de jogos, o que acaba tornando-o demorado e, consequentemente, mais caro (ARAUJO, 2018). Segundo o livro "Procedural Content Generation in Games" (SHAKER; TOGELIUS; NELSON, 2016), uma abordagem eficiente para reduzir os custos de produção de um jogo é utilizar a geração procedural de conteúdo. Essa técnica permite maximizar o desenvolvimento de um jogo, envolvendo o uso de um software de computador capaz de criar conteúdo de jogos automaticamente. Esse software possibilita a geração automatizada de mapas, otimizando o processo de desenvolvimento.

De acordo com Lisboa (2022) é muito comum usar técnicas procedurais em jogos para otimizar o processo de criação combinado com inteligência artificial para melhorar ou personalizar a experiência do jogador. Por exemplo, o jogo RimWorld é um simulador de colônia que utiliza uma IA para gerar histórias de forma procedural, abrangendo aspectos como psicologia, ecologia, combate e diplomacia, dentre outros (LISBOA, 2022).

A aplicação da IA em jogos não se limita apenas à jogabilidade. Ela também é usada em áreas como animação de personagens, reconhecimento de fala e expressões faciais, tradução automática de idiomas nos diálogos do jogo e muito mais. A IA está impulsionando a inovação e a evolução dos jogos, proporcionando experiências cada vez mais envolventes e cativantes para os jogadores (MALAR, 2023; NVIDIA, 2021).

Outro ramo de IA que está em ascensão é o de segmentação de imagem com redes neurais convolucionais, onde é possível classificar os pixels de uma imagem além de ser possível criar uma máscara para cada objeto¹ detectado. As suas aplicações são diversas como por exemplo, carros ou drones autônomos, sistemas de vigilância, sistemas militares inteligentes, entre outros. Nessas aplicações é possível observar que é preciso ter um foco em identificar e segmentar seres humanos, por exemplo em carros autônomos é primordial essa tarefa para o carro tomar a decisão de frear quando estiver muito perto. Logo, se torna um tópico relevante dentro de visão computacional que pode ter diversas aplicações no mundo real, tal como segmentar uma imagem para detectar contornos e a partir disso criar um mapa personalizado (KIRILLOV et al., 2019; ULKU; AKAGÜNDÜZ, 2022).

No contexto da geração procedural de mapas, explorar a relação entre IA e personalização de jogos contribuirá para o avanço dessas áreas de pesquisa, proporcionando aos jogadores experiências mais ricas e variadas.

1.1 Objetivos

O objetivo principal deste trabalho é desenvolver uma ferramenta que ofereça uma alternativa para a geração procedural de mapas 2D de ilhas, utilizando o diagrama

¹ Todas classes que são contáveis como pessoas, carros, etc.

de Voronoi para a construção dos biomas. Além disso, pretende-se combinar técnicas de segmentação com redes neurais convolucionais para permitir a personalização desses mapas. Essa ferramenta terá a capacidade de reconhecer os contornos² de uma imagem selecionada e gerar um mapa que preserva fielmente o contorno escolhido.

Adicionalmente, os seguintes objetivos específicos serão abordados:

- Selecionar e analisar conjuntos de dados contendo classes relevantes, como pessoas, carros, entre outros, para treinar um modelo de rede neural convolucional específico para segmentação de imagens.
- Avaliar o desempenho geral do modelo usando a métrica de avaliação específica para o nicho de segmentação selecionado.
- Utilizar algoritmos para criar diagramas de Voronoi.
- Utilizar o resultado da segmentação para selecionar indicar o que é terreno em cima do diagrama de Voronoi.
- Gerar os biomas no diagrama de Voronoi.
- Aplicar um algoritmo para reconhecer a imagem com o contorno selecionado e gerar como resultado a imagem do mapa gerado.

² Os contornos reconhecidos são os classificados no conjunto de dados, logo o resultado terá uma detecção abrangente dentro do escopo de classes obtidas

2 Fundamentação teórica

Este capítulo apresenta os conceitos fundamentais necessários para a realização dos objetivos propostos na monografia. Os tópicos foram organizados na ordem em que são utilizados na ferramenta final. Primeiro, será apresentado o tópico de geração procedural para gerar o mapa requerido e o diagrama de Voronoi para ser aplicado como um filtro na imagem e para aplicar os biomas. Em seguida, o conceito geral de visão computacional e por fim, serão apresentados os conceitos de inteligência artificial, tanto em um contexto amplo quanto em relação ao conteúdo proposto, que é a segmentação panóptica.

2.1 Geração procedural de conteúdo

Segundo [Yannakakis e Togelius \(2018\)](#), em poucas palavras, a geração procedural de conteúdo são métodos e automações utilizados para gerar conteúdos em jogos. A geração procedural de conteúdo também é uma parte importante da inteligência artificial de um jogo e já vem sendo utilizada desde 1980. Essa técnica pode ser utilizada para gerar níveis, mapas, textura, regras de jogo, historia, entre outras coisas.

É difícil dizer qual algoritmo foi utilizado para geração de conteúdo dos jogos modernos e os códigos fontes não são facilmente acessíveis. Já nos jogos antigos os códigos fontes e as estratégias utilizadas são acessíveis e muito bem documentadas na internet. São geralmente utilizados algoritmos de geração aleatória que podem ser classificados como sendo de força bruta, e são usados para criar estruturas ou mapas dependendo do tipo de jogo ([DORMANS, 2010](#)).

2.1.1 Diagrama de Voronoi

Segundo [Rodrigues \(2019\)](#) diagrama de Voronoi é o particionamento do espaço onde cada região é associada a um ponto do conjunto.

O diagrama de Voronoi é gerado a partir das distâncias euclidianas entre os vizinhos de um conjunto de pontos do plano ([SANTOS, 2016](#)). Esse diagrama possui uma gama de utilizações, por exemplo, estudar epidemias, encontrar o ponto mais próximo, calcular a precipitação de uma área, estudar os padrões de crescimento das florestas, etc, ([POLÍGONOS..., 2018](#)).

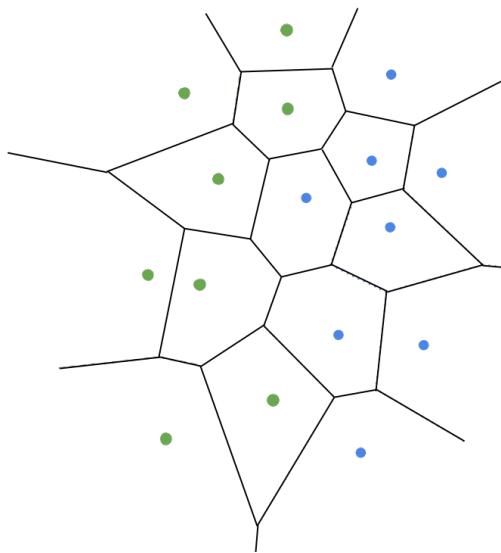
Seja um conjunto de índices $I_n = \{1, 2, 3, \dots, n\}$ e $A = \{p_1, p_2, \dots, p_n\} \subset \mathbb{R}^2$ um conjunto de pontos onde $2 \leq n < \infty$, define-se então como região de Voronoi o conjunto de pontos associado a p_i , onde d é a distância euclidiana

$$V(p_i) = \{p | d(p_i, p) \leq d(p_j, p); i \neq j, i, j \in I_n\}, \quad (2.1)$$

Tem-se um conjunto formado por essas regiões sendo $V(A) = V(1), V(2), \dots, V(n)$ ([RODRIGUES, 2019](#)).

Na figura [Figura 2](#) pode-se ver a relação do conjuntos de pontos com o diagrama de Voronoi.

Figura 2 – Diagrama de Voronoi.



Fonte: [Thomazthz \(2014\)](#)

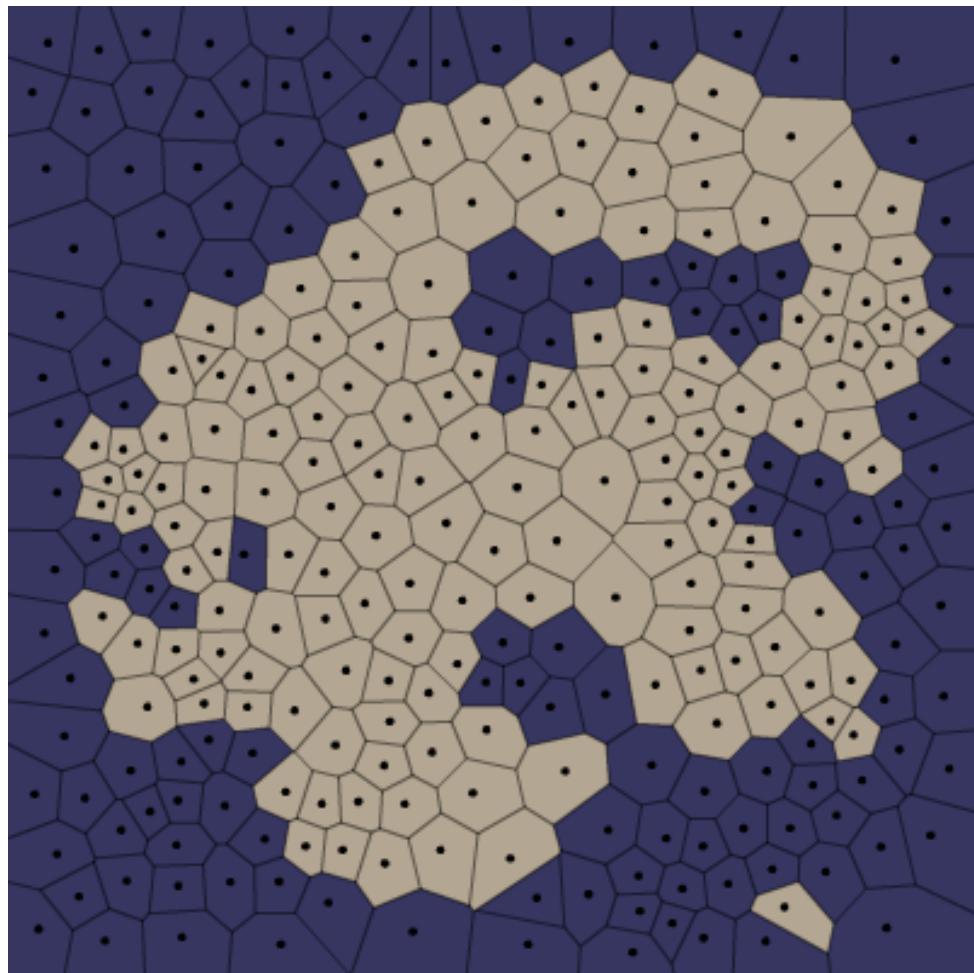
2.1.2 Geração de biomas no diagrama de Voronoi

Biomas são regiões ecológicas que possuem uma fauna e flora com atributos estruturais semelhantes ([BIOMAS..., s. d.](#)). Segundo [Patel \(2010\)](#) o primeiro passo para gerar o mapa e os biomas é gerar o litoral, os litorais serão as bordas que irão dizer o que é água e o que é solo. Existem algumas formas de gerar o formato da ilha:

- Radial: gera ilhas circulares através de ondas senoidais.
- Perlin: utiliza o Perlin Noise para controlar a forma da ilha.
- Quadrado: preenche o mapa inteiro com solo.

É possível utilizar qualquer formato para gerar as ilhas ([PATEL, 2010](#)), neste trabalho será utilizado o resultado da segmentação de imagem para gerar a ilha.

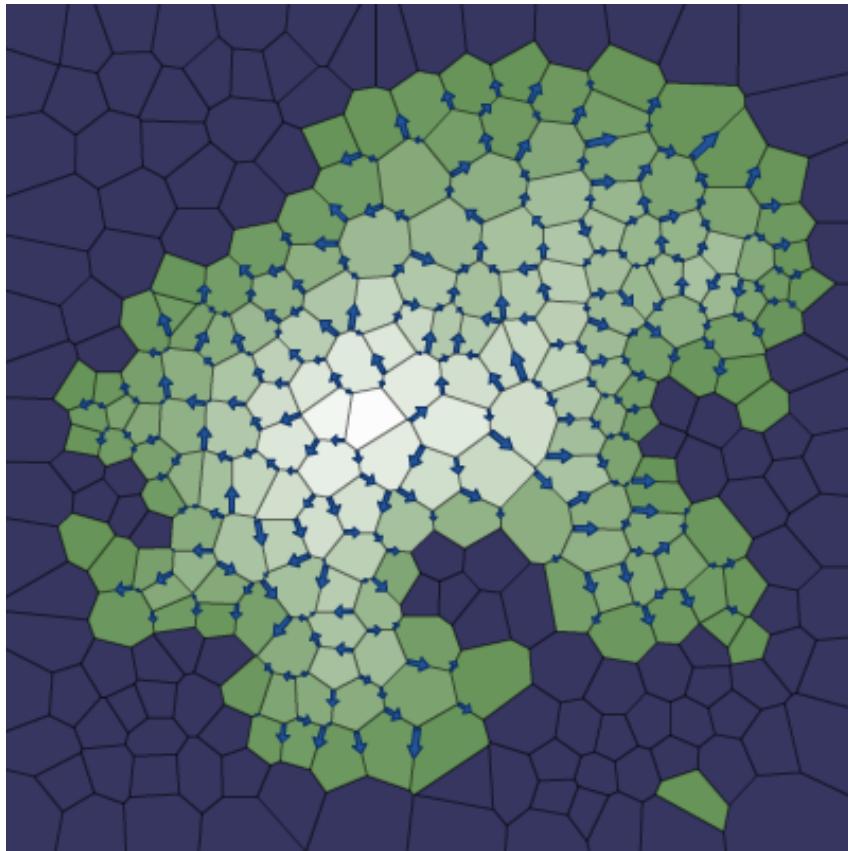
Figura 3 – Diagrama de Voronoi separado em solo e mar



Fonte: [Patel \(2010\)](#)

O proximo passo é calcular a elevação do terro. A elevação será calculada através da distancia de um polígono indicado como solo até o litoral, a elevação é definida pelos cantos dos polígonos ([PATEL, 2010](#)).

Figura 4 – Diagrama de Voronoi separado em solo e mar com os cantos dos polígonos indicando a direção para o litoral



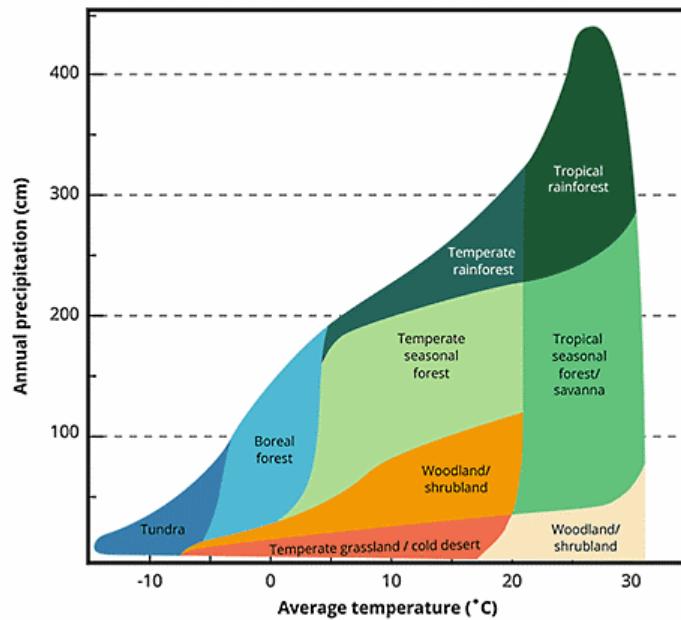
Fonte: [Patel \(2010\)](#)

Com a elevação é possível gerar os biomas, um exemplo seria elevações altas significa que é uma montanha, logo ela deve possuir neve. Adicionando mais uma camada, além da elevação, como a de umidade, podemos gerar uma variedade maior de biomas. A umidade é calculada de quão longe o polígono está de um corpo d'água.

Diagrama de Whittaker

O diagrama de Whittaker é uma forma de dividir os terrenos gerados a partir da técnica de geração procedural, esse diagrama inclui valores de temperatura e umidade para separar os biomas ([WHITTAKER... , 2018](#)).

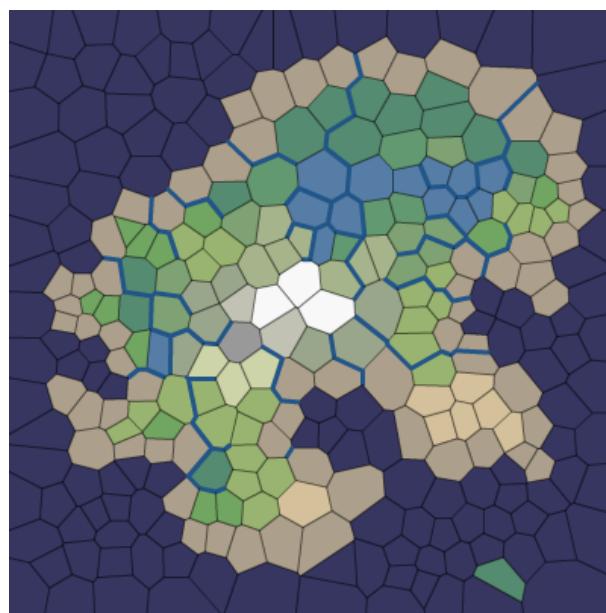
Figura 5 – Diagrama de Whittaker



Fonte: Mendes (2019)

Usando a elevação como representante da temperatura de um bioma é possível utilizar o diagrama de Whittaker, fazendo alterações nesse diagrama é possível adicionar ou remover biomas. Com essa nova camada possibilita a adição de rios ao mapa (PATEL, 2010) e assim obtendo o resultado apresentado na figura 6.

Figura 6 – Resultado final da geração do mapa



Fonte: Patel (2010)

Para utilizar essa técnica de geração procedural de mapas, é necessário começar com uma imagem que sirva como base para a geração do mapa. É empregada a visão computacional para extrair dados da imagem.

2.2 Visão computacional

A visão computacional está em constante avanço, aproximando cada vez mais os computadores da capacidade visual humana. De acordo com Horst Haußecker e Bernd Jähne, no livro "Computer Vision and Applications" ([HAUßECKER BERND JÄHNE, 1999](#)), a visão computacional é uma área da computação que se dedica à interpretação de imagens por meio de algoritmos e técnicas de processamento de imagens. Essa área abrange a aquisição, processamento e análise de imagens, com o objetivo de extrair informações úteis para resolver problemas específicos.

Porém, segundo Richard Szeliski, no livro "Computer Vision: Algorithms and Applications" ([SZELISKI, 2022](#)), nas últimas décadas ocorreram avanços significativos na busca de aproximar a visão computacional da visão humana, porém não obteve total êxito. Isso ocorre porque, enquanto o olho humano enxerga com aparente facilidade as estruturas tridimensionais e suas nuances, a visão computacional depende de técnicas matemáticas altamente precisas para recuperar a forma tridimensional e a aparência dos objetos.

Nas figuras Figura 7 e Figura 8, evidencia-se a notável capacidade de um computador em distinguir, classificar e até mesmo compreender os elementos presentes em uma fotografia.

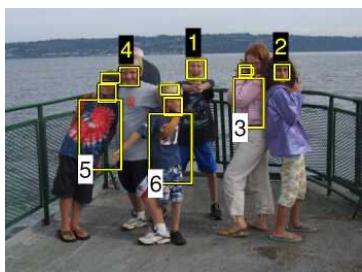


Figura 7 – Algoritmos de detecção facial e de roupas/cabelos por cor localizam e reconhecem pessoas nesta imagem ([SZELISKI, 2022](#))



Figura 8 – Segmentação de instâncias de objetos pode delinear cada pessoa e objeto em uma cena complexa. ([GMBH, 2023](#))

No entanto, apesar do sucesso no uso dessas técnicas, o computador ainda não consegue oferecer a mesma quantidade de detalhes na explicação de uma imagem como o olho humano. Isso se deve à maior facilidade do computador em compreender linguagem em comparação à visualização. A tarefa de ensinar um computador a ver e descrever

com precisão e riqueza de detalhes o que está sendo observado é extremamente complexa ([Szeliski, 2022](#)).

A visão é um elemento crucial para capacitar a inteligência artificial a realizar diversas tarefas. A fim de replicar a visão humana, é necessário que as máquinas sejam capazes de adquirir, processar, analisar e compreender imagens. ([Marr, 2019](#))

No processamento de computação visual, as imagens são adquiridas e representadas como uma matriz 2D de pixels. Cada pixel corresponde a um ponto na imagem e é representado por um valor numérico que varia de 0 a 255. Esses valores de pixel descrevem a intensidade da cor em uma escala de cinza. Dessa forma, um computador interpreta uma imagem como uma matriz de números, permitindo que ele analise e compreenda os detalhes visuais presentes na imagem, como no caso da Figura 9 do presidente dos Estados Unidos, Abraham Lincoln([AMINI, 2023](#)).

Os algoritmos de visão computacional utilizados atualmente são fundamentados em reconhecimento de padrões. O procedimento consiste em treinar computadores por meio de uma vasta quantidade de dados visuais. Os computadores processam imagens, rotulam os objetos nelas contidos e identificam padrões entre esses objetos ([Babich, 2020](#)).

Esse processo de treinamento e reconhecimento de padrões permite que os computadores identifiquem objetos e compreendam seu contexto visual. Com essa capacidade, o computador consegue realizar tarefas como, por exemplo, reconhecimento facial Figura 7.

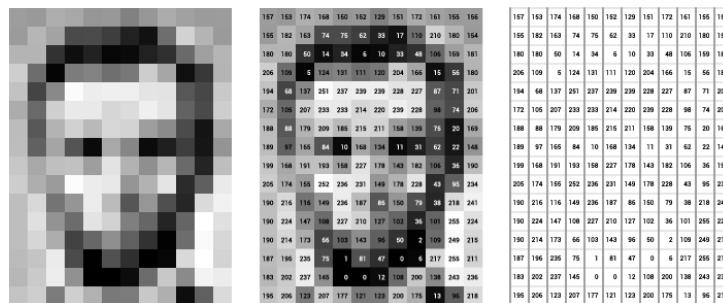


Figura 9 – Diagrama de dados de pixels. À esquerda, nossa imagem de Lincoln; no centro, os pixels rotulados com números de 0 a 255, representando sua luminosidade; e à direita, apenas esses números ([Babich, 2020](#)).

É bastante comum na área de visão computacional contarmos com o auxílio de algoritmos de inteligência artificial que capacitam o computador a reconhecer padrões e características nas imagens processadas.

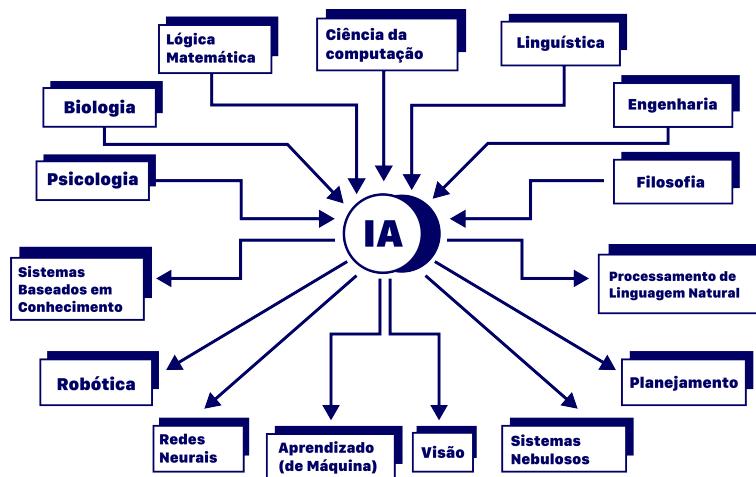
2.3 Inteligência Artificial

Inteligência artificial é uma técnica científica que simula o pensamento humano de forma que possa ser executado em uma máquina, podendo ser utilizada para criar soluções

com uma linha de progressão parecida ao raciocínio lógico como comumente conhecida. Isto permite ao computador reconhecer e interpretar o mundo ao redor com imagens e textos criando uma ampla área de atuação que otimiza tarefas antes só realizadas por seres humanos (SILVA; MAIRINK, 2019).

Este ramo é complexo por se tratar de uma representação cognitiva, se torna necessário usar uma base com diversas áreas científicas como psicologia, biologia, lógica matemática, linguística, engenharia, filosofia, entre outras. E pode ser usado para diversos problemas específicos como, por exemplo, definir as boas rotas para algum processo logístico (GOMES, 2010).

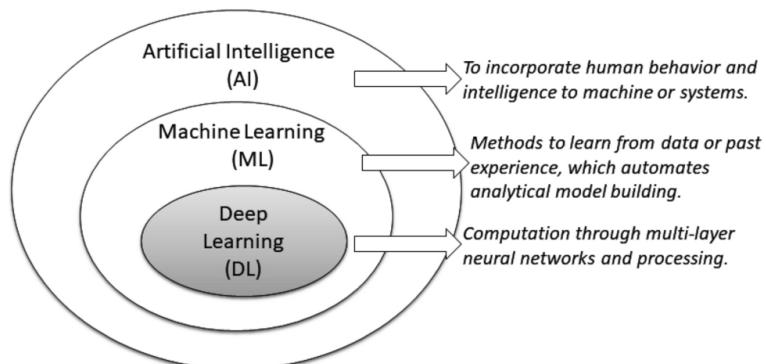
Figura 10 – Diagrama de aprendizado de máquina



Fonte: MONARD e BARANAUKAS (2000)

Segundo Sarker (2021) existe três tópicos sobre inteligência artificial muito populares, sendo eles: inteligência artificial, aprendizado de máquina e aprendizado profundo como mostrado na Figura 11.

Figura 11 – Diagrama de Venn sobre relação entre os tópicos de inteligência artificial



Fonte: Sarker (2021)

2.3.1 Aprendizado de Máquina

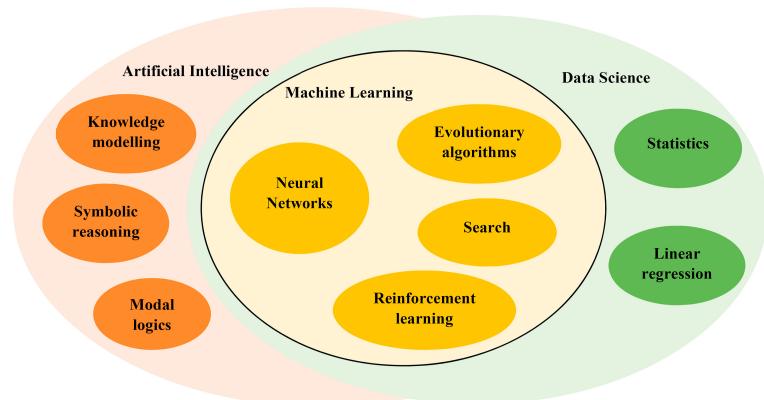
Segundo Woschank, Rauch e Zsifkovits (2020), aprendizado de máquina é uma subcategoria de inteligência artificial que se refere a detecção de padrões importantes de uma base de dados. As ferramentas utilizadas aumentam a eficiência dos algoritmos para lidar com bases de dados grandes.

Portanto, essa técnica permite ao computador melhorar os resultados com base na experiência, isso indica uma relação direta entre o quanto o programa consumiu de dados e qualidade da solução do problema (BROWN, 2021).

Dentro desse nicho existem outros como: redes neurais, algoritmos evolucionários, algoritmos de busca, aprendizado por reforço, dentre outros. (SIRCAR et al., 2021).

Existe relação direta de conceitos entre inteligência artificial, aprendizado de máquina e ciência de dados conforme mostrado na Figura 12.

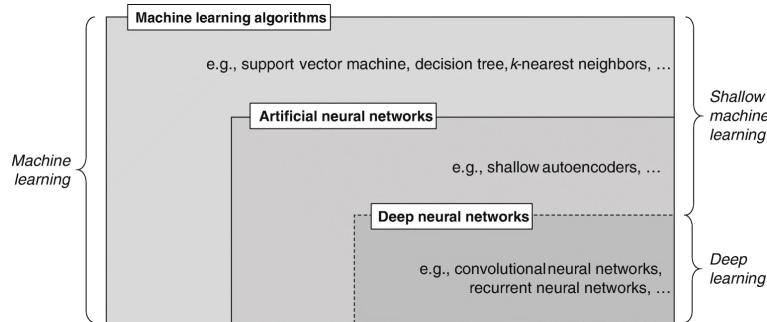
Figura 12 – Diagrama de Venn com aprendizado de máquina



Fonte: Sircar et al. (2021)

É possível observar uma hierarquia entre aprendizado de máquina e os principais termos sendo eles redes neurais artificiais e aprendizado profundo com base em Janiesch, Zschech e Heinrich (2021) mostrado no diagrama da Figura 13.

Figura 13 – Ilustração da relação entre os principais tópicos de aprendizado de máquina

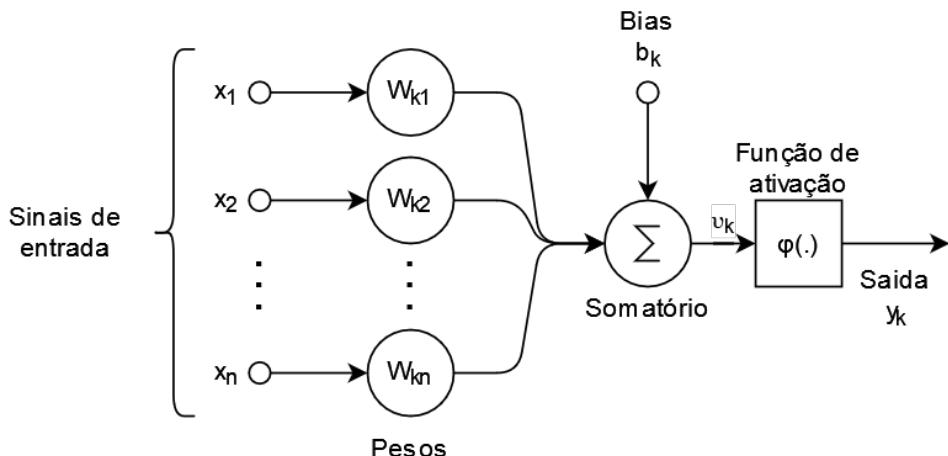


Fonte: Janiesch, Zschech e Heinrich (2021)

2.3.2 Rede neural artificial

Uma rede neural artificial é uma representação matemática de unidades de processamento conectadas chamadas de neurônios artificiais. Essa arquitetura simula sinapses, cada sinal trocado entre os neurônios pode aumentar ou atenuar os sinais de outros durante o aprendizado (JANIESCH; ZSCHECH; HEINRICH, 2021).

Figura 14 – Modelo de um neurônio não-linear.



Fonte: Haykin (1999)

Observando a figura 14 é possível observar o funcionamento de um neurônio k . Os sinais de entradas são partes de um vetor x de tamanho n , sendo o vetor composto por $x_1, x_2 \dots x_n$. Essas componentes são combinadas em uma soma ponderada utilizando seus respectivos pesos, $w_{k1}, w_{k2} \dots w_{kn}$, formando assim a seguinte equação (MARTI; BARROS, 2017 apud HAYKIN, 1999):

$$v_k = \sum_{i=1}^n (x_i * w_{ki}) \quad (2.2)$$

O resultado dessa equação produz o potencial de ativação v_k , esse resultado é somado com o *bias* ou viés b_k para manipular a saída y_k do neurônio, essa soma éposta em uma função não-linear nomeada de função de ativação $\varphi(\cdot)$, essas funções mapeiam a saída em um intervalo $[0, 1]$ ou $[1, -1]$. A função de saída pode ser representada com a seguinte equação (MARTI; BARROS, 2017 apud HAYKIN, 1999):

$$y_k = \varphi(v_k + b_k) \quad (2.3)$$

O aprendizado ocorre na fase de treinamento onde é ajustando os pesos w_k e o viés b_k de cada neurônio k . Os pesos w_k são utilizados para calcular a taxa de crescimento da função e o viés b_k é necessário para descolar a saída da função. Com isso é possível modelar uma função linear $y = w^T * x + b$ (MARTI; BARROS, 2017).

Para cada amostra o modelo compara os resultados dos valores atuais dos pesos w_k e viés b_k com o resultado esperado (alvo). Uma função de perda é utilizada para gerar um vetor de gradientes e para quantificar o erro encontrado para a configuração atual do modelo. O modelo atualiza os pesos w_k e os viés b_k no sentido contrário do vetor de gradientes, buscando minimizar a função de custo de acordo com uma taxa de aprendizado(*learning rate*), esse processo é chamado de retropropagação — ou backpropagation — que será explicado melhor posteriormente (MARTI; BARROS, 2017).

Ao combinar diversos neurônios artificiais forma-se uma rede neural artificial. Essas redes buscam simular o processamento de informação do cérebro humano (FERNEDA, 2006). Nas redes neurais os neurônios são organizados em grupos de unidade de processamento chamados camadas. A primeira e a última camada são nomeadas de camada de entrada e camada de saída e as demais de camadas ocultas. As camadas mais próximas da entrada são responsáveis por identificar características mais primitivas e as seguintes combinam essas informações para identificar padrões mais complexos (MARTI; BARROS, 2017).

Nas redes neurais os neurônios são organizados em grupos de unidade de processamento chamados camadas. A primeira e a última camada são nomeadas de camada de entrada e camada de saída e as demais de camadas ocultas. As camadas mais próximas da entrada são responsáveis por identificar características mais primitivas e as seguintes combinam essas informações para identificar padrões mais complexos (MARTI; BARROS, 2017).

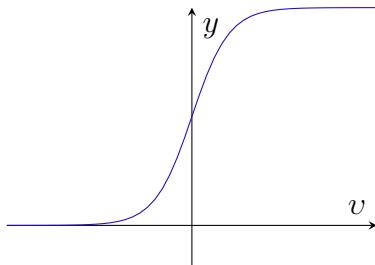
Função de ativação

A função de ativação retorna a saída de um neurônio (HAYKIN, 1999), aqui pode-se ver três tipos de funções de ativação:

- Função *Sigmoid*, uma função não-linear que produz uma curva com a forma de "S". Usada para mapear valores previstos em probabilidades. Tem o valor de saída entre 0 e 1 ([GHARAT, 2019](#)).

Figura 15 – Gráfico da função *Sigmoid*.

$$\varphi(v) = \frac{1}{1 + e^{-v}} \quad (2.4)$$



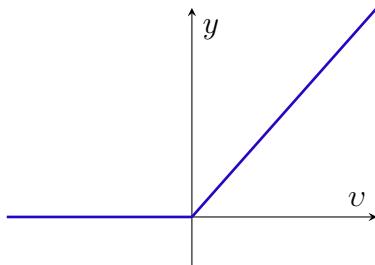
Fonte: Criação própria

Segundo [Gharat \(2019\)](#), a função *Sigmoid* tem uma convergência lenta, é computacionalmente cara e para valores muito extremos causa problemas na previsão.

- Função *ReLU* (Unidade Linear Retificada), função não-linear inspirada nos neurônios do cérebro que retorna um valor positivo ou 0 ([RIZZO; CANATO, 2020](#)).

Figura 16 – Gráfico da função *ReLU*.

$$\varphi(v) = \max(0, v) \quad (2.5)$$



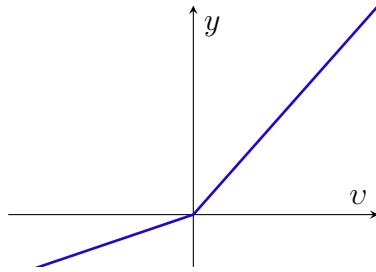
Fonte: Criação própria

A função *ReLU* é computacionalmente eficiente e converge rapidamente, porém quando a entrada da função se aproxima de zero a rede neural não consegue executar o retropropagação, sendo assim não há aprendizado ([GHARAT, 2019](#)).

- Função *Leaky ReLU* (Unidade Linear Retificada com Vazamento), função não-linear variante da *ReLU* que retorna um valor positivo ou v/a_i , sendo a_i um valor na faixa $(1, \infty)$ ([XU et al., 2015](#)).

Figura 17 – Gráfico da função *ReLU*.

$$\varphi(v) = \begin{cases} v & \text{if } v_k \geq 0 \\ \frac{v}{a_k} & \text{if } v_k < 0 \end{cases} \quad (2.6)$$



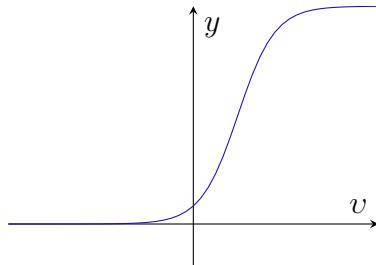
Fonte: Criação própria

Possui as mesmas características da função *ReLU*, mas sem o problema da retropropagação. ([GHARAT, 2019](#)).

4. Função *Softmax*, calcula a distribuição de probabilidades de um evento em "n" eventos e fornece a probabilidade do valor de entrada pertencer a uma classe específica, geralmente usada na camada de saída ([GHARAT, 2019](#)).

Figura 18 – Gráfico da função *Softmax*.

$$\varphi(v) = \frac{e^{v_i}}{\sum_{j=0} e^{v_j}} \quad (2.7)$$



Fonte: Criação própria

Com a função *Softmax* é possível normalizar a saída para valores entre 0 e 1, bem como calcular a probabilidade da entrada, e por causa dessas características é utilizada na camada de saída da rede neural ([GHARAT, 2019](#)).

Função de perda

A função de perda é calculada na camada de saída e serve para mensurar o sucesso obtido comparando com fórmulas o resultado da arquitetura com o resultado real do conjunto de dados. O resultado dessa função irá ajudar na retropropagação, *i.e.*, servirá para ajustar os pesos e vieses da conexão entre os neurônios para minimizar o erro. A seguir algumas funções de perda, pontuando que todo esse subtópico é baseado em [Alzubaidi et al. \(2021\)](#).

Softmax ou entropia cruzada ou logarítmica

Muito utilizada para medir a performance de uma rede neural convolucional principalmente quando o resultado tem várias classes. Antes dessa função de perda é necessário usar a função de ativação softmax descrita na Figura 18 pois precisa de uma saída dentro de uma distribuição de probabilidade. Sendo N o número de classes ou o número de neurônios na camada de saída.

$$H(p, y) = - \sum_{i=1}^N y_i \log(p_i) \quad (2.8)$$

Euclidiana ou erro quadrático médio

Muito utilizada para problemas de regressão.

$$H(p, y) = \frac{1}{2N} \sum_{i=1}^N (p_i - y_i)^2 \quad (2.9)$$

Hinge

Muito utilizado para classificação binária.

$$H(p, y) = \sum_{i=1}^N \max(0, m - (2y_i - 1)p_i) \quad (2.10)$$

Retropropagação

De acordo com [Brilliant.org \(2023\)](#) o algoritmo geral de retropropagação é:

1. Propagação: calcular os pares de entrada-saída (\vec{x}_d, y_d) — \vec{x}_d é o vetor de entrada e y_d a saída verdadeira — e guardar os resultados \hat{y}_d — a saída encontrada no treinamento —, a_j^k, o_j^k para cada neurônio j na camada k , indo da camada de entrada para camada de saída.
2. Retropropagação: Calcular os pares de entrada-saída (\vec{x}_d, y_d), chegando na fórmula $\frac{\partial E_d}{\partial w_{ij}^k}$ que é a derivada parcial do erro total. Na representação E_d é a função de perda e w_{ij}^k é o peso conectado em um neurônio de $k - 1$. Outra forma de representar é $\delta_j^k o_i^{k-1}$ e suas variáveis são: δ_j^k representa o erro do neurônio e o_i^{k-1} representa a saída do neurônio na camada $k - 1$. Essa técnica começa na camada de saída e propaga até a última camada escondida.

$$\frac{\partial E_d}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} \quad (2.11)$$

3. Combinar gradientes individuais: Uma média simples é feita com todos resultados de $\frac{\partial E_d}{\partial w_{ij}^k}$ formando assim o gradiente total representado como $\frac{\partial E(X, \theta)}{\partial w_{ij}^k}$.

$$\frac{\partial E(X, \theta)}{\partial w_{ij}^k} = \frac{1}{N} \sum_{d=1}^N \frac{\partial E_d}{\partial w_{ij}^k} \quad (2.12)$$

4. Atualiza os pesos: usando α como taxa de aprendizado e o gradiente total $\frac{\partial E(X, \theta)}{\partial w_{ij}^k}$ tem-se a seguinte equação.

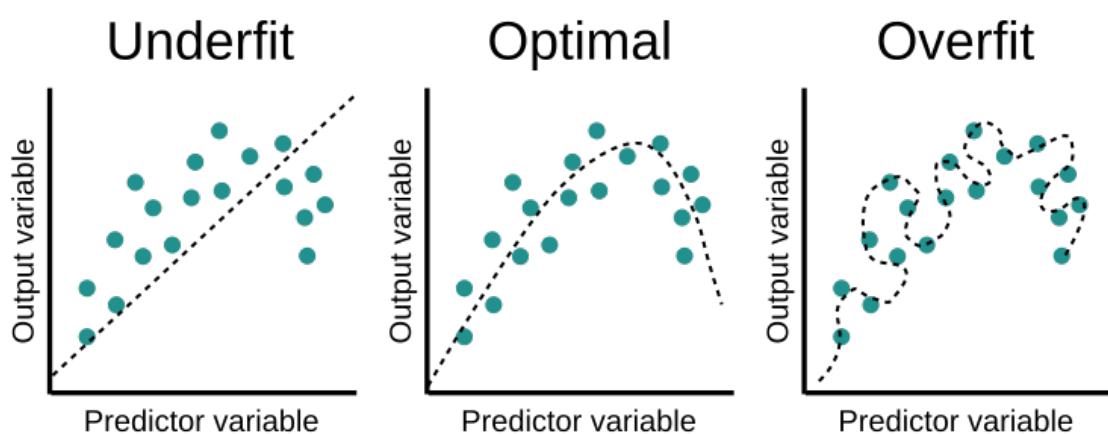
$$\Delta w_{ij}^k = -\alpha \frac{\partial E(X, \theta)}{\partial w_{ij}^k} \quad (2.13)$$

Observação: basta trocar w_{ij}^k para b_{ij}^k — trocar o peso pelo viés — em todo algoritmo para ajustar o viés do modelo com a técnica de retropropagação.

Regularização

Quando se monta uma arquitetura de redes neurais pode se chegar em três casos, sendo eles: subajuste (underfit), balanceado (optimal) e sobreajuste (overfit). O sobreajuste é quando no treinamento o modelo acerta as classes porém nos testes não, isso mostra uma dificuldade em generalizar as características. Já o subajuste não consegue pontuar bem em nenhum caso mostrando que o conjunto de dados de treinamento está pequeno para detectar padrões. Por outro lado, o balanceado é quando produz resultados bons tanto no conjunto de dados de treinamento quanto no de testes (ALZUBAIDI et al., 2021; TAYE, 2023).

Figura 19 – Gráficos mostrando subajuste, balanceado e sobreajuste respectivamente



Fonte: [Educative \(2022\)](#)

2.3.3 Aprendizado profundo

O aprendizado profundo é uma área do aprendizado de máquina caracterizada por utilizar dados brutos como entrada e descobrir as representações necessárias para permitir o mapeamento adequado e assim tornando as soluções mais simples (MARTI; BARROS, 2017 apud LECUN; BENGIO; HINTON, 2015).

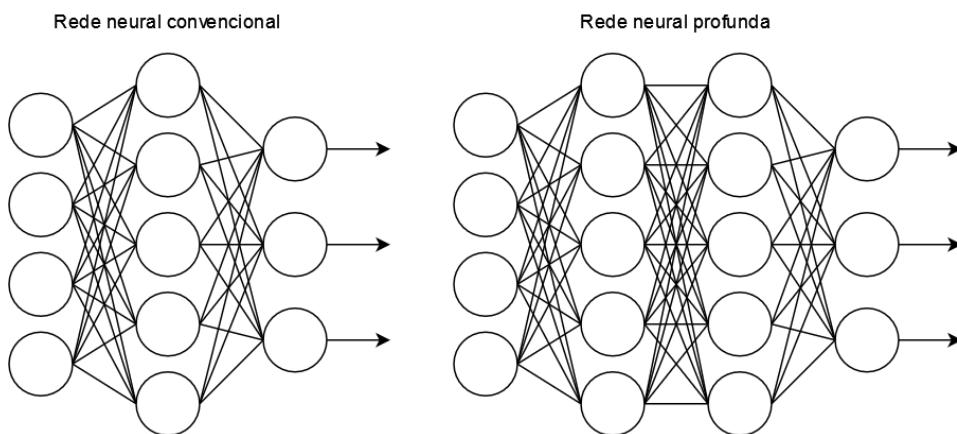
Segundo LeCun, Bengio e Hinton (2015), o aprendizado profundo são métodos de representação de aprendizado com vários níveis, obtidos por meio da decomposição de módulos simples e lineares, que transformam a representação de um nível em uma representação mais alta e abstrata. Por exemplo a representação de uma imagem é transformada em informações que identificam objetos.

Dividindo um problema complexo em problemas menores torna os métodos especializados, viabilizando tarefas mais complexas, depois essas tarefas que foram divididas são recombinaadas e é gerado uma solução do problema (MARTI; BARROS, 2017).

Utilizando o exemplo anterior, reconhecimento de imagem, cada um desses métodos especializados seria responsável por reconhecer uma parte da imagem, como bordas, objetos, tamanho, etc. E após a junção desses métodos é feito a predição da imagem (MARTI; BARROS, 2017).

A principal diferença entre uma rede neural convencional e uma rede neural profunda é a quantidade de camadas, uma rede neural profunda possui várias camadas de processamento (MARTI; BARROS, 2017 apud HAYKIN, 1999).

Figura 20 – Comparação de uma rede neural convencional com uma rede neural profunda.



Fonte: Criação própria

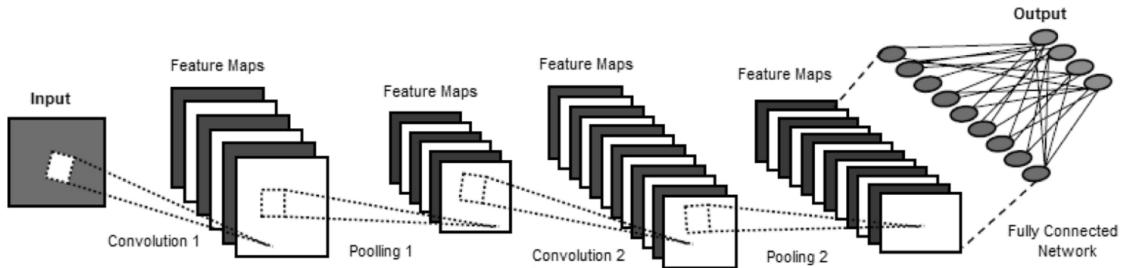
2.3.4 Redes neurais convolucionais

Uma rede neural convolucional é análoga à rede neural artificial, i.e., feita de neurônios que otimizam o aprendizado através dele mesmo. A principal diferença é que a

rede neural convolucional é amplamente utilizada em soluções que detectam padrões em imagens, logo existem funcionalidades específicas da própria arquitetura para essa tarefa (O'SHEA; NASH, 2015).

Uma arquitetura básica de uma rede neural convolucional tem as seguintes camadas: convolucional, agrupamento e totalmente conectada (SARKER, 2021).

Figura 21 – Camadas principais de uma rede neural convolucional



Fonte: Sarker (2021)

Camada convolucional

Segundo Taye (2023) camada convolucional é essencial para esse tipo de arquitetura e usa um filtro — ou kernel — para aplicar na imagem e direcionar para o próximo neurônio. Esse filtro é uma matriz de números que terá uma operação aplicada em todos os píxeis da imagem — que também é representado por matriz(es) — as informações cruciais para esse filtro são: tamanho, largura e pesos. Isto é utilizado para extrair características com uma base matemática, criando uma relação direta entre um píxel e os píxeis ao redor. Os pesos começam de forma pseudoaleatórias e são ajustados no decorrer do aprendizado. O resultado dessa camada é chamado de mapa de características. O tamanho da saída será baseado na fórmula abaixo sendo os tamanhos I da imagem, F do filtro e a S da saída (TAYE, 2023).

$$\begin{aligned} \mathbf{I}x - \mathbf{F}x + 1 &= \mathbf{S}x \\ \mathbf{I}y - \mathbf{F}y + 1 &= \mathbf{S}y \end{aligned} \quad (2.14)$$

A seguir um exemplo dos passos para construir a matriz resultante baseado em Alzubaidi et al. (2021).

| Matriz 2x4 | Filtro 2x2 | Resultado |
|--|------------|---|
| $\begin{bmatrix} 0 & 2 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$ | \otimes | $\begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix}$ |
| | | $=$ |
| | | $\begin{bmatrix} 1 & - & - \end{bmatrix}$ |

$$\begin{bmatrix} 0 & \boxed{2 & 1} & 0 \\ 1 & \boxed{0 & 0} & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & \boxed{1} & - \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & \boxed{1 & 0} \\ 1 & 0 & \boxed{0 & 1} \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & \boxed{2} \end{bmatrix}$$

Tamanho do passo e preenchimento

O Tamanho do passo — ou stride — serve para especificar a distância de pixels entre os passos da camada. No exemplo acima esse parâmetro é definido como 1, por isso a matriz selecionada pula 1 pixel para direita entre os passos. Esse valor altera o tamanho da matriz resultante ([SARKER, 2021](#)).

O preenchimento — ou padding — é uma técnica utilizada para manter o mesmo tamanho da entrada, adicionando bordas com zeros antes das operações da camada para ter como saída uma matriz da mesma dimensão da matriz original. Isso é usado devido a desvantagem em perder os detalhes nas bordas das imagens no processamento de uma camada ([SARKER, 2021](#)).

Camada de agrupamento

A camada de agrupamento — ou pooling — tem como tarefa primordial uma técnica para reduzir o tamanho do mapa de características, porém preservando os padrões mais relevantes. Dentre os recursos essenciais dessa camada estão o tamanho do agrupamento e a operação que será realizada. O maior problema dessa camada é pelo fato dela apenas identificar onde essas características estão e não se tem ou não, *i.e.*, dependendo de qual operação e a quantidade de camadas pode não ser possível guardar as principais características de forma íntegra causando uma redução no desempenho final da predição ([SARKER, 2021](#)).

Existem vários tipos de agrupamento, os mais utilizados são: agrupamento máximo, agrupamento médio e agrupamento global médio que estão explicados abaixo em exemplos baseados em [Alzubaidi et al. \(2021\)](#).

Agrupamento máximo

É definido o resultado com base no máximo encontrado pelo tamanho do agrupamento, exemplo a seguir usando um mapa de características com tamanho 4x4 e agrupamento de tamanho 2x2.

$$\left[\begin{array}{cc|cc} 4 & 25 & 44 & 10 \\ 8 & 14 & 8 & 33 \\ \hline 17 & 2 & 16 & 34 \\ 5 & 13 & 24 & 7 \end{array} \right] = \begin{bmatrix} 25 & 44 \\ 17 & 34 \end{bmatrix}$$

Agrupamento médio

É definido o resultado com base na média encontrada pelo tamanho do agrupamento, exemplo a seguir usando um mapa de características com tamanho 4x4 e agrupamento de tamanho 2x2.

$$\left[\begin{array}{cc|cc} 4 & 25 & 44 & 10 \\ 8 & 14 & 8 & 33 \\ \hline 17 & 2 & 16 & 34 \\ 5 & 13 & 24 & 7 \end{array} \right] = \begin{bmatrix} 12 & 23 \\ 9 & 20 \end{bmatrix}$$

Agrupamento global médio

É definido o resultado com base na média geral do mapa o que sempre tem como saída uma matrix 1x1, exemplo a seguir usando um mapa de características com tamanho 4x4.

$$\left[\begin{array}{cccc} 4 & 25 & 44 & 10 \\ 8 & 14 & 8 & 33 \\ 17 & 2 & 16 & 34 \\ 5 & 13 & 24 & 7 \end{array} \right] = [16]$$

Camada totalmente conectada

A camada totalmente conectada geralmente é utilizada no final da arquitetura e cria a partir de cada neurônio uma ligação direta para cada etiqueta final. Isso torna essa camada extremamente pesada computacionalmente. O número de neurônios dessa camada é equivalente ao número de classes propostas. Além disso é quando chega nessa camada que a função de perda é calculada e se inicia a retropropagação ([ALZUBAIDI et al., 2021](#); [TAYE, 2023](#)).

Aperfeiçoamento

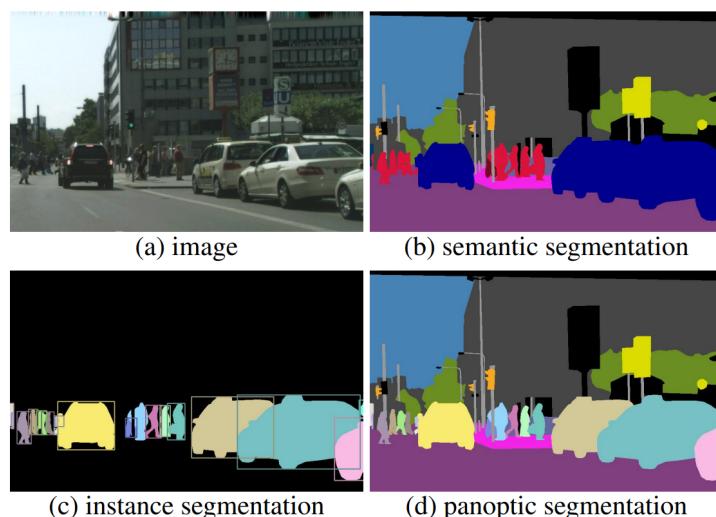
Segundo [Alzubaidi et al. \(2021\)](#), [Taye \(2023\)](#) existem algumas técnicas para aperfeiçoar os resultados do modelo, sendo elas:

- Dropout: Muito utilizada para evitar sobreajuste pois está técnica irá desligar um neurônio aleatoriamente colocando a saída dele como zero no processo de treinamento e portanto forçara o modelo a aprender a identificar características diferentes em outros neurônios possibilitando a generalização do modelo.
- Aumentar o tamanho do conjunto de dados: caso não seja possível criar ou encontrar um maior existem técnicas para aumentar artificialmente acrescentando pequenas mudanças nas imagens existentes, algumas são rotacionar, recortar e inverter horizontalmente ou verticalmente.
- Normalização em lote: normaliza os pesos para evitar o problema de mudança interna de covariância
- Aumentar o tempo de treinamento
- Aumentar a profundidade ou largura da arquitetura
- Ajustar os hiperparâmetros

2.3.5 Segmentação

O estudo de segmentação semântica dentro da área de redes neurais convolucionais têm três principais nichos, sendo eles: segmentação semântica que é a classificação por pixel, a segmentação de instância que atribui um id para cada objeto encontrado de uma classe, e a segmentação panóptica que junta as duas anteriores para criar uma imagem semelhante a saída de segmentação semântica porém separando objetos de mesma classe sendo essa a mais recente e completa, a diferença entre esses três tipos está ilustrado na Figura 22 ([ULKU; AKAGÜNDÜZ, 2022](#); [WANGENHEIM, 2021](#)).

Figura 22 – Tipos de segmentação em redes neurais convolucionais

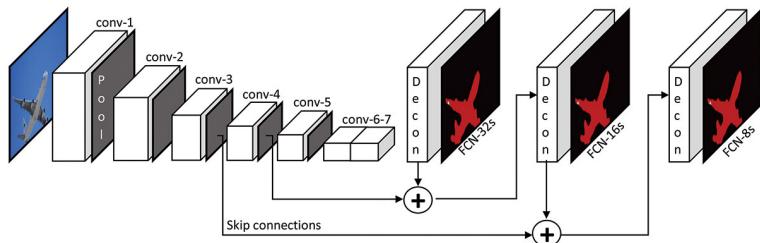


Fonte: Kirillov et al. (2019)

Segmentação semântica

A segmentação semântica começou a ter resultados satisfatórios a partir de redes totalmente convolucionais, com o objetivo de segmentar imagens classificando pixels, esse modelo descarta a camada totalmente conectada pois a saída deverá ser uma imagem e não uma classificação — isso a torna mais rápida para treinar do que as redes neurais convolucionais —, logo usa camadas deconvolucionais para transformar a matriz de características em uma imagem de qualquer dimensão na saída. A RTC criou a arquitetura chamada de salto (ou conexões) que serve para evitar perdas em camadas de agrupamento criando conexões entre camadas não consecutivas — geralmente entre camadas convolucionais e deconvolucionais — como apresentado na Figura 23, a arquitetura de salto evoluiu para arquitetura codificador-decodificador.

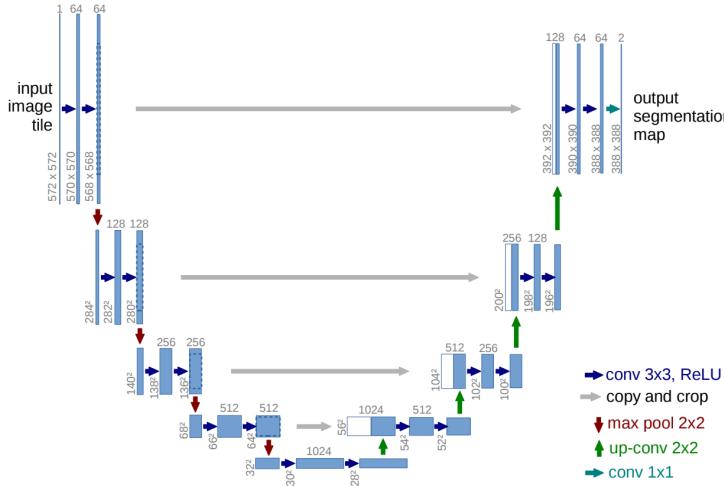
Figura 23 – Exemplo de arquitetura de rede totalmente convolucional



Fonte: [Ulku e Akagündüz \(2022\)](#)

A arquitetura codificador-decodificador — ou Encoder-Decoder —, é separada em dois passos: o primeiro para convergir no mapa de características — chamado de codificador — e o segundo para reverter — chamado de decodificador — as camadas de agrupamento para aumentar a dimensão da saída, usando camadas deconvolucionais e de desagrupamento, outra característica importante é a conexão entre camadas de mesmo nível, como por exemplo a arquitetura UNet que foi a primeira a implementar o padrão Codificador-Decodificador. Na Figura 24 pode-se observar que tem formato da letra U, sendo a descida a parte de codificação e subida decodificação ([ULKU; AKAGÜNDÜZ, 2022](#); [WANGENHEIM, 2021](#); [RONNEBERGER; FISCHER; BROX, 2015](#)).

Figura 24 – Arquitetura codificador-decodificador UNet



Fonte: Ronneberger, Fischer e Brox (2015)

Segmentação de instância

Outro problema dentro da área de visão computacional é a detecção de objetos, a primeira solução foi com Características de regiões com RNC — Regions with CNN features (R-CNN) — que se resume em dividir a imagem de entrada em regiões de interesse e nessas regiões aplicar uma RNC. A arquitetura que seleciona essas regiões é chamada de Rede de proposta de região — ou Region Proposal Network (RPN) — o que auxilia na detecção por caixas delimitadoras. Essa ideia inicial foi estendida para segmentação de instância criando também máscara nos objetos, como por exemplo a arquitetura Mask R-CNN ([ULKU; AKAGÜNDÜZ, 2022](#); [WANGENHEIM, 2021](#)).

A arquitetura Mask R-CNN é derivado do Fast R-CNN — aprimoramento do R-CNN aplicando conceito RoIPool para classificar — onde há uma segmentação de máscara em cada ROI — ou Região de interesse — paralela com a classificação da caixa delimitadora. A máscara é classificada com uma pequena FCN — ou rede totalmente convolucionial — em cada ROI. Além de ter uma pequena melhoria na RoIPool, pois havia um problema de alinhamento nas localizações espaciais exatas, essa camada é chamada de ROIAlign ([HE et al., 2017](#)).

Segmentação panóptica

Um problema encontrado na segmentação semântica é que objetos de mesma classe não são separados igual na segmentação de instância, logo surgiu uma ideia para criar uma solução usando as duas técnicas. Esse conceito surgiu do trabalho [Kirillov et al. \(2019\)](#) e consiste na definição geral da ideia, uma métrica — que será explicada posteriormente — unificada para classificar os resultados do modelo além de fazer a distinção entre coisas

— ou stuff — que não são contáveis, como o céu e os objetos — ou things — que são contáveis como carros, pessoas, etc. Os principais conjunto de dados para tarefa panóptica são COCO-Panoptic, Cityscapes, Mapillary Vistas, ADE20K, Indian Driving Dataset. Cada conjunto de dados possui diversas classes para o modelo aprender ([BARLA, 2022](#)).

Métricas, técnicas e resultados

Com base no contexto acima percebe-se que a segmentação panóptica é a mais completa e por efeito de estudos será utilizado o mesmo para concluir o trabalho. Como nesse nicho existem várias alternativas será abordado algumas métricas, técnicas e resultados para selecionar o modelo.

União sobre intersecção

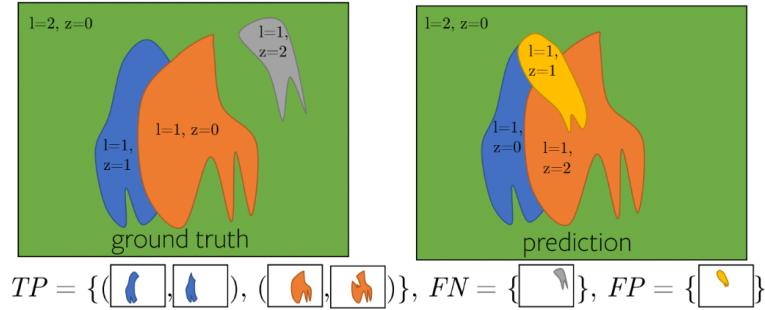
A união sobre intersecção — Intersection over Union (IoU) — ou índice de Jaccard é uma métrica muito utilizada para calcular a eficiência de modelos de segmentação, ela se baseia em encontrar uma relação entre a área das classes da imagem de saída com as classes na imagem do conjunto de dados, segue a fórmula([ULKU; AKAGÜNDÜZ, 2022](#); [WANGENHEIM, 2021](#); [KIRILLOV et al., 2019](#)):

$$IoU = \frac{\text{Área de intersecção}}{\text{Área de união}} \quad IoU(p_i, g) = \frac{p_i \cap g}{p_i \cup g} \quad (2.15)$$

Classificação de conjuntos

A classificação de conjuntos é uma técnica para criar relações entre a imagem de predição e a imagem do conjunto de dados. Ela se divide em três classificações sendo elas: Positivos Verdadeiros — ou True Positives(TP) — sendo o requisito ter uma intersecção significativa entre classes iguais, *i.e.*, $IoU > 0.5$, Falso Positivos — ou False Positives(FP) — quando um objeto não é correspondido na imagem de predição e por fim Falso Negativos — False Negatives(FN) — quando um objeto não é correspondido na imagem do conjunto de dados, pode-se observar esses conceitos de conjuntos na Figura 25 ([KIRILLOV et al., 2019](#)).

Figura 25 – Exemplo da classificação dos conjuntos usados nas métricas de segmentação



Fonte: [Kirillov \(2019\)](#)

Qualidade panóptica

A qualidade panóptica — ou Panoptic Quality (PQ) — foi definido pela primeira vez no artigo [Kirillov et al. \(2019\)](#), e se resume na fórmula:

$$PQ = \frac{\sum_{(p,g) \in TP} IoU(p, g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \quad (2.16)$$

Multiplicando a Equação (2.16) por $\frac{|TP|}{|TP|}$ tem-se:

$$PQ = \underbrace{\frac{\sum_{(p,g) \in TP} IoU(p, g)}{|TP|}}_{\text{Segmentation Quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{Recognition Quality (RQ)}} \quad (2.17)$$

Portanto pode-se concluir que PQ é apenas uma simplificação para uma fórmula que contém uma relação entre métricas de segmentação semântica e de instância.

Resultados de modelos do nicho de segmentação panóptica

Os resultados são de uma competição em aberto criada pela Cytoscapes Dataset, essa competição tem várias modalidades e esses são referentes ao nicho de segmentação panóptica utilizando a métrica PQ na classe de pessoas ([DATASET, 2023](#)).

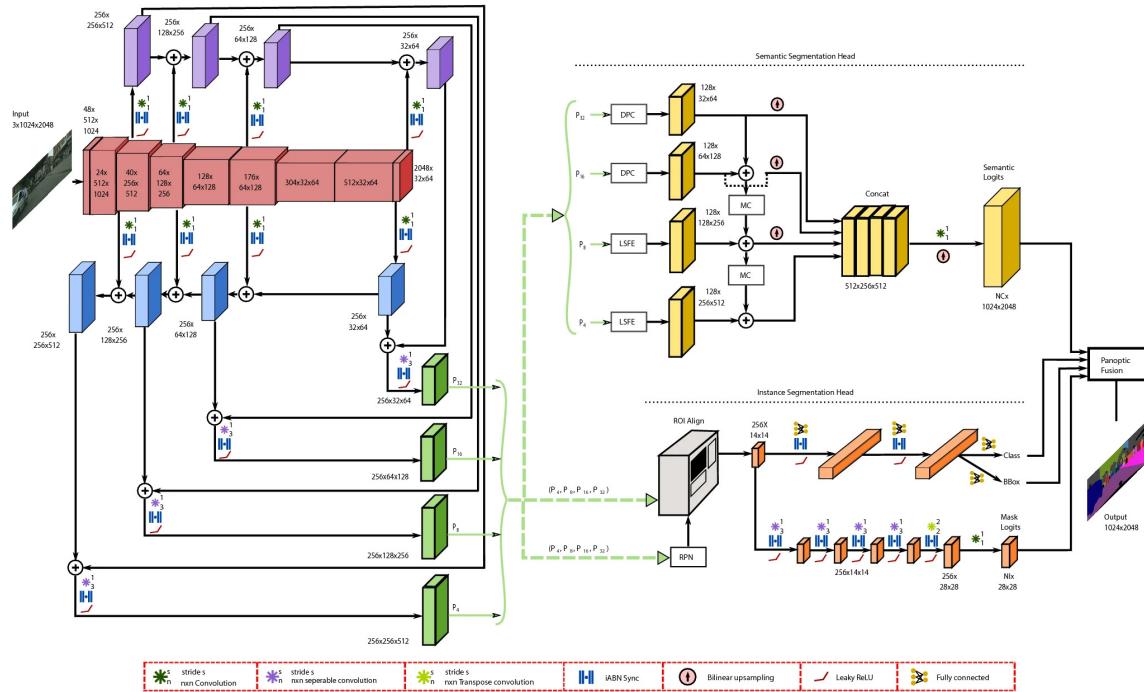
Tabela 1 – Top 15 modelos que melhor classificam pessoas com métrica P em segmentação panóptica

| Nome do modelo | PQ (%) |
|--|--------|
| EfficientPS [Mapillary Vistas] | 61,6 |
| EfficientPS [Cityscapes-fine] | 60,9 |
| Panoptic-DeepLab w/ SWideRNet [Mapillary Vistas + Pseudo-labels] | 60,6 |
| hri_panoptic | 60,6 |
| Naive-Student (iterative semi-supervised learning with Panoptic-DeepLab) | 60,2 |
| Panoptic-DeepLab w/ SWideRNet [Mapillary Vistas] | 59,8 |
| iFLYTEK-CV | 59,2 |
| Panoptic-DeepLab [Mapillary Vistas] | 58,5 |
| Panoptic-DeepLab w/ SWideRNet [Cityscapes-fine] | 58,4 |
| Seamless Scene Segmentation | 57,7 |
| Axial-DeepLab-XL [Mapillary Vistas] | 57,2 |
| Unifying Training and Inference for Panoptic Segmentation [COCO] | 56,5 |
| kMaX-DeepLab [Cityscapes-fine] | 56 |
| Axial-DeepLab-L [Mapillary Vistas] | 55,9 |
| TASCNet-enhanced | 55,2 |

2.3.6 EfficientPS

EfficientPS é uma solução para a segmentação panóptica proposta no artigo [Mohan e Valada \(2021\)](#), o trabalho apresenta uma arquitetura que se inicia com um backbone — parte para identificar características — usando uma FPN de 2 caminhos seguido de dois cabeçotes paralelos um para uma arquitetura de segmentação semântica que é autoria deles e outra de instância com modificações baseadas na topologia Mask R-CNN e finalmente a saída dos dois cabeçotes são combinadas no módulo de fusão panóptica para gerar a saída final com a imagem de segmentação panóptica, esta arquitetura é ilustrada na Figura 26.

Figura 26 – Arquitetura geral do EfficientPS



Fonte: Mohan e Valada (2021)

Backbone da rede

A espinha dorsal — ou backbone — se consiste em uma codificação combinado a uma bifurcação paralela usando FPN. O codificador é essencial para arquiteturas de segmentação e para melhorar a capacidade de representação é necessário aumentar o número de parâmetros e a complexidade, porém nesse artigo os autores chegaram numa solução balanceada nesse quesito. O codificador contém nove blocos (em vermelho), mostrado na Figura 26 e a 2º, 3º, 5º e 9º saídas — da esquerda para direita — correspondem aos fatores de redução de amostragem x4,x8,x16 e x32 respectivamente. Essas saídas vão conectar com a bifurcação paralela que são de sentidos opostos para gerar mais detecções de características, após isso será feita uma combinação entre camadas de mesma dimensão utilizando camadas de convolução separável em profundidade — divide em etapa espacial e de canal, aplicada a cada canal e cada pixel de saída respectivamente — resultando nas saídas $P_4 + P_8 + P_{16} + P_{32}$ (MOHAN; VALADA, 2021; LIMA, 2021).

Cabeçote de Segmentação Semântica

O cabeçote de segmentação semântica é autoria dos autores e é dividido em três módulos sendo eles: extrator de características em larga escala — ou Large Scale Feature Extractor (LSFE) — para capturar recursos finos em larga escala de forma eficiente, módulo DPC deve ser capaz de capturar contexto de longo alcance porém em pequena

escala e o módulo MC deve ser capaz de mitigar a incompatibilidade entre recursos de grande e pequena escala nas camadas de agregação (MOHAN; VALADA, 2021).

As quatro entradas do cabeçote $P_4 + P_8 + P_{16} + P_{32}$ são separadas, sendo $P_{16} + P_{32}$ — pequena escala — alimentam dois módulos DPC paralelos e $P_4 + P_8$ — larga escala — alimentam dois módulos LSFE paralelos (MOHAN; VALADA, 2021).

Cabeçote de segmentação de instância

Este cabeçote é derivada da arquitetura Mask R-CNN e as modificações foram três, sendo elas: trocar a convolução padrão por convolução separável em profundidade — para reduzir o número de parâmetros consumidos pela rede —, camada de normalização em lote foi substituída por iABN Sync — normalização em lotes entre cores de GPU para aumentar o desempenho — e a função ReLU por Leaky ReLU (MOHAN; VALADA, 2021; LIMA, 2021; SCHUMACHER, s. d.).

Módulo de fusão panóptica

O módulo da fusão panóptica é necessário para construir a imagem com segmentação panóptica, nessa parte é fundido os resultados dos dois cabeçotes anteriormente explicados. Esta tarefa não é simples pois é necessário criar uma lógica para obter o melhor resultado diante das sobreposições encontradas. O módulo foi criado no intuito de ser adaptativo e usar as duas entradas de forma equivalente (MOHAN; VALADA, 2021).

Resumindo o módulo aplica algumas técnicas para reduzir o número de instâncias baseando-se na métrica logist — valor numérico que pontua confiança —, aplica algumas agregações entre os resultados dos dois cabeçotes, e desenha com fundo preto as instâncias com melhor classificação de confiança, logo depois preenche com a parte de stuff — classes semânticas sem importância — da entrada semântica (MOHAN; VALADA, 2021).

2.4 Trabalhos relacionados

Esta seção destina-se a análise e discussão da metodologia e dos resultados propostos por Leite e Lima (2015), Kirillov et al. (2019).

Geração Procedural de Mapas para Jogos 2D

No trabalho Leite e Lima (2015), é apresentado uma solução simples para criar mapas de cavernas, calabouços e ilhas para jogos 2D. O algoritmo foi dividido em três partes sendo elas: geração recursiva de terrenos, validação de tamanho e correção da coesão. Os autores concluíram que não existe literatura sobre geração procedural de salas diversas e corredores distintos como o algoritmo proposto. Sugerem duas possibilidades

para trabalhos futuros sendo elas: usar algoritmos genéticos para mensurar a qualidade dos mapas gerados e promover pela seleção natural e a outra possibilidade é mesclar o algoritmo proposto com técnicas de geração de salas interligadas por corredores, de forma a possibilitar a criação de mapas com algumas salas pré-definidas inseridas em um mapa aberto contínuo.

Panoptic Segmentation

No trabalho [Kirillov et al. \(2019\)](#) é definido a ideia geral de segmentação panóptica além de definir conceitos importantes como coisas e objetos e a métrica unificada para medir o desempenho de modelos dessa área. Também é feito alguns testes comparando resultados humanos com um modelo simples proposto com eles combinando PSPNet e Mask R-CNN usando a métrica de qualidade panóptica definida por eles. Os resultados mostraram a superioridade humana na segmentação panóptica em três conjuntos de dados diferentes, sendo eles: Cityscapes, ADE20k e Vistas, as métricas usadas foram qualidade panóptica, qualidade semântica, qualidade de reconhecimento, qualidade panóptica de coisas e qualidade panóptica de objetos. O melhor resultado para a máquina em comparação com o humano foi no conjunto de dados Cityscapes avaliando a qualidade semântica, sendo 84,1 para o humano e 80,9 para máquina. O pior resultado para a máquina em relação ao humano foi no conjunto de dados ADE20k na qualidade panóptica de coisas, sendo 71,0 para os humanos e 24,5 para a máquina.

Polygonal Map Generation for Games

No artigo de [Patel \(2010\)](#) é apresentado toda uma jornada de desenvolvimento de um algoritmo de geração procedural de conteúdo, é mostrando as técnicas para gerar o mapa com o diagrama de Voronoi, gerar os rios e biomas utilizando as camadas de elevação e umidade do polígono e a aplicando isso no diagrama de Whittaker para definir o bioma do polígono. Também é apresentado uma técnica para adicionar ruídos nas arestas dos polígonos fazendo com que o mapa se torne mais orgânico e realista.

3 Desenvolvimento

3.1 Proposta

Este trabalho tem como proposta a utilização de um modelo de inteligência artificial para segmentação panóptica que irá classificar os pixels na imagem e permitir que os usuários gerem mapas a partir da seleção de um dos segmentos da imagem.

Utilizando o modelo EfficientPS é possível fazer a segmentação panóptica, segue um exemplo na Figura 27 e na Figura 28. O modelo citado está disponível no GitHub dos próprios autores [Mohan e Valada \(2021\)](#) e será treinado com a combinação de pelo menos dois conjuntos de dados citados anteriormente, no resultado apenas será identificado os pixels de classes contidas nos conjuntos de dados escolhidos, portanto é possível que em uma imagem não seja identificado nada:

Figura 27 – Imagem de entrada para rede neural.



Fonte: [Kirillov et al. \(2019\)](#)

Após a segmentação da imagem o usuário poderá selecionar qual parte da imagem será utilizada para gerar a ilha.

Feito a seleção será gerado um diagrama de Voronoi que funcionará como um filtro em cima dessa imagem, assim gerando a ilha e os biomas.

3.2 Cronograma

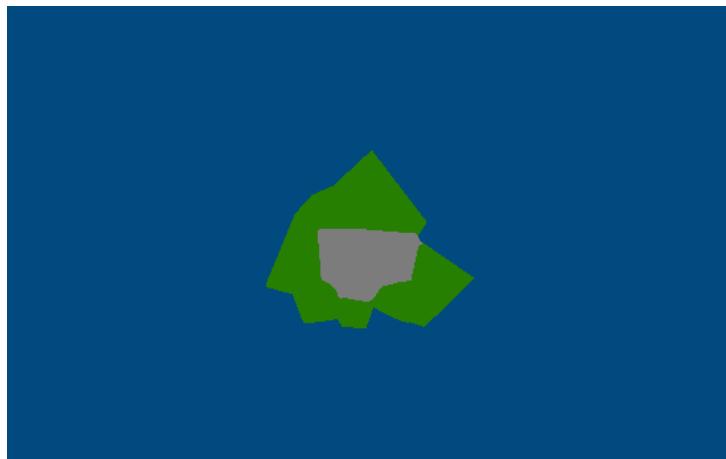
O processo de desenvolvimento será separado em 3 tópicos principais, inteligência artificial, diagrama de Voronoi e interface de usuário. O desenvolvimento de cada tópico do software será feito em paralelo, pois os tópicos não possuem acoplamento.

Figura 28 – Imagem saída de um modelo de segmentação panóptica de segmentação panóptica.



Fonte: Kirillov et al. (2019)

Figura 29 – Ilha gerada a partir da segmentação panóptica e aplicando um filtro com o diagrama de Voronoi, azul representa oceano, verde floresta, cinza montanhas.



Fonte: Criação própria

inteligência Artificial

Será necessário decidir quais conjuntos de dados utilizar, o modelo está pronto e disponível no GitHub ([MOHAN; VALADA, 2021](#)) portanto será necessário treinar o modelo e avaliar-lo com base na métrica PQ Equação (2.16).

As especificações do modelo proposto são: Linux, Python 3.7, PyTorch 1.7, CUDA 10.2, GCC 7 ou 8 além dos pacotes inseridos no arquivo requirements.txt.

O tempo estimado para o desenvolvimento é de 1 a 2 meses, a maior parte será para treinar e validar o resultado.

Diagrama de Voronoi

Para o desenvolver código do diagrama de Voronoi será preciso primeiro gerar os pontos e desses pontos as áreas, fazer o algoritmo entender se a área tocou no segmento de imagem, caso tenha tocado armazenar para um processamento posterior que irá especificar qual bioma aquela áreas será, para fazer os teste será necessário uma imagem com um polígono.

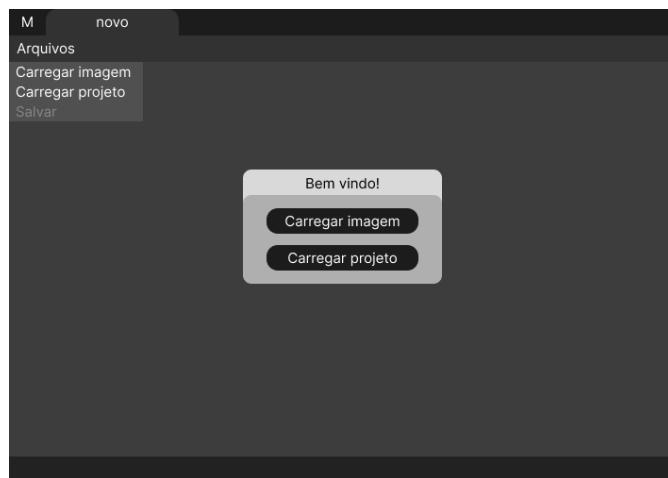
O tempo estimado para o desenvolvimento é de 1 mes.

Interface de Usuário

A interface de usuário terá 5 telas principais, inicio, processamento da segmentação, seleção, processamento de seleção, resultado.

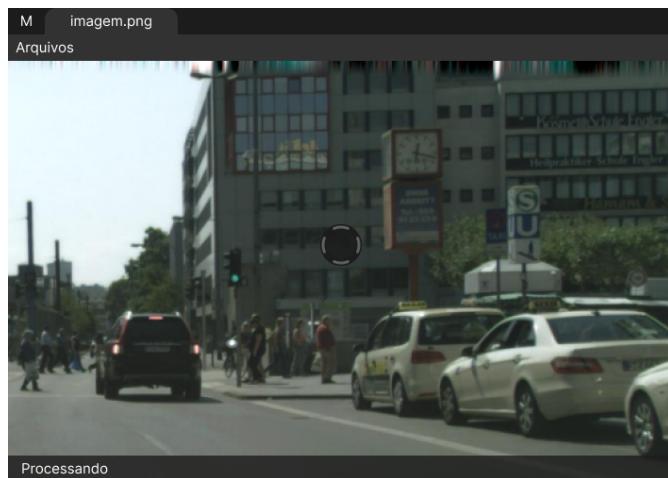
As telas terão o seguinte fluxo:

Figura 30 – Tela de inicio, botões de carregar imagem e carregar projeto, menu de contexto arquivos com 3 botões, carregar imagem, carregar projeto e salvar.



Fonte: Criação própria

Figura 31 – Tela de processamento da segmentação



Fonte: Criação própria

Figura 32 – Tela de seleção de segmentação da imagem.



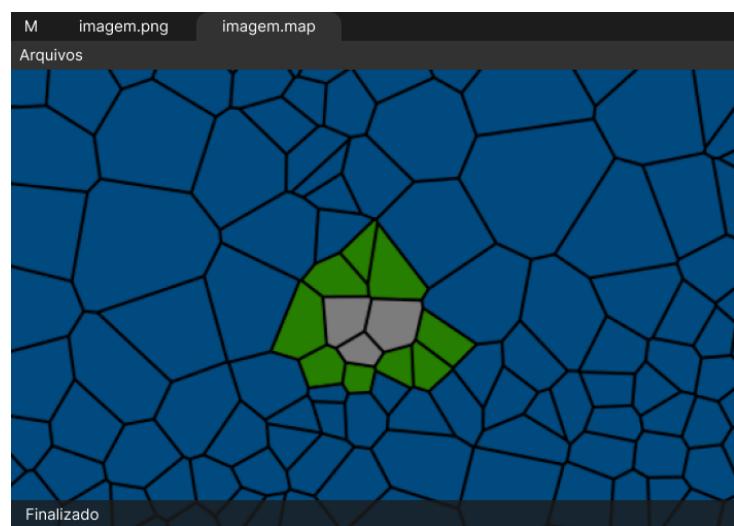
Fonte: Criação própria

Figura 33 – Tela de processamento para geração do mapa com a seleção do segmento.



Fonte: Criação própria

Figura 34 – Tela de resultado com o mapa gerado após processamento.



Fonte: Criação própria

Após isso a interface permitirá o usuário salvar o projeto bem como exportar o resultado.

O tempo de desenvolvimento será em torno de 1 mês.

Referências

- ALZUBAIDI, L. et al. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, v. 8, n. 1, p. 53, Mar 2021. ISSN 2196-1115. Disponível em: <<https://doi.org/10.1186/s40537-021-00444-8>>. Citado 5 vezes nas páginas 35, 37, 39, 40 e 41.
- AMINI, A. *MIT 6.S191: Convolutional Neural Networks*. 2023. Disponível em: <https://www.youtube.com/watch?v=NmLK_WQBxB4&t=1337s>. Citado na página 29.
- ARAUJO, E. A. e. C. M. Wendell Oliveira de. Geração procedural de conteúdo aplicada a jogos digitais educacionais. 2018. Citado na página 20.
- BABICH, N. *What Is Computer Vision? How Does It Work?* 2020. <<https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/>>. Acesso em: 18-05-2023. Citado 2 vezes nas páginas 11 e 29.
- BAIRD, S. *How Much Grand Theft Auto 5 Cost To Make*. 2021. <<https://screenrant.com/grand-theft-auto-5-how-much-cost-make/>>. Acessado: 2023-03-20. Citado na página 19.
- BARLA, N. *Panoptic Segmentation: Definition, Datasets & Tutorial 2023*. 2022. V7 Labs. Disponível em: <<https://www.v7labs.com/blog/panoptic-segmentation-guide>>. Citado na página 44.
- BIOMAS: características e tipos. [S.l.]: Maestrovirtuale.com, s. d. <<https://maestrovirtuale.com/biomass-caracteristicas-e-tipos/>>. Acessado: 2023-06-05. Citado na página 24.
- BRILLIANT.ORG. *Backpropagation*. [S.l.]: Brilliant.org, 2023. <<https://brilliant.org/wiki/backpropagation/>>. Acessado: 2023-06-01. Citado na página 36.
- BROWN, S. *Machine learning, explained*. 2021. <<https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>>. Acessado: 2023-05-11. Citado na página 31.
- CLEMENT, J. *Number of games released on Steam worldwide from 2004 to 2022*. 2023. <<https://www.statista.com/statistics/552623/number-games-released-steam/>>. Acessado: 2023-03-14. Citado na página 19.
- DATASET, C. *Panoptic Semantic Labeling Task - PQ on class-level*. 2023. <<https://www.cityscapes-dataset.com/benchmarks>>. Citado na página 45.
- DORMANS, J. Adventures in level design: Generating missions and spaces for action adventure games. Weesperzijde 190, 1097DZ Amsterdam, The Netherlands, 2010. Citado na página 23.
- EDUCATIVE. *Overfitting and underfitting*. [S.l.]: Educative, 2022. <<https://www.educative.io/>>. Acessado: 2023-06-01. Citado na página 37.

FANDOM. *Mapa del tesoro - Sea of Thieves Wiki*. 2021. <https://seaofthieves.fandom.com/es/wiki/Mapa_del_tesoro>. Acessado em 4 de junho de 2023. Citado na página 19.

FERNEDA, E. Redes neurais e sua aplicação em sistemas de recuperação de informação. *Ciência da Informação*, SciELO Brasil, v. 35, n. 1, p. 41–53, 2006. Disponível em: <<https://www.scielo.br/j/ci/a/SQ9myjZWlxnyXfstXMgCdcH/>>. Citado na página 33.

FOFFANO, G. Sea of thieves: Branle-bas de combat bande de forbans. Le Café du Geek, 2020. Disponível em: <<https://lecafedugeek.fr/sea-of-thieves-branle-bas-de-combat-bande-de-forban/>>. Citado na página 19.

GHARAT, S. *What, Why and Which?? Activation Functions*. 2019. Medium. Acessado: 2023-05-24. Disponível em: <<https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441>>. Citado 3 vezes nas páginas 33, 34 e 35.

GIANNOTTI, R. *Pesquisa Game Brasil 2022 mostra que 74,5 dos brasileiros jogam games regularmente*. 2022. <<https://www.adrenaline.com.br/games/pesquisa-game-brasil-2022-mostra-que-745-dos-brasileiros-jogam-games-regularmente/>>. Acessado: 2023-03-12. Citado na página 19.

GMBH, H. *Instance Segmentation*. 2023. <<https://hasty.ai/docs/mp-wiki/model-families/instance-segmentor>>. Citado 2 vezes nas páginas 11 e 28.

GOMES, D. D. S. Inteligência artificial: Conceitos e aplicações. *Olhar Científico - Faculdades Associadas de Ariquemes*, v. 1, n. 2, p. 234–246, 2010. Disponível em: <https://www.professores.uff.br/screspo/wp-content/uploads/sites/127/2017/09/ia_intro.pdf>. Citado na página 30.

HAUßECKER BERND JÄHNE, B. J. H. *Handbook of computer vision and applications*. [S.l.]: ACADEMIC PRESS, 1999. ISBN ISBN 0-12-379770-5 (set). — ISBN 0-12-379771-3 (v. 1). Citado na página 28.

HAYKIN, S. S. *Neural Networks: A Comprehensive Foundation*. [S.l.]: Prentice Hall, 1999. Citado 3 vezes nas páginas 32, 33 e 38.

HE, K. et al. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. Disponível em: <<http://arxiv.org/abs/1703.06870>>. Citado na página 44.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. *Electronic Markets*, v. 31, n. 3, p. 685–695, Sep 2021. ISSN 1422-8890. Disponível em: <<https://doi.org/10.1007/s12525-021-00475-2>>. Citado 2 vezes nas páginas 31 e 32.

KIRILLOV, A. *Panoptic Segmentation: Task and Approaches*. [S.l.]: CVPR, 2019. <<http://feichtenhofer.github.io/cvpr2019-recognition-tutorial/>>. Acessado em 2023-06-05. Citado na página 45.

KIRILLOV, A. et al. *Panoptic Segmentation*. 2019. Citado 8 vezes nas páginas 20, 42, 44, 45, 48, 49, 51 e 52.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Citado na página 37.

- LEITE, G.; LIMA, E. Soares de. Geração procedural de mapas para jogos 2d. In: . [s.n.], 2015. Disponível em: <https://www.researchgate.net/publication/297704013_Geracao_Procedural_de_Mapas_para_Jogos_2D>. Citado na página 48.
- LIMA, A. Redes neurais convolucionais separáveis em profundidade. *Acervo Lima*, 11 2021. Disponível em: <<https://acervolima.com/redes-neurais-convolucionais-separaveis-em-profundidade/>>. Citado 2 vezes nas páginas 47 e 48.
- LISBOA, A. O que é um jogo procedural? *Canaltech*, 2022. Disponível em: <<https://canaltech.com.br/games/o-que-e-um-jogo-procedural-228162/>>. Citado na página 20.
- MALAR, J. P. Nvidia usa inteligência artificial em conversas com personagens de jogos. *Exame*, 2023. Disponível em: <<https://exame.com/future-of-money/nvidia-usa-inteligencia-artificial-conversas-personagens-jogos/>>. Citado na página 20.
- MARR, B. *7 Amazing Examples Of Computer And Machine Vision In Practice*. 2019. <<https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/?sh=4ee6506b1018>>. Acesso em: 18-05-2023. Citado na página 29.
- MARTI, L.; BARROS, T. Aprendizado profundo: Fundamentos, histórico e aplicações. In: SBC. *Anais do XIV Simpósio Brasileiro de Sistemas Colaborativos*. [S.l.], 2017. Citado 4 vezes nas páginas 32, 33, 37 e 38.
- MENDES, M. *Ecologia 02 - Os grandes biomas terrestres*. 2019. <<http://maxaug.blogspot.com/2019/07/ecologia-02-os-grandes-biomas-terrestres.html>>. Acessado: 2023-06-05. Citado na página 27.
- MOHAN, R.; VALADA, A. Efficientps: Efficient panoptic segmentation. *International Journal of Computer Vision (IJCV)*, 2021. Disponível em: <<https://github.com/DeepSceneSeg/EfficientPS>>. Citado 5 vezes nas páginas 46, 47, 48, 51 e 52.
- MONARD, M. C.; BARANAUKAS, J. A. *Aplicações de Inteligência Artificial: Uma Visão Geral*. 2000. <<https://dcm.ffclrp.usp.br/~augusto/publications/2000-laptec.pdf>>. Citado na página 30.
- NVIDIA. *NVIDIA Omniverse ACE*. 2021. <<https://developer.nvidia.com/omniverse/ace>>. Acessado em 4 de junho de 2023. Citado na página 20.
- O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015. Disponível em: <<http://arxiv.org/abs/1511.08458>>. Citado na página 38.
- PATEL, A. *Polygonal Map Generation for Games*. 2010. <<http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/>>. Acessado: 2023-06-05. Citado 5 vezes nas páginas 24, 25, 26, 27 e 49.
- POLÍGONOS de Thiessen ou Voronoi- Como gerar e para que utilizá-los. 2018. <<https://forest-gis.com/2018/02/poligonos-de-thiessen-como-gerar-e-para-que-utiliza-los.html>>. Acessado: 2023-03-26. Citado na página 23.

RIZZO, I. V.; CANATO, R. L. C. Inteligência artificial: funções de ativação. *Prospectus* (ISSN: 2674-8576), v. 2, n. 2, 2020. Disponível em: <<https://www.prospectus.fatecitapira.edu.br/index.php/pst/article/view/37>>. Citado na página 34.

RODRIGUES, D. S. M. *Diagrama de Voronoi : uma abordagem sobre jogos*. Dissertação (Mestrado) — Universidade Estadual de Maringá, Maringá, 2019. Disponível em: <<http://repositorio.uem.br:8080/jspui/handle/1/6748>>. Citado 2 vezes nas páginas 23 e 24.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. Disponível em: <<http://arxiv.org/abs/1505.04597>>. Citado na página 43.

SANTANA, W. *Games vão movimentar R\$ 1 tri em 2023 e empresas estão de olho nisso*. 2022. <<https://www.infomoney.com.br/negocios/games-movimentar-r-1-tri-em-2023-empresas-de-olho/>>. Acessado: 2023-03-15. Citado na página 19.

SANTOS, P. R. S. dos. *Diagrama de voronoi: Uma Exploracão nas Distâncias Euclidiana e do Táxi*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná - UTFPR, 2016. Citado na página 23.

SARKER, I. H. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, v. 2, n. 6, p. 420, Aug 2021. ISSN 2661-8907. Disponível em: <<https://doi.org/10.1007/s42979-021-00815-1>>. Citado 4 vezes nas páginas 30, 38, 39 e 40.

SCHUMACHER, D. *Synchronized Batch Normalization*. s. d. <<https://serp.ai/synchronized-batch-normalization>>. Acessado em 2023-06-09. Citado na página 48.

SHAKER, N.; TOGELIUS, J.; NELSON, M. J. *Procedural Content Generation in Games*. [S.l.]: Springer, 2016. Citado na página 20.

SILVA, J. A. S. d.; MAIRINK, C. H. P. Inteligência artificial. *LIBERTAS: Revista de Ciências Sociais Aplicadas*, v. 9, n. 2, p. 64–85, dez. 2019. Disponível em: <<https://famigvirtual.com.br/famig-libertas/index.php/libertas/article/view/247>>. Citado na página 30.

SIRCAR, A. et al. Application of machine learning and artificial intelligence in oil and gas industry. *Petroleum Research*, v. 6, n. 4, p. 379–391, 2021. ISSN 2096-2495. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2096249521000429>>. Citado na página 31.

SZELISKI, R. *Computer Vision: Algorithms and Applications*. [S.l.]: Springer, 2022. Citado 2 vezes nas páginas 11 e 28.

TAYE, M. M. Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. *Computation*, v. 11, n. 3, 2023. ISSN 2079-3197. Disponível em: <<https://www.mdpi.com/2079-3197/11/3/52>>. Citado 3 vezes nas páginas 37, 39 e 41.

THOMAZTHZ. *Diagrama de Voronoi completo. União do diagrama esquerdo com o diagrama direito*. 2014. Online. <https://pt.m.wikipedia.org/wiki/Ficheiro:Diagrama_de_Voronoi.png>. Citado na página 24.

- ULKU, I.; AKAGÜNDÜZ, E. A survey on deep learning-based architectures for semantic segmentation on 2d images. *Applied Artificial Intelligence*, Taylor & Francis, v. 36, n. 1, p. 2032924, 2022. Disponível em: <<https://doi.org/10.1080/08839514.2022.2032924>>. Citado 4 vezes nas páginas 20, 42, 43 e 44.
- WANGENHEIM, A. von. *Segmentação Semântica*. [S.l.]: Lapix, 2021. <<https://lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/deep-learningsegmentacao-semanatica/>>. Acessado em 2023-06-05. Citado 3 vezes nas páginas 42, 43 e 44.
- WHITTAKER Diagram. [S.l.]: Procedural Content Generation Wiki, 2018. <<http://pcg.wikidot.com/pcg-algorithm:whittaker-diagram>>. Acessado: 2023-06-05. Citado na página 26.
- W!N, F. T. *Video game maps*. [S.l.]: For The W!n, 2023. <<https://ftw.usatoday.com/lists/video-game-maps>>. Acessado em 4 de junho de 2023. Citado na página 19.
- WOSCHANK, M.; RAUCH, E.; ZSIFKOVITS, H. A review of further directions for artificial intelligence, machine learning, and deep learning in smart logistics. *Sustainability*, v. 12, n. 9, 2020. ISSN 2071-1050. Disponível em: <<https://www.mdpi.com/2071-1050/12/9/3760>>. Citado na página 31.
- XU, B. et al. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015. Disponível em: <<http://arxiv.org/abs/1505.00853>>. Citado na página 34.
- YANNAKAKIS, G. N.; TOGELIUS, J. *Artificial Intelligence and Games*. [S.l.]: Springer, 2018. <<http://gameaibook.org>>. Citado na página 23.