

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

Modelo canônico de trabalho monográfico
acadêmico em conformidade com as normas
ABNT apresentado à comunidade de usuários
LAT_EX.

Senac: Serviço Nacional de Aprendizagem Comercial
Bacharelado em ciência da computação

Orientador: Lauro César Araujo
Coorientador: Equipe abnT_EX2

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

Modelo canônico de trabalho monográfico
acadêmico em conformidade com as normas
ABNT apresentado à comunidade de usuários
`LATEX`.

Lauro César Araujo
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.*

Agradecimentos

Os agradecimentos principais são direcionados à Gerald Weber, Miguel Frasson, Leslie H. Watter, Bruno Parente Lima, Flávio de Vasconcellos Corrêa, Otavio Real Salvador, Renato Machnievscz¹ e todos aqueles que contribuíram para que a produção de trabalhos acadêmicos conforme as normas ABNT com L^AT_EX fosse possível.

Agradecimentos especiais são direcionados ao Centro de Pesquisa em Arquitetura da Informação² da Universidade de Brasília (CPAI), ao grupo de usuários *latex-br*³ e aos novos voluntários do grupo *abnTEX2*⁴ que contribuíram e que ainda contribuirão para a evolução do abnTEX2.

¹ Os nomes dos integrantes do primeiro projeto abnTEX foram extraídos de <<http://codigolivre.org.br/projects/abntex/>>

² <<http://www.cpai.unb.br/>>

³ <<http://groups.google.com/group/latex-br>>

⁴ <<http://groups.google.com/group/abntex2>> e <<http://abntex2.googlecode.com/>>

*“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)*

Sumário

1	INTRODUÇÃO	13
1.1	Objetivos	14
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	Visão computacional	17
2.2	Inteligência Artificial	18
2.2.1	Aprendizado de Máquina	20
2.2.2	Rede neural artificial	21
2.2.3	Aprendizado profundo	26
2.2.4	Redes neurais convolucionais	27
2.2.5	Segmentação	30
2.2.6	EfficientPS	35
2.3	Geração procedural	37
2.3.1	Diagrama de Voronoi	37
2.4	Trabalhos relacionados	38
3	DESENVOLVIMENTO	41
3.1	Proposta	41
3.2	Cronograma	42
	Conclusão	47
	REFERÊNCIAS	49

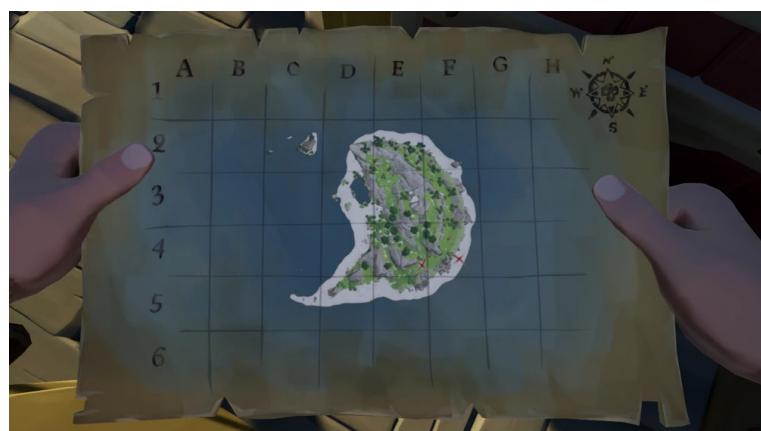
1 Introdução

A indústria de jogos digitais cresce cada vez mais a cada dia, de acordo com a consultora 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 publicados 10.963 novos títulos em 2022 ([CLEMENT, 2023](#)).

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](#)).

Ainda, 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. 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 criar cenários pode ser um desafio especialmente levando em consideração o orçamento disponível ([W!N, 2023; FOFFANO, 2020](#)).

Figura 1 – Mapa de tesouro do jogo Sea of Thieves



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

Uma abordagem eficiente para resolver o problema é a geração procedural de mapas,

que consiste em criar mapas usando algoritmos, tornando o jogo mais dinâmico e menos repetitivo. No entanto, a criação de cenários visualmente atraentes e diversos usando esse método ainda é um desafio ([LEITE; LIMA, 2015](#)).

Contextualizando, a área de Geometria Computacional é um ramo da ciência da computação que estuda algoritmos e estruturas de dados para resolução computacional de problemas geométricos e o diagrama de Voronoi é um dos tópicos mais discutidos dessa área ([RODRIGUES, 2019b](#)). O diagrama de Voronoi é gerado a partir das distâncias euclidianas entre os vizinhos mais próximos de um conjunto de pontos do plano ([SANTOS, 2016](#)). Esse diagrama possui uma gama de utilizações e dentre elas pode ser utilizado para resolver alguns problemas relacionados à jogos como por exemplo marcar pontos no mapa e desses pontos criar regiões, a partir dessas regiões criar biomas para serem usados no algoritmo de geração procedural de conteúdo para criar mapas.

De acordo com [Lisboa \(2022\)](#) é muito comum em jogos usar técnicas procedurais 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](#)).

Ao explorar a relação entre IA e jogos no contexto da geração procedural de mapas, este trabalho contribuirá para a compreensão e o avanço dessa importante área de pesquisa, oferecendo experiências mais ricas e variadas aos jogadores. Dito isso, nosso projeto tem a ideia de fornecer recursos baseados em matemática aplicada dentro de ciência da computação que proporcione uma funcionalidade de escolher o contorno do mapa no qual irá jogar através de imagens. Abordaremos a arquitetura de redes neurais convolucionais, que é muito utilizada para trabalhar com imagens. Mais especificamente, abordaremos uma arquitetura derivada da arquitetura mencionada anteriormente, específica para segmentação de imagens, o que possibilita classificar contornos em imagens.

1.1 Objetivos

O objetivo geral deste trabalho é segmentar imagens utilizando uma arquitetura de rede neural convolucional e a partir do objeto selecionado gerar um mapa com uma aplicação procedural a partir do resultado do diagrama de Voronoi. Ademais visto especificamente temos como objetivos:

- Encontrar um conjunto de dados para treinar a inteligência artificial que irá identificar contornos em imagens
- Treinar uma inteligência artificial para identificar contornos em imagens
- Testar algoritmos de gerar ruídos para criar o mapa
- Aplicar um algoritmo para reconhecer a imagem com o contorno e gerar como saída a imagem do mapa gerado

2 Fundamentação teórica

Este capítulo tem objetivo de apresentar conceitos necessários para entendimento do trabalho.

2.1 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 2 e Figura 3, evidencia-se a notável capacidade de um computador em distinguir, classificar e até mesmo compreender os elementos presentes em uma fotografia.

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

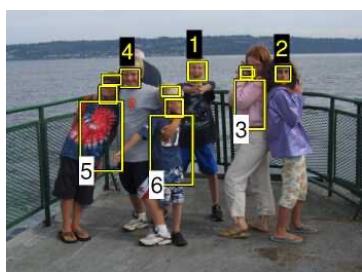


Figura 2 – Algoritmos de detecção facial e de roupas/cabelos por cor localizam e reconhecem pessoas nesta imagem (Szeliski, 2022)

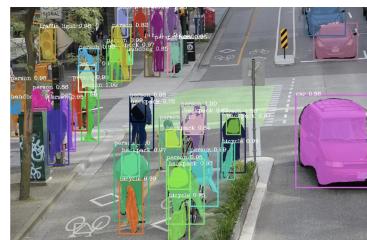


Figura 3 – Segmentação de instâncias de objetos pode delinear cada pessoa e objeto em uma cena complexa. (GMBH, 2023)

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](#))

Como a visão computacional funciona?

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 4 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 2.

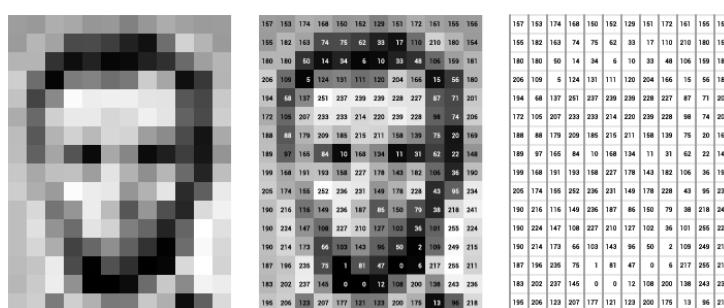


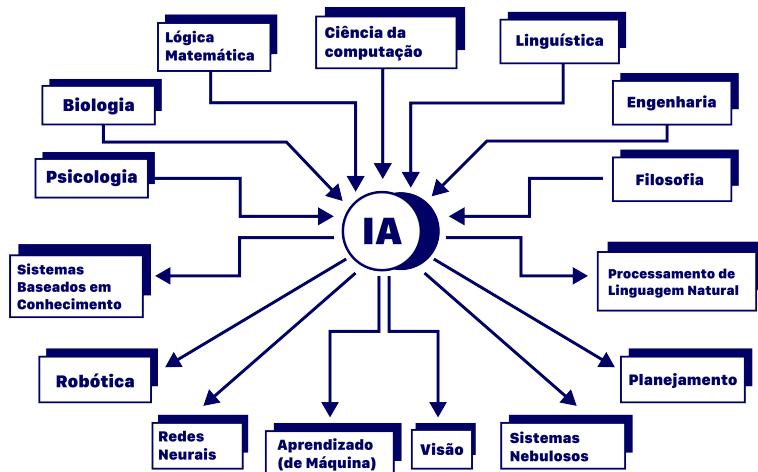
Figura 4 – Diagrama de dados de pixels. À esquerda, nossa imagem de Lincoln; no centro, os pixels rotulados com números de 0 a 255, representando sua luminosidade; e à direita, apenas esses números ([BABICH, 2020](#)).

2.2 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 conhecemos. 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).

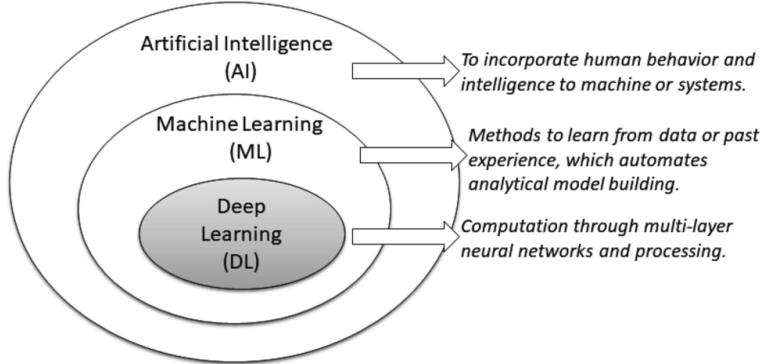
Figura 5 – Diagrama de aprendizado de máquina



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 6.

Figura 6 – Diagrama de Venn sobre relação entre os tópicos de inteligência artificial



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

2.2.1 Aprendizado de Máquina

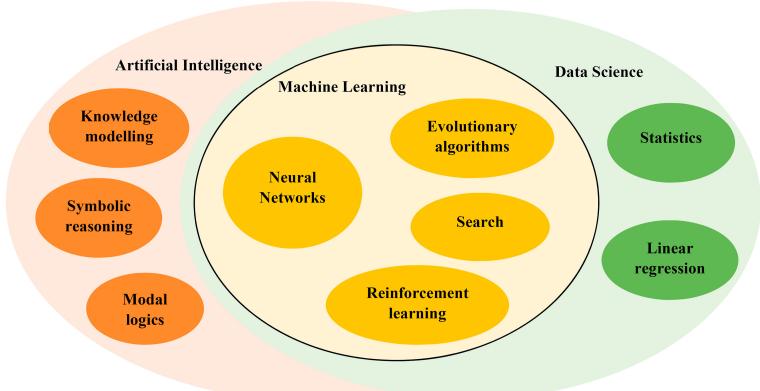
Segundo [Woschank, Rauch e Zsifkovits \(2020\)](#), aprendizado de máquina é uma subcategoria de inteligência artificial que se refere a detecção de padrões importantes de uma base de dados. As ferramentas utilizadas aumentam a eficiência dos algoritmos para lidar com bases de dados grandes.

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 7.

Figura 7 – Diagrama de aprendizado de máquina

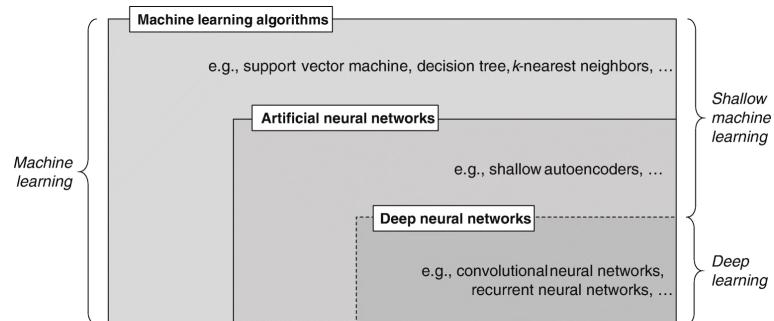


Fonte: [Sircar et al. \(2021\)](#)

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 8.

Figura 8 – Diagrama de Venn sobre tópicos de aprendizado de máquina



Fonte: [Janiesch, Zschech e Heinrich \(2021\)](#)

2.2.2 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](#)).

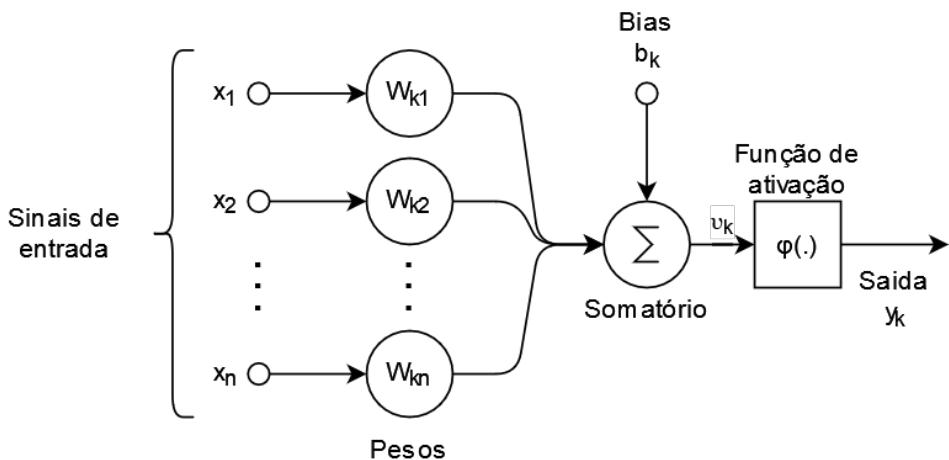


Figura 9 – Modelo de um neurônio não-linear ([HAYKIN, 1999](#)).

Observando a figura 9 vemos 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](#)):

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

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 é posta em uma função não-linear nomeada de função de ativação $\varphi(.)$, 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](#)):

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

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 ultima 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](#)).

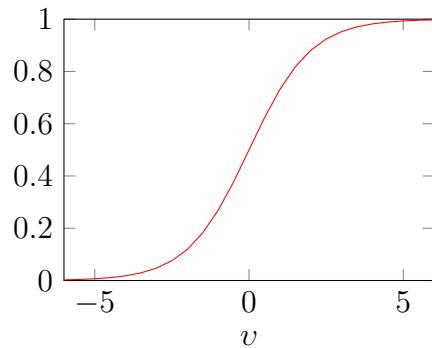
Função de ativação

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

1. 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 10 – Gráfico da função *Sigmoid*.

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



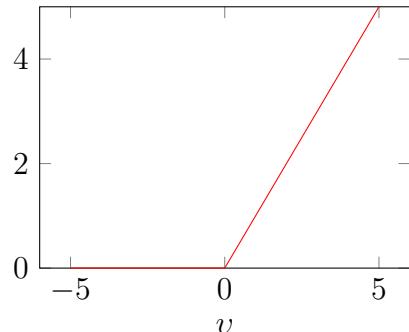
Fonte: Criação própria

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

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

Figura 11 – Gráfico da função *ReLU*.

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



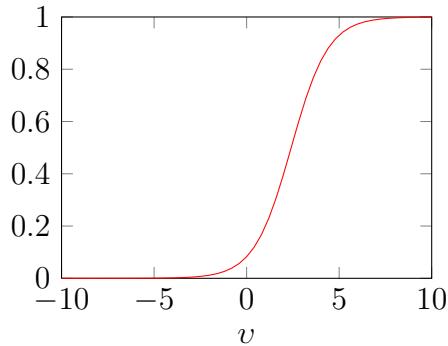
Fonte: Criação própria

A função *ReLU* é computacionalmente eficiente e converge rapidamente, porém quando a entrada da função se aproxima de zero não a rede neural não consegue executar o back-propagation, sendo assim não há aprendizado. citegharat2019what.

3. 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 12 – Gráfico da função *Softmax*.

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



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 final têm várias classes. Antes dessa função de perda é necessário usar a função de ativação softmax descrita na Figura 12 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.

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

Euclidiana ou erro quadrático médio

Muito utilizada para problemas de regressão.

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

Hinge

Muito utilizado para classificação binária.

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

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.

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

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 $\frac{\partial E(X, \theta)}{\partial w_{ij}^k}$.

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

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

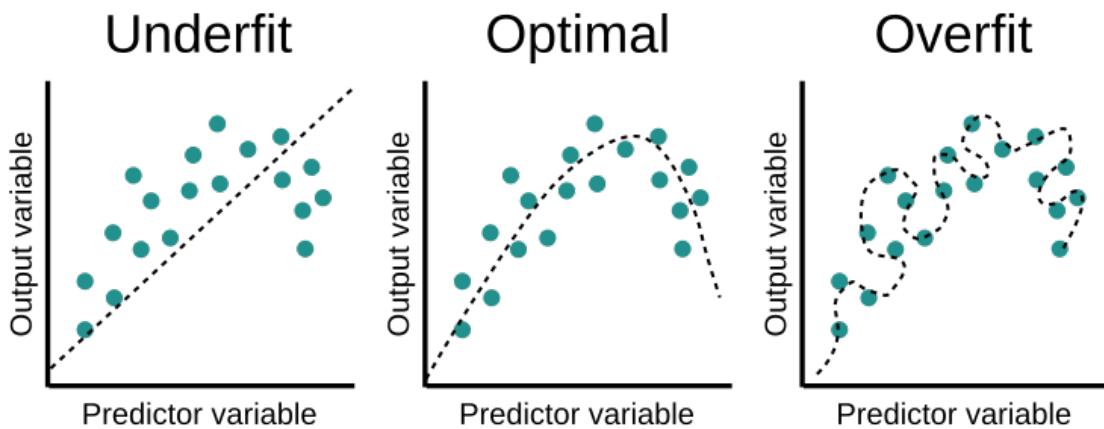
$$\Delta w_{ij}^k = -\alpha \frac{\partial E(X, \theta)}{\partial w_{ij}^k}$$

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 convolucionais 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 13 – Gráficos mostrando subajuste, balanceado e sobreajuste respectivamente



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

2.2.3 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](#)).

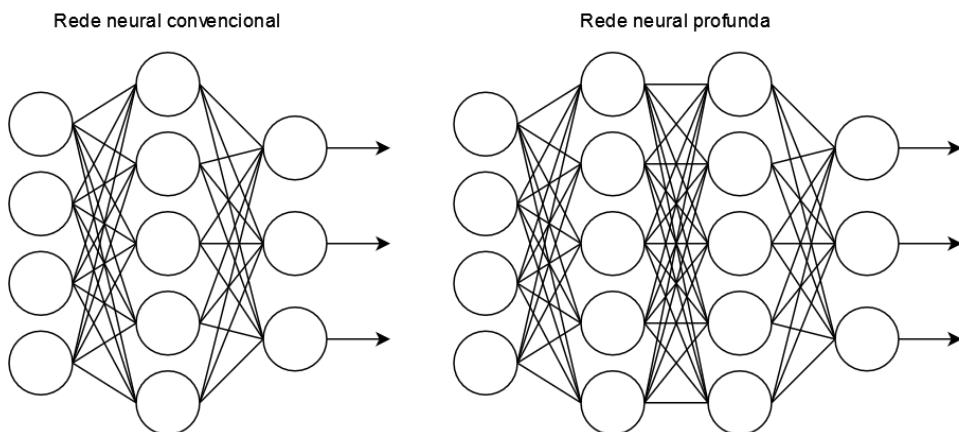
Segundo [LeCun, Bengio e Hinton \(2015\)](#), o aprendizado profundo são métodos de representação de aprendizado com vários níveis, obtidos por meio da decomposição de módulos simples e lineares, que transformam a representação de um nível em uma representação mais alta e abstrata. Por exemplo a representação 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 recombinadas 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 14 – Comparação de uma rede neural convencional com uma rede neural profunda.



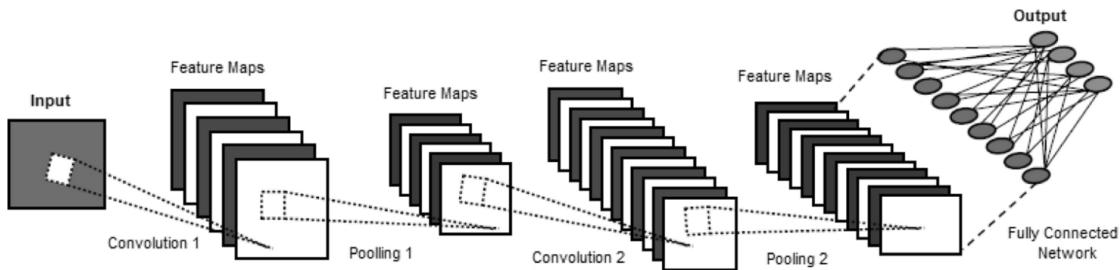
Fonte: Criação própria

2.2.4 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 15 – 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](#)).

$$\mathbf{I}x - \mathbf{F}x + 1 = \mathbf{S}x$$

$$\mathbf{I}y - \mathbf{F}y + 1 = \mathbf{S}y$$

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 $\begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix}$	$=$ $\begin{bmatrix} \boxed{1} & - & - \end{bmatrix}$
$\begin{bmatrix} 0 & \boxed{2} & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$	\otimes $\begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix}$	$=$ $\begin{bmatrix} 1 & \boxed{1} & - \end{bmatrix}$
$\begin{bmatrix} 0 & 2 & \boxed{1} & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$	\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 do 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 aonde 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 final 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 final 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 final 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:

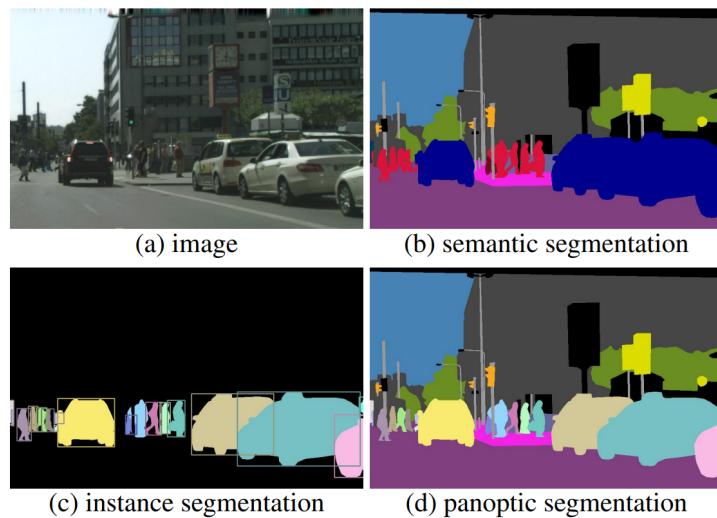
- Dropout: Muito utilizada para evitar sobreajuste pois está técnica irá desligar um neurônio aleatoriamente colocando a saída dele como zero no processo de treinamento e portanto fará o modelo a aprender a identificar características diferentes em outros neurônios possibilitando a generalização do modelo.
- 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

2.2.5 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 16 (ULKU; AKAGÜNDÜZ, 2022; WANGENHEIM, 2021).

Figura 16 – Tipos de segmentação em redes neurais convolucionais



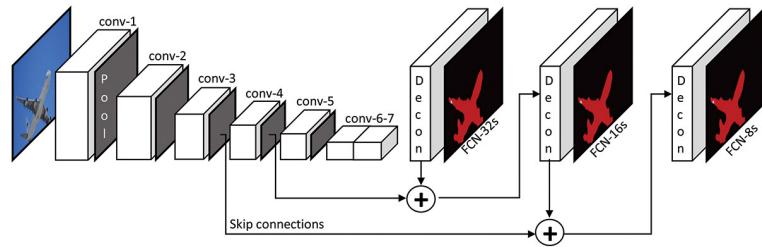
Fonte: Kirillov et al. (2019)

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 17, a arquitetura de salto evoluiu para arquitetura codificador-decodificador.

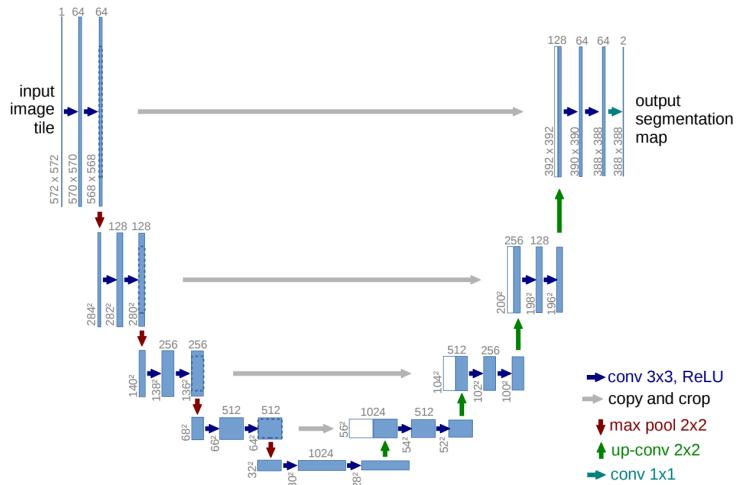
Figura 17 – Exemplo de arquitetura de rede totalmente convolucional



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 18 podemos 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 18 – 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 extendida 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 convolucional — 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.

Modelo escolhido

Com base no contexto acima percebemos que a segmentação panóptica é a mais completa e por efeito de estudos utilizaremos o mesmo para concluir o trabalho. Como nesse nicho existem várias alternativas abordaremos 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

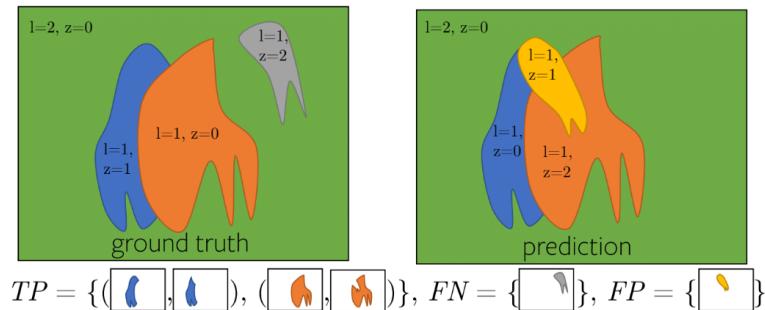
classes na imagem do conjunto de dados, segue a fórmula ([ULKU; AKAGÜNDÜZ, 2022](#); [WANGENHEIM, 2021](#); [KIRILLOV et al., 2019](#)):

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

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, podemos observar esses conceitos de conjuntos na Figura 19 ([KIRILLOV et al., 2019](#)).

Figura 19 – 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 formula:

$$PQ = \frac{\sum_{(p,g) \in TP} IoU(p, g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \quad (2.1)$$

Multiplicando a Equação (2.1) por $\frac{|TP|}{|TP|}$ temos:

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

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

Resultados

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 PG na classe de pessoas ([DATASET, 2023](#)).

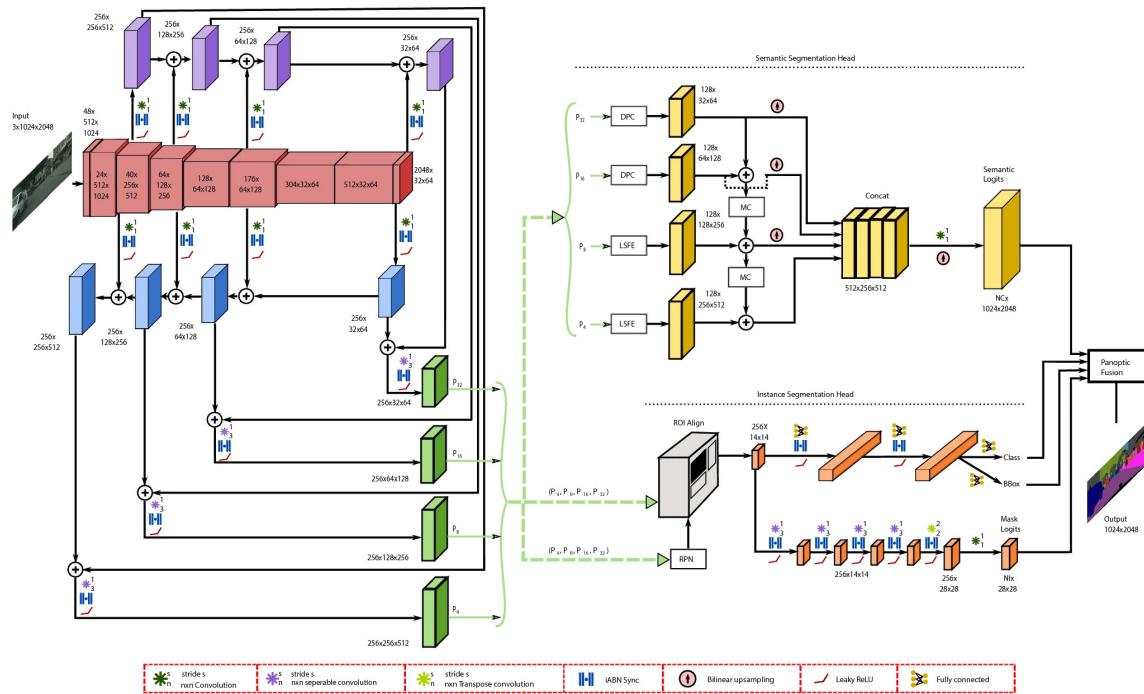
Tabela 1 – Top 15 modelos que melhor classificam pessoas

Nome do modelo	Precisão (%)
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

2.2.6 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 20.

Figura 20 – 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 20 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 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,).

Módulo de fusão panóptica

O modulo 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 da 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).

2.3 Geração procedural

2.3.1 Diagrama de Voronoi

Segundo Rodrigues (2019a) 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 distancias 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).

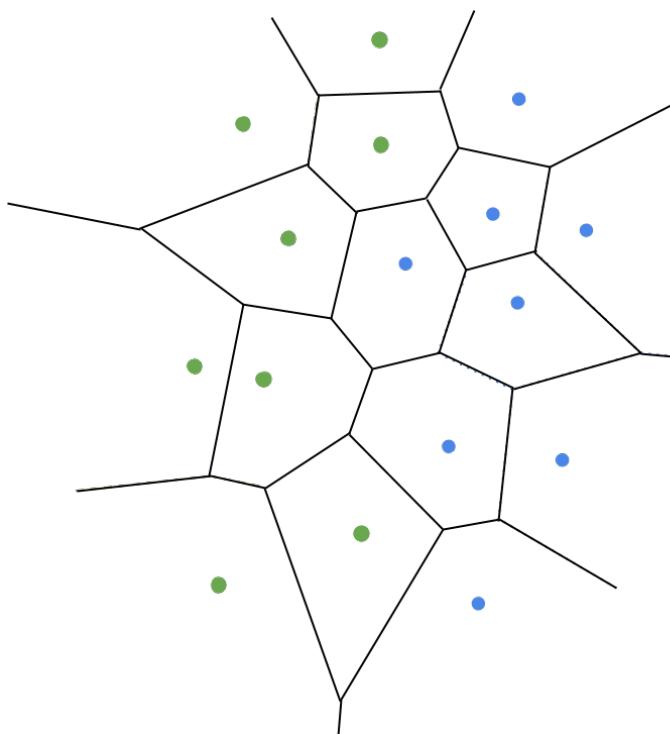
Seja um conjunto de índices $I_n = \{1, 2, 3, \dots, n\}$ e $A = \{p_1, p_2, \dots, p_n\} \subset \mathbb{R}^2$ um conjunto de pontos onde $2 \leq n < \infty$, definimos como região de Voronoi o conjunto de pontos associado a p_i , onde d é a distância euclidiana

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

temos o conjunto formado por essas regiões sendo $V(A) = V(1), V(2), V(3), \dots, V(n)$ (RODRIGUES, 2019a).

Na figura Figura 21 podemos ver a relação do conjuntos de pontos com o diagrama de Voronoi.

Figura 21 – Diagrama de Voronoi.



Fonte: Thomazthz (2014)

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.

3 Desenvolvimento

3.1 Proposta

Este trabalho tem como proposta a utilização de um modelo de inteligência artificial para segmentação de imagem e permitir os usuários gerarem mapas a partir da seleção de um dos segmentos da imagem.

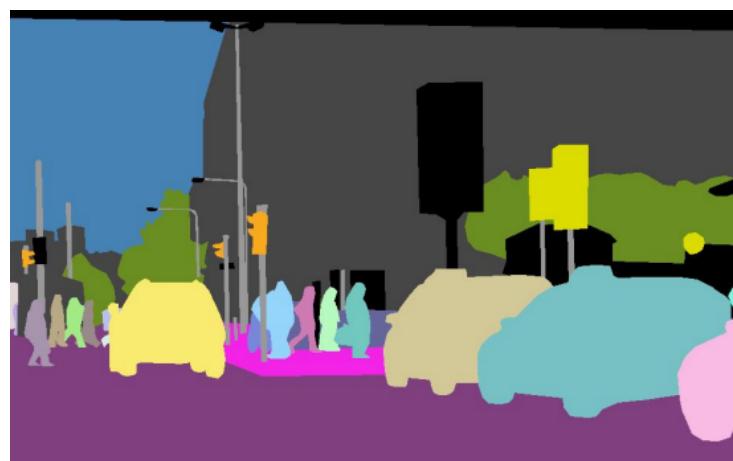
Utilizando o modelo de rede neural totalmente convolucional é possível fazer a segmentação panóptico da imagem da seguinte forma:

Figura 22 – Imagem de entrada para rede neural.



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

Figura 23 – Imagem segmentada de saída da rede neural totalmente convolucional de segmentação panóptico.

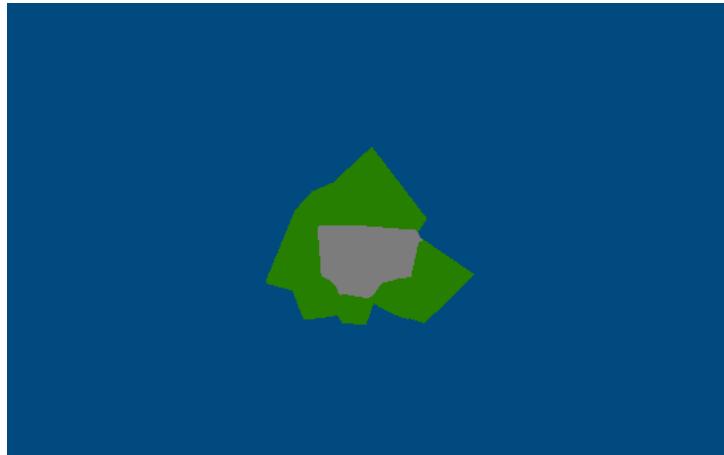


Fonte: [Kirillov et al. \(2019\)](#)

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

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

Figura 24 – Ilha gerada a partir da segmentação de imagem e aplicando um filtro com o diagrama de Voronoi, azul representa oceano, verde floresta, cinza montanhas.



Fonte: Criação própria

3.2 Cronograma

O processo de desenvolvimento será separado em 3 tópicos principais, inteligencia 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.

Inteligencia Artificial

Primeiro será necessário desenvolver o código da rede neural utilizando *TensorFlow*, especificar a quantidade de camadas e definir o tamanho de entrada e saída das imagens. O proximo passo será buscar um dataset para fazer o treinamento desse modelo e em seguida validar as saídas.

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

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 área será, para fazer o 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.

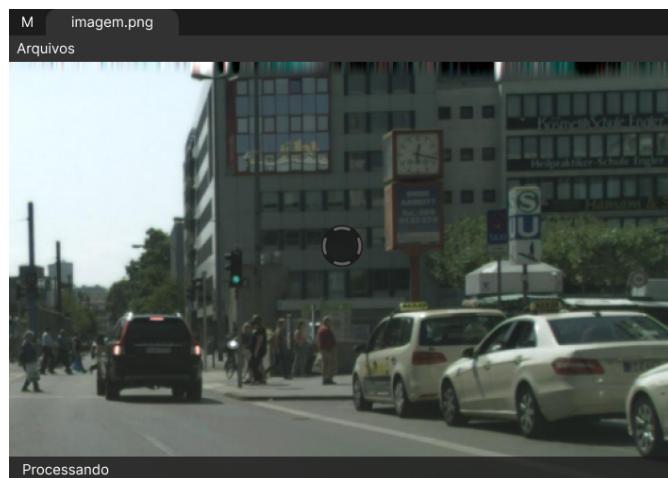
As telas terão o seguinte fluxo:

Figura 25 – 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 26 – Tela de processamento da segmentação



Fonte: Criação própria

Figura 27 – Tela de seleção de segmentação da imagem.



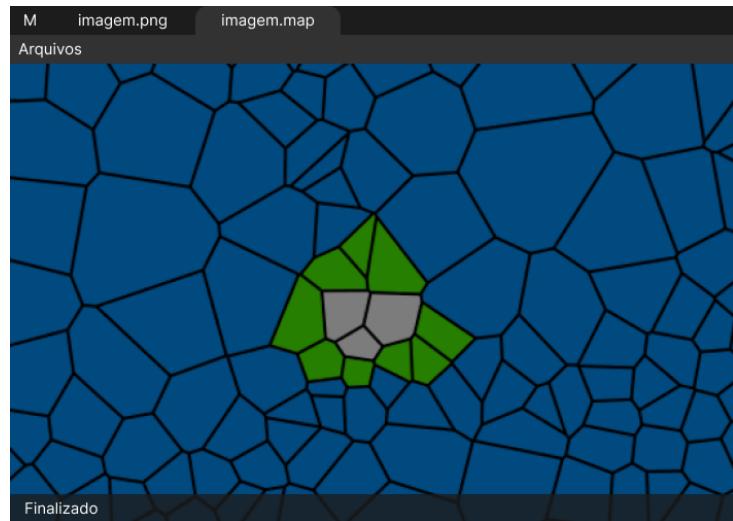
Fonte: Criação própria

Figura 28 – Tela de processamento para geração do mapa com a seleção do segmento.



Fonte: Criação própria

Figura 29 – Tela de resultado com o mapa gerado após processamento.



Fonte: Criação própria

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

O tempo de desenvolvimento será em torno de 1 mês.

Conclusão

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetuer nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.

Sed eleifend, eros sit amet faucibus elementum, urna sapien consectetuer mauris, quis egestas leo justo non risus. Morbi non felis ac libero vulputate fringilla. Mauris libero eros, lacinia non, sodales quis, dapibus porttitor, pede. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi dapibus mauris condimentum nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Etiam sit amet erat. Nulla varius. Etiam tincidunt dui vitae turpis. Donec leo. Morbi vulputate convallis est. Integer aliquet. Pellentesque aliquet sodales urna.

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 24, 25, 28, 29 e 30.
- AMINI, A. *MIT 6.S191: Convolutional Neural Networks*. 2023. Disponível em: <https://www.youtube.com/watch?v=NmLK_WQBxB4&t=1337s>. Citado na página 18.
- 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 na página 18.
- 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 13.
- BRILLIANT.ORG. *Backpropagation*. [S.l.]: Brilliant.org, 2023. <<https://brilliant.org/wiki/backpropagation/>>. Acessado: 2023-06-01. Citado na página 24.
- BROWN, S. *Machine learning, explained*. 2021. <<https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>>. Acessado: 2023-05-11. Citado na página 20.
- 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 13.
- DATASET, C. *Panoptic Semantic Labeling Task - PQ on class-level*. 2023. <<https://www.cityscapes-dataset.com/benchmarks>>. Citado na página 34.
- EDUCATIVE. *Overfitting and underfitting*. [S.l.]: Educatice, 2022. <<https://www.educative.io/>>. Acessado: 2023-06-01. Citado na página 26.
- 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 13.
- 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 22.
- 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 13.
- 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 2 vezes nas páginas 22 e 23.

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 13.

GMBH, H. *Instance Segmentation*. 2023. <<https://hasty.ai/docs/mp-wiki/model-families/instance-segmentor>>. Citado na página 17.

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 19.

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 17.

HAYKIN, S. S. *Neural Networks: A Comprehensive Foundation*. [S.l.]: Prentice Hall, 1999. Citado 3 vezes nas páginas 21, 22 e 26.

HE, K. et al. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. Disponível em: <<http://arxiv.org/abs/1703.06870>>. Citado na página 32.

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 20 e 21.

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 34.

KIRILLOV, A. et al. *Panoptic Segmentation*. 2019. Citado 6 vezes nas páginas 31, 33, 34, 38, 39 e 41.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Citado na página 26.

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 2 vezes nas páginas 14 e 38.

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 36 e 37.

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 14.

- 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 14.
- 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 18.
- MARTI, L.; BARROS, T. Aprendizado profundo: Fundamentos, histórico e aplicações. In: SBC. *Anais do XIV Simpósio Brasileiro de Sistemas Colaborativos*. [S.l.], 2017. Citado 3 vezes nas páginas 21, 22 e 26.
- MOHAN, R.; VALADA, A. Efficientps: Efficient panoptic segmentation. *International Journal of Computer Vision (IJCV)*, 2021. Citado 3 vezes nas páginas 35, 36 e 37.
- 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 19.
- NVIDIA. *NVIDIA Omniverse ACE*. 2021. <<https://developer.nvidia.com/omniverse/ace>>. Acessado em 4 de junho de 2023. Citado na página 14.
- 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 27.
- POLÍGONOS. *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 37.
- 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.fatecitarira.edu.br/index.php/pst/article/view/37>>. Citado na página 23.
- 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 37 e 38.
- RODRIGUES, D. S. M. *Diagrama de Voronoi: uma abordagem sobre jogos*. [S.I.]: Universidade Estadual de Maringá, 2019. <<http://repositorio.uem.br:8080/jspui/handle/1/6748>>. Citado na página 14.
- 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 32.
- 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 13.

SANTOS, P. R. S. dos. *Diagrama de voronoi: Uma Exploração nas Distâncias Euclidiana e do Táxi*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná - UTFPR, 2016. Citado 2 vezes nas páginas 14 e 37.

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 19, 27, 28 e 29.

SCHUMACHER, D. *Synchronized Batch Normalization*. <<https://serp.ai/synchronized-batch-normalization/>>. Acessado em 2023-06-09. Citado na página 37.

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 19.

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 20.

SZELISKI, R. *Computer Vision: Algorithms and Applications*. [S.l.]: Springer, 2022. Citado 2 vezes nas páginas 17 e 18.

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 4 vezes nas páginas 25, 27, 28 e 30.

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 38.

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 30, 31, 32 e 33.

WANGENHEIM, A. von. *Segmentação Semântica*. [S.l.]: Lapix, 2021. <<https://lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/deep-learningsegmentacao-semantica/>>. Acessado em 2023-06-05. Citado 3 vezes nas páginas 30, 32 e 33.

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 13.

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 20.