UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
DEPARTAMENTO DE ESTATÍSTICA E INFORMÁTICA
PROGRAMA DE PÓS–GRADUAÇÃO EM INFORMÁTICA APLICADA

LUCAS VIEIRA GALVÃO

DESENVOLVIMENTO DE MODELOS DE SEGMENTAÇÃO E CLASSIFICAÇÃO PARA IMAGENS DE CAMARÃO

RECIFE – PE

2025

LUCAS VIEIRA GALVÃO

DESENVOLVIMENTO DE MODELOS DE SEGMENTAÇÃO E CLASSIFICAÇÃO PARA IMAGENS DE CAMARÃO

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

ORIENTADOR: Prof. Dr. Péricles Barbosa Cunha de Miranda

RECIFE – PE

2025

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas da UFRPE
Biblioteca Central, Recife-PE, Brasil

G182d Galvão, Lucas Vieira.
Desenvolvimento de modelos de segmentação e classificação para imagens de camarão / Lucas Vieira Galvão. – Recife, 2025.
94 f.: il.

Orientador(a): Péricles Barbosa Cunha de Miranda.
Dissertação (Mestrado) – Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, BR-PE, 2025.

Inclui referências.

1. Segmentação de imagem 2. Imagem – Classificação 3. Inteligência artificial 4. Aquicultura I. Miranda, Péricles Barbosa Cunha de, orient. II. Título

CDD 004

Agradecimentos

Agradeço aos meus pais por sempre terem incentivado a dedicação aos estudos e a busca pelo conhecimento. Ao meu pai, agradeço por ter me dado todas as ferramentas necessárias para que eu pudesse encontrar minha a área de interesse. A minha mãe, agradeço por ser um exemplo como educadora e pesquisadora que me inspirou para buscar novos desafios acadêmicos.

Agradeço ao professor Péricles Miranda por todo apoio que me deu ao longo do processo do mestrado, me guiando na pesquisa, porém permitindo que eu ditasse como conduzi-la. Agradeço também a compreensão com as dificuldades e obstaculos que encontrei pelo caminho.

Agradeço a minha esposa Maria Luiza por todo apoio e incentivo que me deu durante a pesquisa. Também agradeço pela compreensão quando a pesquisa precisou ser colocada a frente de outros planos.

Aos demais que contribuiram, de forma direta ou indireta, para conclusão desse trabalho, deixo o meu muito obrigado!

Resumo

A aquicultura, responsável pelo cultivo de organismos aquáticos como peixes, algas e crustáceos, tem se consolidado como uma alternativa sustentável à pesca extrativista, apresentando crescimento contínuo nas últimas décadas. Esse avanço é especialmente expressivo no continente asiático, mas países como o Brasil também demonstram grande potencial devido à abundância de recursos hídricos e extensa costa marítima. No cenário nacional, destaca-se a produção de camarões, que representa uma fração significativa da aquicultura local. Apesar do crescimento do setor, o processo de biometria, etapa essencial no monitoramento e manejo dos cultivos, em muitos casos, ainda é realizada manualmente, demandando esforço elevado e provocando estresse aos animais, que por sua vez, pode desencadear processo de doenças e mortalidade total do viveiro. Nesse contexto, técnicas de aprendizagem de máquina, especialmente classificação e segmentação de imagens, surgem como alternativas promissoras para automatizar a biometria de forma não invasiva e eficiente. O presente trabalho tem como objetivo realizar uma análise comparativa entre diferentes algoritmos de classificação e segmentação para a identificação e extração de pixels correspondentes ao corpo de camarões em imagens, sendo eles VGGNet, ResNet e GoogLeNet para classificação e FCN, UNet, LinkNet, DeepLab-V3, Attention UNet e TransUNet para segmentação. Essa etapa faz parte de um projeto mais amplo voltado à automação da biometria no cultivo de camarões. A análise considera métricas quantitativas como desempenho, número de parâmetros treináveis, tempo de execução e tamanho dos modelos, além de uma avaliação qualitativa utilizando técnicas de IA explicável. Os resultados obtidos foram capazes de delimitar o algoritmo mais adequado para cada uma das tarefas. Além disso, o uso da técnica de IA explicável SegGradCam possibilitou extrair conhecimento em relação ao comportamento e limitações de cada uma das arquiteturas utilizadas para a segmentação. Ao fim do processo o a arquitetura GoogLeNet se mostrou superior em relação as demais para a tarefa de classificação de imagem. Já para a segmentação, foi encontrado que os algoritmos DeepLab-V3 e o Attention UNet obtiveram a melhor performance.

Palavras-chave: Segmentação de Imagem. Classificação de Imagem. IA Explicável. Aquicultura

Abstract

Aquaculture, which involves the cultivation of aquatic organisms such as fish, algae and crustaceans, has established itself as a sustainable alternative to extractive fishing, with continuous growth in recent decades. This progress is particularly significant in Asia, but countries such as Brazil also demonstrate great potential due to their abundant water resources and extensive coastline. In Brazil, shrimp production stands out, representing a significant portion of local aquaculture. Despite the growth of the sector, the biometric process, an essential step in monitoring and managing crops, is still often performed manually, being labor intensive and causing stress to the animals, which, in turn, can trigger disease outbreaks and lead to complete mortality in the pond. In this context, machine learning techniques, especially image classification and segmentation, have emerged as promising alternatives for automating biometrics in a non-invasive and efficient manner. This study aims to conduct a comparative analysis of different classification and segmentation algorithms for the identification and extraction of pixels corresponding to shrimp bodies in images. The classification algorithms considered are VGGNet, ResNet, and GoogLeNet, while the segmentation algorithms include FCN, UNet, LinkNet, DeepLab-V3, Attention UNet, and TransUNet. This stage is part of a broader project focused on the automation of biometrics in shrimp farming. The analysis considers quantitative metrics such as performance, number of trainable parameters, execution time and model size, in addition to a qualitative evaluation using explainable AI techniques.The results obtained were able to determine the most appropriate algorithm for each of the tasks. In addition, the use of the explainable AI technique SegGradCam made it possible to extract knowledge regarding the behavior and limitations of each of the architectures used for segmentation. At the end of the process, the GoogLeNet architecture proved to be superior to the others for the image classification task. As for segmentation, the DeepLab-V3 and Attention UNet algorithms achieved the best performance.

Keywords: Image Segmentation. Image Classification. Explainable AI. Aquaculture

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ha	*Hectare*
ANN	*Artificial Neural Network*
CNN	*Convolutiuonal Neural Network*
FCN	*Fully Convolutional Network*
FCL	*Fully Connected Layer*
GPU	*Graphics Processing Unit*
ASPP	*Atrous Spatial Pyramid Pooling*
AG	*Attention Gate*
ViT	*Vision Transformers*
MLP	*Multilayer Perceptron*
MSA	*Multi-Headed Self-Attention*
CUP	*Cascade Upsampler*
RAM	*Random Access Memory*

Sumário

1 Introdução

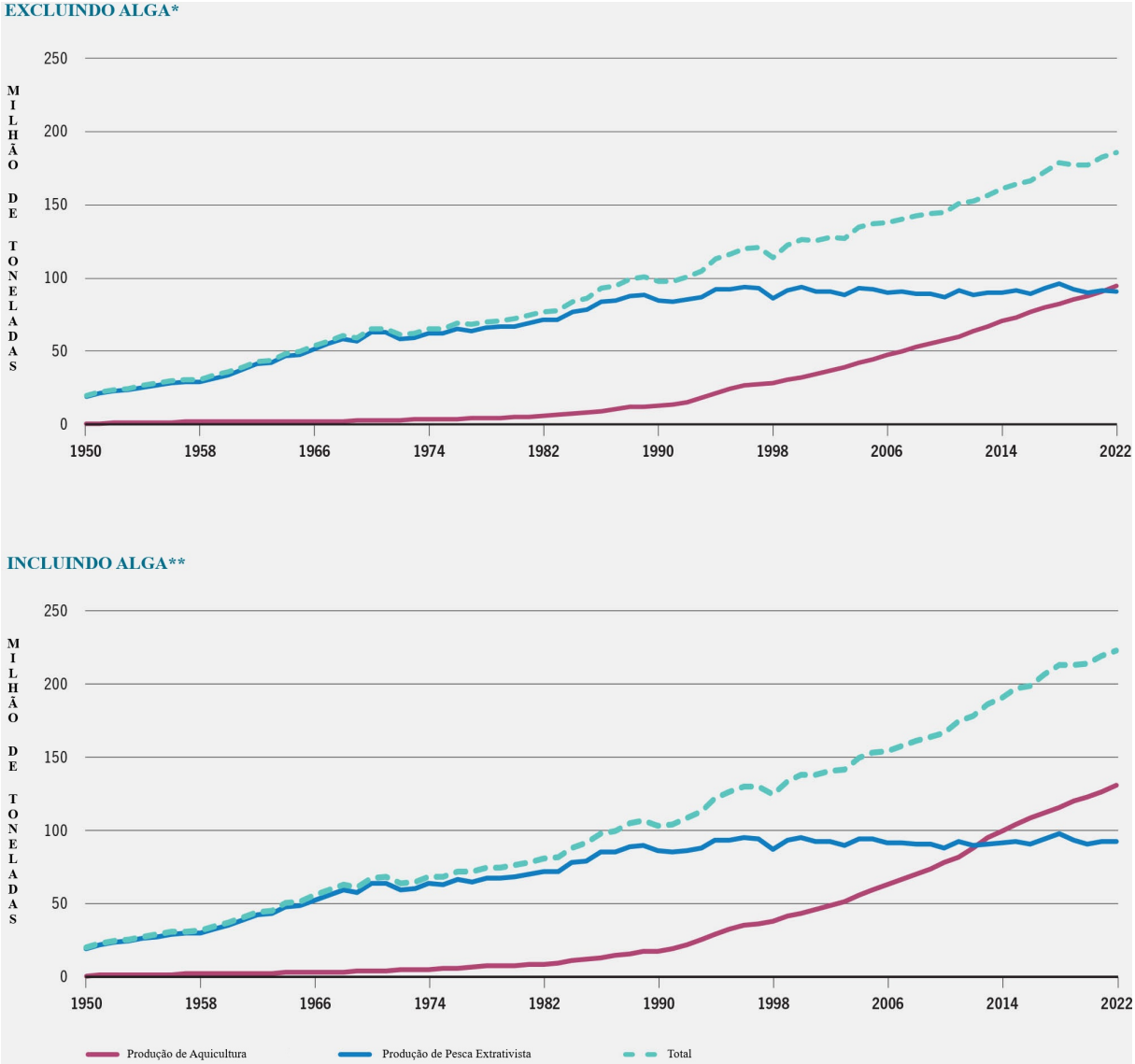
1.1 Motivação

Aquicultura, ou aquacultura, é a atividade de cultivo de animais, incluindo crustáceos, peixes e moluscos, como também de plantas, incluindo algas e macrófitas aquáticas, em águas costeiras ou continentais. Essa atividade pode ser considerada o equivalente aquático da agricultura tradicional.

Segundo (Edwards e Demaine 1997), para categorizar como cultivo é necessário que exista alguma forma de intervenção no processo para aprimorar a produção, seja através da alimentação, estocagem, proteção de predadores, etc. Outro fator importante para essa categorização, é a relação de propriedade, de um indivíduo ou empresa, dos animais ou plantas sendo cultivados. Essas especificações são importantes para diferenciar a aquicultura da pesca extrativista.

A produção total de animais marinhos vem crescendo ano após ano desde 1950, ano em que a FAO, Organização de Alimentação e Agricultura da Organização das Nações Unidas, começou a registrar as quantidades, até 2022, ano do último relatório. No início desse período, a aquicultura representava cerca de 5% da produção total, sendo o restante vindo da pesca extrativista. Essa proporção se manteve constante até o início da década de 1970, quando a produção de aquicultura acelerou seu crescimento, chegando a representar 20% da produção total de animais marinhos na década de 1990, e superando a pesca extrativista em 2022, com representatividade de 51%. A evolução desses valores ao longo dos anos é apresentada na figura 1

Figura 1 – Produção Mundial de Pesca Extrativista e Aquicultura.



Fonte: Adaptado de (Nations 2024).

Nesse contexto, o Brasil apresenta um enorme potencial para o mercado de aquicultura, pois possui uma das maiores bacias hidrográficas do mundo, com aproximadamente 12% de toda a água fresca do planeta, como também 4 milhões ha de represas e reservatórios artificiais de diferentes dimensões. Além dessas medidas de águas continentais, o Brasil também possui uma extensa costa, com cerca de 8700 km de extensão, com diversos estuários (Valenti et al. 2021).

Apesar do evidente aumento da produção de aquicultura no mundo, esse crescimento não ocorre de maneira uniforme para diferentes regiões. Como ilustra a figura 2,

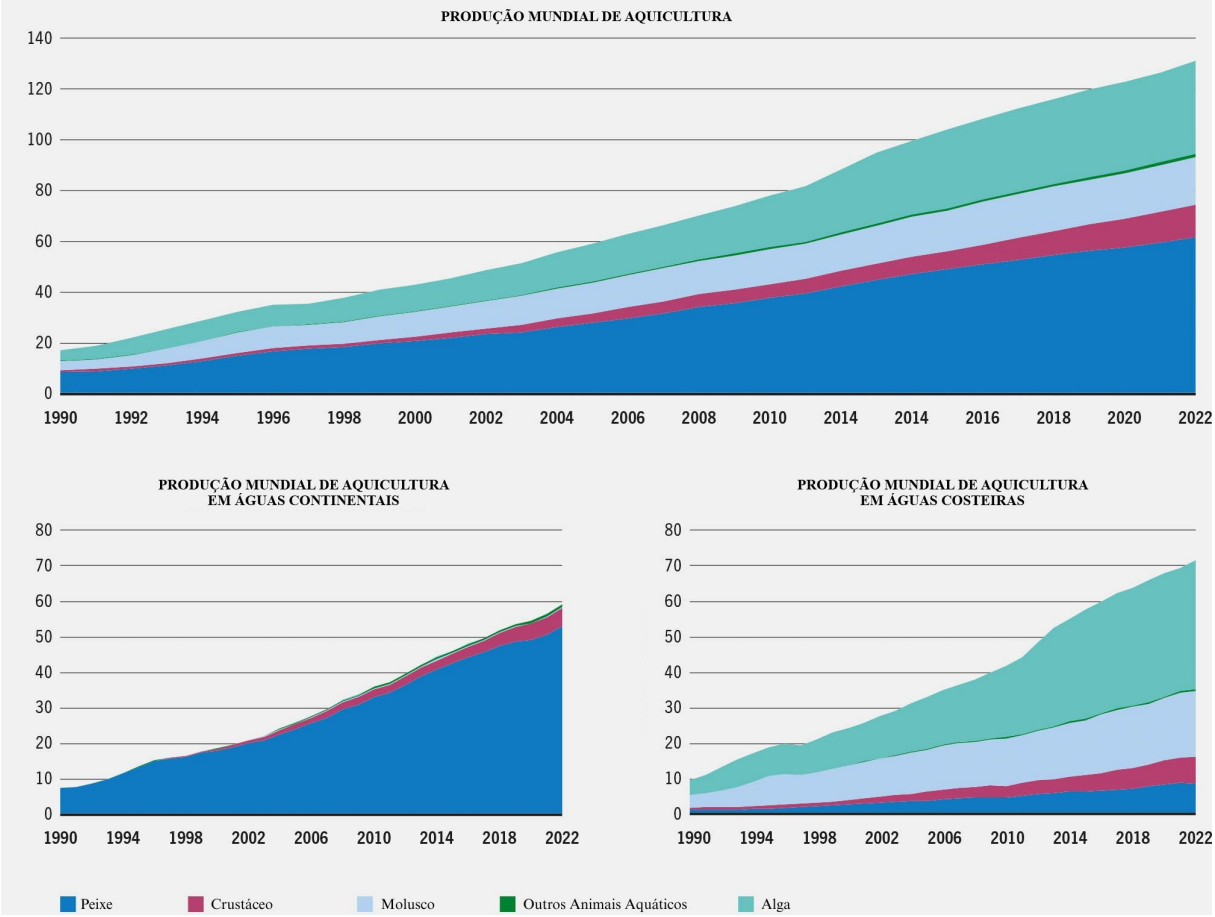
Figura 2 – Proporção de produção Aquicultura em Relação a Produção Total por Região, 2000-2022.

Fonte: Adaptado de (Nations 2024).

essa evolução se deu principalmente no continente asiático, com os demais continentes apresentando maior foco na pesca extrativista. Olhando especificamente para o Brasil, em 2020 foram extraídas 1339 mil toneladas de organismos marinhos, cerca de 68% do total, através da pesca, enquanto a aquicultura foi responsável por 629 mil toneladas, cerca de 32%.

Mesmo apresentando um valor reduzido quando comparado a produção asiática,

Figura 3 – Produção Mundial por Aquicultura, 1990-2022.



Fonte: Adaptado de (Nations 2024).

a aquicultura brasileira se assemelha bastante a essa. Essa semelhança se dá tanto pelo período de profissionalização do processo produtivo, sendo este acelerado a partir da década de 70, como também no caráter da produção, sendo em sua maior parte composta de pequenos produtores, possuindo reservatórios com área inferior a 2 ha (Valenti et al. 2021).

O cultivo de camarões, apesar de distante das quantidades de peixes e algas, representa uma parcela da produção de aquicultura mundial, com um total 7,9 milhões de toneladas produzidas em 2022, cerca de 6% da produção total, como apresenta a figura 3. Já no Brasil, o camarão tem uma maior representatividade na produção nacional, das 629 mil toneladas produzidas em aquicultura em 2020, 63,17 mil toneladas, em torno de 10%, foram de camarão.

Com a expansão do processo de aquicultura ao longo dos anos, tanto em números absolutos em toneladas, como também em números relativos à produção total de animais marinhos, a otimização dessa forma de cultivo se torna cada vez mais relevante. Dentre os obstáculos encontrados para esse tipo de produção, um dos principais ocorre durante

o processo monitoramento e biometria. Essa tarefa envolve a retirada do animal do reservatório, seguido pela medição de características biométricas, como comprimento e peso. Ao fim, dessa etapa, o camarão pode ser descartado ou devolvido ao reservatório. Da presente forma, esse processo demanda de muito esforço humano, sendo necessário uma pessoa para retirar o animal do reservatório e realizar a mensuração. Além disso, a manipulação do camarão causa estresse ao animal, podendo aflorar alguma doença que este possua, possivelmente infectando outros camarões no reservatório, por esse motivo é feito a escolha entre o descarte e a devolução.

Tendo esse procedimento em mente, o uso de aprendizagem de máquina tem se apresentado como uma possível solução que permitiria não apenas reduzir o esforço humano empregado nessa tarefa, como também oferece uma forma de realizar a tarefa de maneira não invasiva, assim reduzindo o estresse a que o animal é submetido (Poonnoy e Asavasanti 2021), (Setiawan et al. 2022) e (Lai et al. 2022).

Aprendizagem de máquina é um campo de ciência da computação dedicado à construção de algoritmos e técnicas para a automatização de soluções para problemas que não são adequadamente resolvidos por métodos convencionais de programação (Rebala et al. 2019). Existem algumas áreas de aprendizagem de máquina capazes de auxiliar no processo descrito acima, uma delas é a classificação de imagem, que consiste em determinar se uma imagem pertence ou não a uma determinada classe. Segmentação de imagem é outra técnica com objetivo semelhante, porém, ao invés de classificar a imagem como um todo, esta realiza sua a classificação a nível de pixel, buscando determinar a qual classe cada um destes pertence.

A hipótese desse trabalho é que as técnicas descritas acima podem ser utilizadas em conjunto para auxiliar no processo de monitoramento e biometria, assim possibilitando a obtenção das imagens do cultivo de forma automatizada ou manual, um modelo de classificação seria capaz de determinar se uma dada imagem possui a presença do animal de interesse, e, por fim, um modelo de segmentação iria extrair a área representada por esse na imagem e com esse segmento poderia se estimar as medidas biométricas através de relação, previamente determinada, das dimensões da imagem com o as métricas que se busca determinar, como, por exemplo, comprimento e peso. Todo esse processo seria realizado de maneira mais rápida, demandando menos esforço humano além de ser uma forma não invasiva de mensuração.

1.2 Objetivo

Dado o cenário apresentado, este trabalho tem como objetivo estabelecer uma metodologia estruturada e replicável para o desenvolvimento e avaliação comparativa entre diferentes algoritmos de classificação e segmentação de imagem para determinar a sua adequação ao problema proposto, sendo este a identificação e extração dos pixels pertencentes ao corpo de camarão em uma imagem. A metodologia proposta compõem uma etapa de um projeto maior, que busca automatizar o processo de biometria de camarão, de forma que, a partir de imagens capturadas, seja possível extrair as medidas biométricas do animal, assim reduzindo o esforço humano necessário, como também, o estresse infligido ao camarão pela manipulação deste.

A análise dos resultados obtidos foi realizada baseada em diversas métricas, sendo elas: a performance em relação a sua tarefa (classificação ou segmentação), quantidade de parâmetros treináveis, tamanho do arquivo resultante, tempo necessário para treinamento e avaliação. Além de aspectos quantitativos, para os algoritmos de segmentação de imagem, também foi realizada uma análise qualitativa através do uso de IA explicável para auxiliar no entendimento do padrão de segmentação de cada um dos modelos.

Este estudo compõe uma etapa de um projeto maior, envolvendo a colaboração entre o Departamento de Computação (DC) e o Departamento de Pesca e Aquicultura (DEPAQ). Esse projeto busca a automatização do processo de biometria para o cultivo de camarão.

Dessa forma, os objetivos deste trabalho são descritos a seguir:

- Construção e treinamento de modelos de classificação e segmentação de imagens utilizando algoritmos do estado da arte.
- Análise quantitativa dos resultados obtidos para os modelos de classificação e segmentação de imagens.
- Análise qualitativa dos resultados obtidos para os modelos de segmentação de imagens.

1.3 Estrutura do Trabalho

Este trabalho segue a seguinte estrutura:

1. Neste presente capítulo 1, foi apresentado o contexto do problema a ser estudado e a motivação para esse estudo.
2. O capítulo 2, introduz todos os conceitos necessários para a compreensão do trabalho desenvolvido. Estes conceitos foram divididos em 6 seções, sendo elas: Cultivo de Camarão, Teste de Significância de Friedman e Post-Hoc Nemenyi, Redes Neurais Artificiais, Redes Neurais Convolucionais, Segmentação de Imagem e IA Explicável.
3. O capítulo 3 apresenta pesquisas recentes sobre temas correlatos ao da pesquisa desenvolvida.
4. O capítulo 4 discorre sobre a metodologia empregada para o desenvolvimento do trabalho, detalhando sobre o ambiente, base de dados e algoritmos utilizados na execução, e também sobre os métodos empregados para a avaliação dos resultados.
5. No capítulo 5 são apresentados e discutidos todos os resultados obtidos após a aplicação dos métodos descritos no capítulo 4.
6. Finalmente, o capítulo 6 detalha as conclusão obtidas após a pesquisa, como também oportunidades que podem ser exploradas em trabalhos futuros.

2 Fundamentação Teórica

2.1 Cultivo de Camarão

O cultivo de camarão pela aquicultura, ou carcinicultura como é conhecido esse ramo, apresenta uma série de desafios, sendo um deles de caráter ambiental, como, por exemplo, o crescente desmatamento de áreas de mangue para a construção de reservatórios e também o despejo de dejetos provenientes do cultivo. Outra adversidade enfrentada na carcinicultura é referente a saúde do camarão, sendo comum o desenvolvimento de doenças nas criações, esse problema é combatido de várias formas, desde uso de antibióticos, até a seleção da alimentação (Bondad-Reantaso et al. 2012). Além das soluções citadas, outra maneira de zelar pela saúde do camarão é reduzindo o estresse a que o animal é submetido.

Estresse foi definido por (Chrousos e Gold 1992) como uma ameaça, por fatores externos ou internos, a homeostase, por sua vez, homeostase seria o estado de equilíbrio harmônico do ambiente mantido pelos seres vivos que possibilitam seu funcionamento ideal. O estresse pode ocorrer por diferentes causas em camarões com diferentes consequências, um dos modos que pode afetar o animal é pela sua contínua manipulação. (Mercier et al. 2006) estudaram os efeitos da manipulação em camarões, os resultados variam a depender do ambiente em que o animal era criado, mas em todos os casos foi encontrado que o metabolismo do animal foi afetado por esse estresse.

2.2 Teste de Significância de Friedman e Post-Hoc Nemenyi

Testes de significância estatísticas, são métodos que permitem quantificar o grau de confiança que as observações de um dado experimento confirmam ou negam uma dada hipótese nula (Zhu 2005). Uma categoria desses testes são os não paramétricos, baseados em *ranking*, que permitem a avaliação de hipóteses de igualdades, ou diferenças, entre variáveis em uma escala ordinal (Sheldon et al. 1996). Sendo um desses o teste de Friedman.

Dado um conjunto de experimentos realizados com mais de um procedimento, sendo obtido um resultado para cada combinação de experimento e procedimento, o teste de Friedman busca determinar a validade da hipótese nula, sendo esta que a variável

independente (procedimento) não tem influência sobre a variável dependente (resultado). Por exemplo, para a tarefa de classificação de imagens de camarão sejam testados três modelos de *machine learning* diferentes (procedimentos), e cada um desses modelos sejam executados 30 vezes (número de experimentos), e para cada par modelo e execução seja aferida a acurácia da classificação (resultado). O teste de Friedman fará uma comparação de forma a confirmar ou negar a hipotese nula que a seleção do modelo não tem efeito sobre o valor da acurácia.

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_{j=1}^{k} R_j^2 \right] - 3N(k+1) \tag{2.1}$$

Para alcançar o resultado, o teste de Friedman utiliza a Expressão 2.1, onde N representa o número de experimentos, k é o número de procedimentos e R_j é a soma dos rankings para cada um dos procedimentos. Já os números 12 e 3 são constantes independentes dos demais valores. O valor resultante é então comparado com o valor crítico do grau de confiânça determinado, caso seja superior a este, a hipótese nula é rejeitada, sendo possível o uso de um teste *post-hoc* para determinar onde se encontra a diferença significante (Sheldon et al. 1996).

O *post-hoc* Nemenyi é um teste não paramétrico que realiza a comparação entre pares pertencentes a um dado grupo de procedimentos para determinar se estes são estatisticamente diferentes. Essa determinação é feita pela mensuração do *ranking* médio de cada um dos procedimentos ao longo dos experimentos, sendo considerados estatisticamente diferente caso a diferença entre essas medidas seja superior a distância crítica calculada (Panichella 2021).

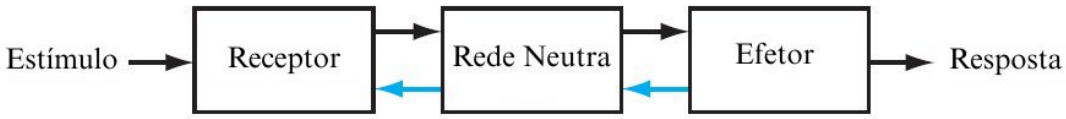
2.3 Redes Neurais Artificiais

Redes neurais artificiais (ANNs), ou somente redes neurais, é uma técnica de aprendizagem de máquina que busca reproduzir a forma que as redes neurais do cérebro humano processam informações para construção de um modelo (Wu e Feng 2018).

O cérebro humano é um sistema de processamento de informação altamente complexo, não linear e paralelizado. Possui componentes estruturais, que se conectam, chamados de neurônios. O sistema nervoso humano pode ser visto como um processo de

três etapas: o cérebro se encontra no centro, onde recebe os estímulos, internos ou externos ao corpo humano, a partir de receptores. Após a captação do estímulo, a informação é processada e transmitida para os efetores, que por sua vez, convertem os impulsos elétricos, gerados pela rede de neurônios, em respostas. Na direção contrária desse fluxo, ocorre um processo de feedback, onde a informação de cada etapa é transmitida de volta a etapa anterior (Haykin 2009). A figura 4 ilustra esse processo onde as setas pretas representam a passagem de informação e as setas azuis o feedback

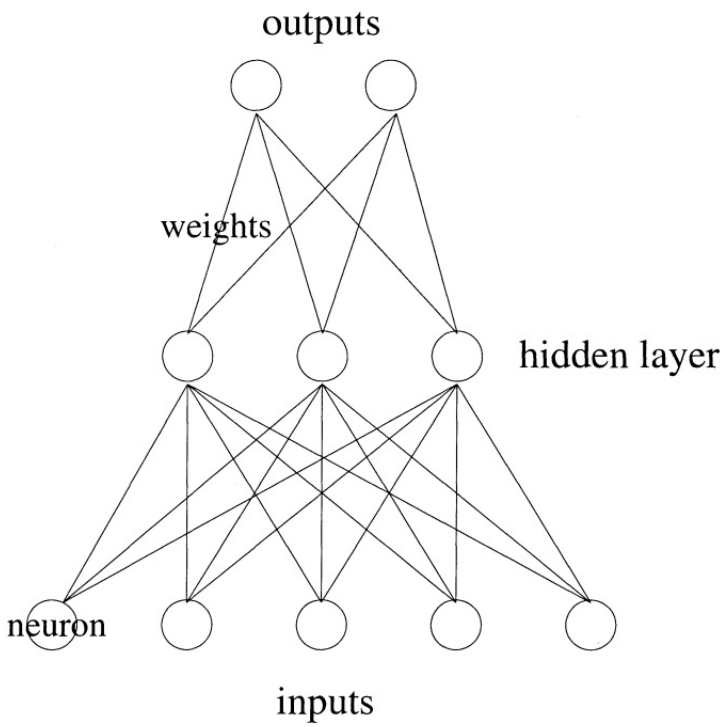
Figura 4 – Representação em Bloco do Sistema Nervoso



Fonte: Adaptado de (Haykin 2009)

A estrutura básica de uma rede neural busca adaptar esse processo do sistema nervo com a seguinte arquitetura, inicia com uma camada de neurônios de entrada, seguida de uma, ou múltiplas, camadas de neurônios escondidas e, finalmente, a camada de neurônios de saída. Cada neurônio nessa rede é conectado a outro, e para cada uma dessas conexões está associado um peso, esse valor varia para cada uma das conexões. Essa arquitetura pode ser visualizada na imagem 5

Figura 5 – Arquitetura Rede Neural



Fonte: (Wang 2003)

Outro elemento importante de redes neurais é a função de ativação, esta é empregada para introduzir não linearidade a rede. A função de ativação também limita a amplitude dos valores dos neurônios, de forma que a rede não seja paralisada por neurônios divergentes. Existem uma ampla gama de funções de ativação, cada uma cumprindo um objetivo específico, entre as principais podem ser citadas: Sigmoide, Tangente Hiperbólica e Relu.

Sigmoide é uma função de ativação não linear que converte valores entre *-inf* e *inf* para valores entre 0 e 1, seguindo a fórmula 2.2. No início do uso de redes neurais, essa função era bastante utilizada, porém caiu em desuso nos últimos tempos por conta de algumas limitações, sendo a principal que para valores absolutos muito grandes, a sua derivada fica muito próxima de zero, o que por sua vez reduz a eficiência do treinamento dos parâmetros (Rasamoelina et al. 2020). Tangente hiperbólica é uma versão atualizada da sigmoide com a conversão de valores variando entre -1 e 1, conforme a expressão 2.3, a principal melhoria oferecida por essa função é que sua inclinação é superior, mitigando o principal problema da sigmoide, porém não o resolvendo.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

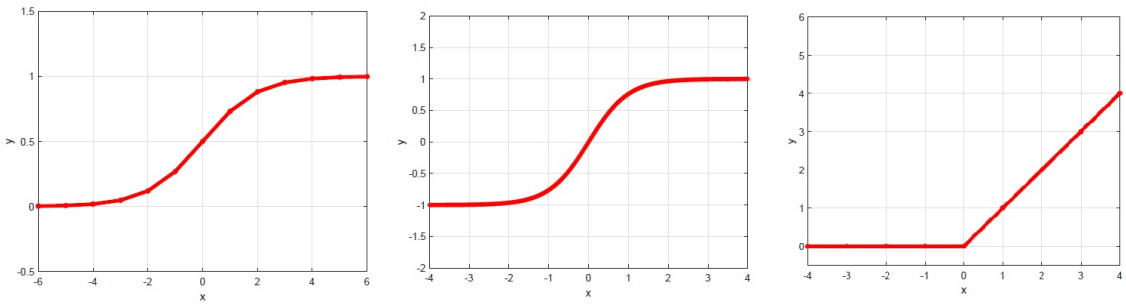
$$f(x) == \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$

Relu é uma função de ativação linear bastante simples, mas que tem se mostrado eficaz. Os valores de saída variam entre 0 e *inf*, sendo 0 para qualquer valor negativo e o próprio valor para os demais casos, conforme a fórmula 2.4. Apesar de apresentar vários pontos positivos, a função Relu também possui uma desvantagem nos casos em que os valores são negativos, pois sua derivada será sempre 0 (Wang et al. 2020).

$$f(x) = \max(0, x) \tag{2.4}$$

Abaixo, na figura 6, segue a representação gráfica das funções de ativação Sigmoide, Tangente Hiperbólica e Relu, respectivamente.

Figura 6 – Representação Gráfica das Funções de Ativação Sigmoide, Tangente Hiperbólica e Relu



Fonte: Adaptado de (Wang et al. 2020)

2.4 Redes Neurais Convolucionais

Redes neurais convolucionais (CNNs) são redes neurais *feedfoward* baseadas no córtex visual de animais, e contruidas para aprender *features* de padrões de baixo nível para alto nível (Yamashita et al. 2018). Essas redes se aproveitam que muitos sinais naturais são hierárquicos, com *features* de alto nível sendo composta da combinação de *features* de baixo nível (LeCun et al. 2015).

A estrutura de CNNs é semelhante à de ANNs, de forma que ambas são compostas de neurônios que são otimizados através do treinamento, recebendo valores de entrada na

primeira camada e retornando o resultado na camada final. A principal diferença entre as redes é pelo fato de CNNs serem otimizadas para o processamento de imagens, então os neurônios são organizados em três dimensões, sendo elas altura, comprimento (essas chamadas de dimensões espaciais) e profundidade.

Existem várias arquiteturas diferentes de CNNs, mas a estrutura base de cada uma é similar, sendo composta por três tipos de camadas diferentes, sendo elas: convolucionais, *pooling* e completamente conectadas (Gu et al. 2018).

2.4.1 Camada Convolucional

A camada convolucional, como o próprio nome indica, é essencial para CNNs. Essa se contrapõem a camada completamente conectada, onde cada neurônio é conectado a todos os neurônios da camada anterior, o que gera uma quantidade muito grande de parâmetros treináveis quando se está trabalhando com imagens. Uma possível solução para esse problema seria reduzir a quantidade de neurônios nas camadas escondidas, porém essa mudança afetaria a qualidade dos resultados da rede. A camada convolucional soluciona essa questão se aproveitando do fato que os pixels possuem alta correlação com os seus vizinhos próximos e baixa correlação com os distantes. Dessa forma, os neurônios das camadas convolucionais se conectam apenas aos pixels ao redor da região da imagem que busca extrair informações (Aghdam et al. 2017). Para isso, a camada faz o uso de filtros, ou *kernels*.

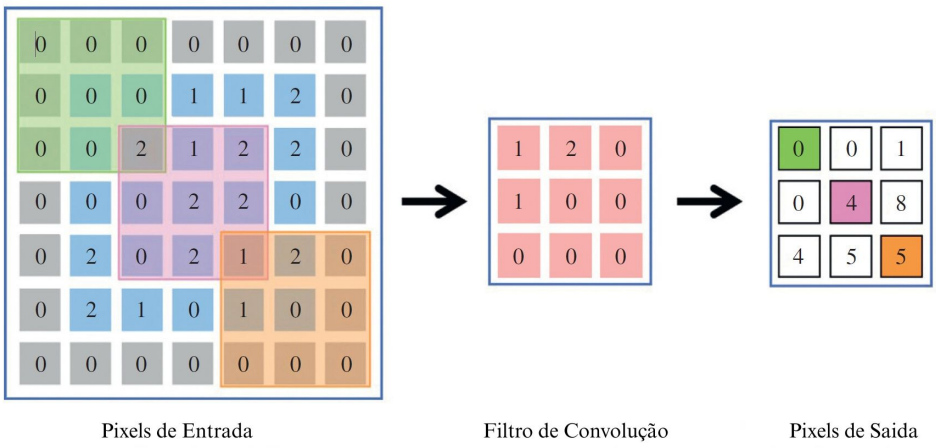
Os filtros são matrizes quadradas com dimensões $k_n \times k_n$, onde n é um valor inteiro, geralmente baixo entre 3 e 5. A convolução é uma operação realizada entre dois tensores, o valor de entrada e o filtro, e retorna um tensor (Michelucci 2019). A operação 2.5 demostra uma convolução entre dois tensores de dimensões 3 x 3, onde a_i é o elemento i da matriz de entrada e k_i é o elemento i do filtro.

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} * \begin{bmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \\ k_7 & k_8 & k_9 \end{bmatrix} = \sum_{n=1}^{9} a_i k_i \tag{2.5}$$

Na maioria dos casos, a convolução é realizada entre uma imagem com dimensões

muito maiores que as do filtro. Para se adequar, o filtro desliza pelo tensor aplicando a convolução em recortes que possuem a mesma dimensão, como demostrado na imagem 7. O tamanho desse deslize é determinado por um parâmetro chamado *stride*, quando o valor deste é 1, o filtro é deslocado uma coluna a direita ou uma linha abaixo da posição inicial na imagem, porém, caso o valor seja superior, como por exemplo 2, o deslocamento será de duas colunas ou duas linhas. Com essa operação, a rede aprende a reconhecer padrões em locais específicos das imagens que acionam os filtros, esse processo é comumente chamado de ativação (O'shea e Nash 2015).

Figura 7 – Convolução

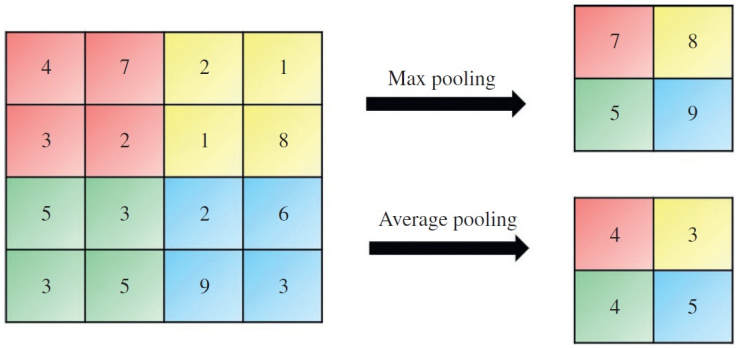


Fonte: Adaptado de (Lei e Nandi 2022)

2.4.2 Camada *Pooling*

A camada *pooling* tem como objetivo principal reduzir a dimensão dos tensores, para tanto faz uso de matrizes quadradas de dimensões $d \times d$ que deslizam ao longo da imagem. Similar a operação de convolução, o deslocamento é determinado pelo parâmetro *stride*, chamado de s. A medida que o filtro passa por uma região da imagem, esse reduz os $d \times d$ valores contidos nessa área para um único valor. A operação mais comum é o *max pooling*, em que é mantido apenas o maior valor, mas em alguns casos também é empregada a *average pooling*, onde é calculada a média dos valores contidos no filtro (Aghdam et al. 2017). A imagem 8 demostra como é feita uma operação de *max pooling* e *average pooling*.

Figura 8 – *Pooling*



Fonte: (Lei e Nandi 2022)

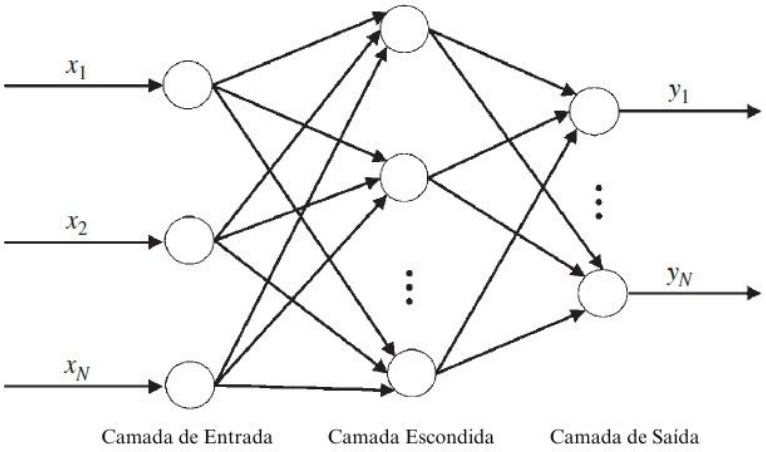
A redução de dimensão realizada pela operação de *pooling* é determinada pelo valor de s, sendo a quantidade final igual a inicial dividida pelo *stride*, conforme a expressão 2.6, onde n_a, n_k e n_b são os números de dimensões da matriz de entrada, filtro e matriz de saida, respectivamente. então, se para $s = 2$, o resultado da operação terá metade do tamanho do original (Michelucci 2019).

$$n_b = \lfloor \frac{n_a - n_k}{s} + 1 \rfloor \tag{2.6}$$

2.4.3 Camada Completamente Conectada

A camada completamente conectada tem o mesmo comportamento que as ANNs, onde cada neurônio está conectado com todos os neurônios da camada anterior e da subsequente. Para tanto a saída da etapa precedente, geralmente, é *flattened*, ou seja, é transformada em um tensor de uma única dimensão, ou um vetor. Esse resultado então é passado para uma, ou mais, camadas completamente conectadas, e, após seu processamento, é encaminhada para a camada de saída da rede. Toda camada completamente conectada é seguida por uma função de ativação, dessa forma o resultado sofre a transformação antes de prosseguir para a próxima camada (Yamashita et al. 2018).

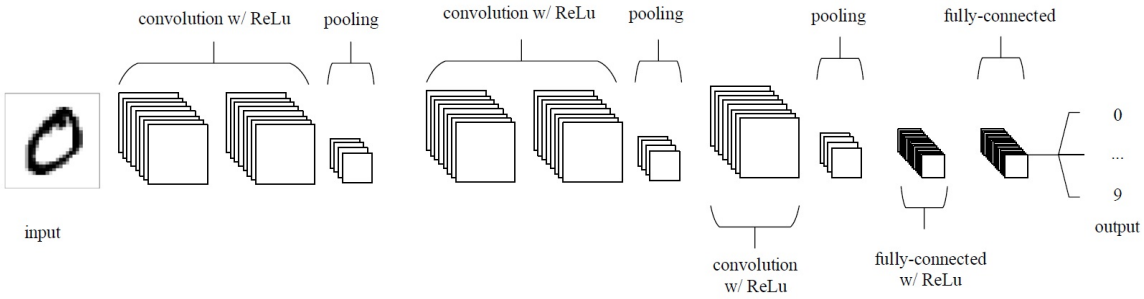
Figura 9 – Camada Completamente Conectada



Fonte: Adaptado de (Lei e Nandi 2022)

A imagem 10 ilustra o fluxo de uma CNN, desde do recebimento da imagem de entrada, passando por cada uma das camadas, até o resultado final.

Figura 10 – Aquitetura CNN



Fonte: (O'shea e Nash 2015)

2.4.4 Arquiteturas CNN

Como citado anteriormente, as CNNs são otimizadas para o processamento de imagens, sendo bastante utilizadas em tarefas de classificação de imagem. Existe uma grande diversidade de arquiteturas CNNs desenvolvidas buscando superar limitações em relação ao desempenho, profundidade e eficiência computacional. Entre essas estão três as três redes selecionadas para este trabalho, sendo elas: VGGNet, ResNet e GoogLeNet.

A escolha dessas arquiteturas justifica-se por sua relevância na literatura e desempenho comprovado para variadas tarefas de classificação. Nesta seção será feita uma apresentação das características e principais inovações dessas redes.

2.4.4.1 VGGNet

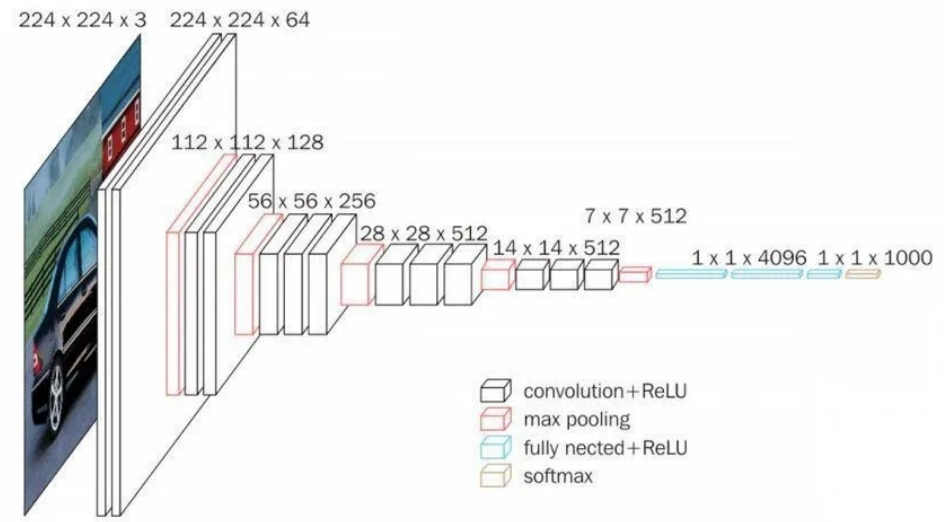
VGGNet é uma arquitetura de CNN apresentada por (Simonyan e Zisserman 2014) com objetivo de obter uma melhora na performance de classificação com o uso de redes mais profundas. Para tanto, o modelo tem como entrada imagens de dimensões fixas 224×224, no formato RGB, com uma única etapa de pré-processamento onde é subtraído de cada pixel o valor da médio da respectiva camada de cor.

Foram desenvolvidas várias versões da rede, alterando a profundidade, porém seguindo a mesma estrutura. A imagem é passada por uma série de camadas de convolução, utilizando filtros 3×3, sendo esse o menor tamanho capaz de capturar informações direcionais, com valor fixo para o *stride* de 1. Também é utilizado *padding*, técnica que consiste na adição de n pixels vázios as bordas da imagem, com $n = 1$, de forma que a resolução original da figura seja preservada.

Entre algumas das camadas de convolução, porém não para todas contidas na rede, são inseridas camadas de *max pooling*, sendo cinco a quantidade total dessas camadas para a estrutura completa. Cada uma dessas operações são realizadas com uma janela 2×2 e *stride* com valor 2. Ao fim da sequência de camadas de convolução, são colocadas três camadas completamente conectadas, sendo as duas primeiras com 4096 canais, cada, e a última com 1000 para realizar a classificação de acordo com a quantidade de classes da base em que a rede foi treinada, a ILSVRC [1], sendo essa configuração a mesma para todas as versões. A última etapa da arquitetura é uma operação *SoftMax*. A figura 11 apresenta a arquitetura da versão VGG16.

[1] https://www.image-net.org/challenges/LSVRC

Figura 11 – Arquitetura VGG16



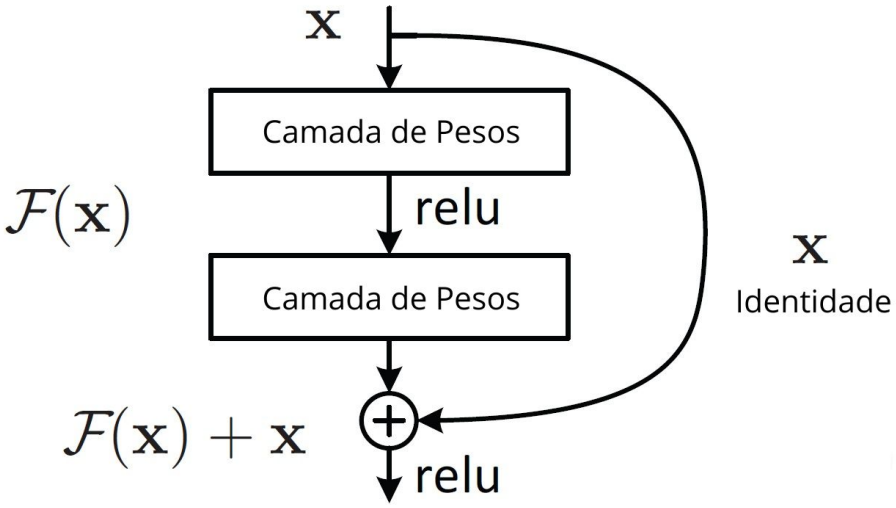
Fonte: <https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918>

2.4.4.2 ResNet

As arquiteturas ResNet, nome comumente utilizado para as redes neurais residuais, foram introduzidas por (He et al. 2016), com o intuito de mitigar o problema do desaparecimento do gradiente para redes profundas, onde a acurácia sofre uma degradação a medida que se aprofunda na estrutura. Com isso, a adição de camadas a uma arquitetura ocasiona em um aumento do erro de treinamento.

Para reduzir esse efeito, a ResNet é construída utilizando blocos residuais, onde, em vez de realizar operações para obter a saída desejada, o objetivo é encontrar o resíduo entre a entrada do bloco e essa saída. Dado a entrada, ou identidade, x e o mapeamento para saída $H(x)$, um bloco da ResNet busca encontrar $F(x)$, de forma que $F(x) = H(x) - x$, assim $H(x)$ seria obtido pela relação $H(x) = F(x) + x$. Para realizar esse cálculo, o bloco é estruturado com uma sequência de camadas convolucionais que possuem como entrada x e retornam $F(x)$, ao fim do bloco $F(x)$ é somado com a identidade x através de uma conexão direta, de forma que x "pula" a sequência de camadas convolucionais. A figura 12 apresenta um exemplo desses blocos residuais.

Figura 12 – Bloco Residual



Fonte: Adaptado de (He et al. 2016)

Com o uso de blocos residuais, é possível o desenvolvimento de redes mais profundas sem o problema da saturação da acurácia, outra vantagem é que as conexões diretas não adicionam parâmetros novos nem aumentam a complexidade computacional, com isso, a arquitetura ResNet pode ser construída com uma grande quantidade de camadas, com versões variando de 18 até 152. As camadas convolucionais, na maior parte, utilizam filtros 3×3, variando de acordo com as dimensões de saída, sendo usados esses valores quando essas coincidem com as dimensões de entrada, porém quando as dimensões são reduzidas pela metade, o número de filtros é dobrado. A rede é finalizada com uma camada de *global average pooling* e uma camada completamente conectada com 1000 canais acompanhadas da função de ativação *SoftMax*.

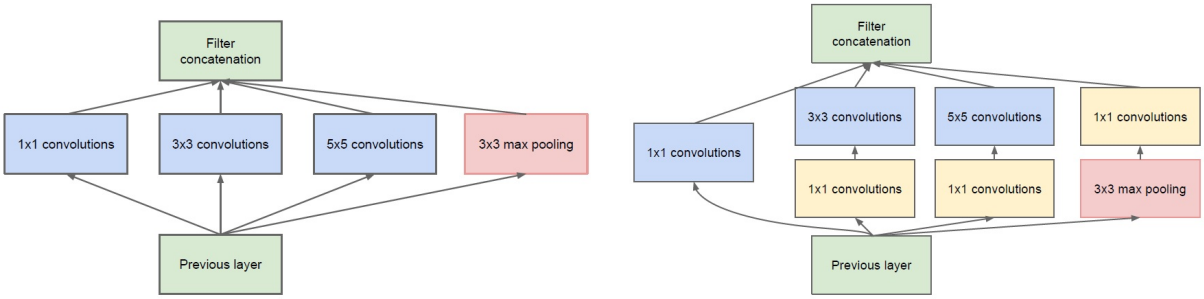
2.4.4.3 GoogLeNet

GoogLeNet é uma arquitetura desenvolvida por (Szegedy et al. 2015), com o principal objetivo sendo a criação de uma rede mais profunda e larga e, ao mesmo tempo, limitar o crescimento do custo computacional. Para alcançar essa meta, o modelo se utiliza de estudos que apontam a melhora na performance de multiplicação de matrizes esparsas quando estas são agrupadas em submatrizes densas. Com isso foram construídos os blocos *inceptions*, a principal inovação da GoogLeNet.

Os blocos *inceptions* consistem na aplicação isolada de três operações de convolução

com diferentes tamanhos de filtro (1×1, 3×3 e 5×5) e uma operação *max poooling* 3×3, sendo, ao fim, os resultados dessas concatenados para formar o mapeamento final do bloco. Considerando que o resultado de cada operação deve possuir as mesmas dimensões espaciais, é utilizado *padding* de 1 no *max poooling*, de forma a preservar os valores de entrada. Uma limitação dessa estrutura é que o seu custo computacional pode ser muito elevado, em particular por causa da convolução 5×5. Para reduzir esse custo, foram adicionadas convoluções 1×1 antes das 3×3 e 5×5, buscando diminuir as dimensões de entrada destas operações. A figura 13 apresenta a estrutura do bloco na sua forma *naive* (a), sem aplicação do procedimento descrito anteriormente, e com redução de dimensionalidade (b).

Figura 13 – Bloco Inception



(a) Inception module, naïve version (b) Inception module with dimension reductions

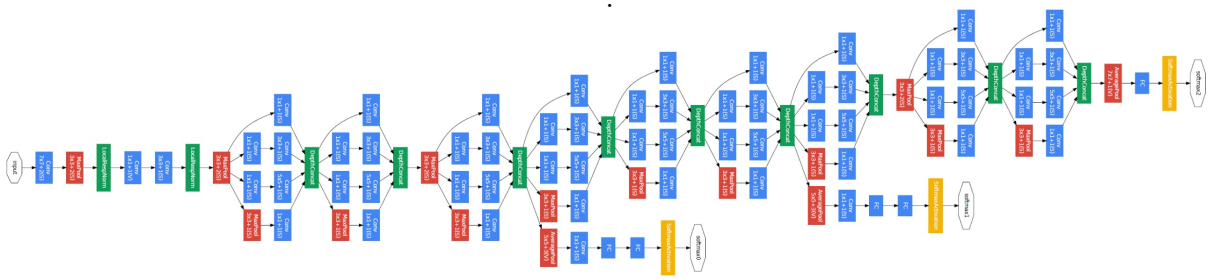
Fonte: (Szegedy et al. 2015)

Uma preocupação dos autores ao construir a rede foi que a eficiência da propagação do gradiente ao retornar pela estrutura seria prejudicada pela profundidade desta. A solução desenvolvida foi a criação de classificadores auxiliares conectados as camadas intermediárias, se comportando como CNNs menores, incentivando o poder de discrição desde esses estágios, potencializando o sinal do gradiente e reduzindo o risco de *overfit*. Esses classificadores auxiliares são composto por uma camada *max pooling* 5×5 e *stride* 3, seguida por uma convolução 1×1, duas camadas completamente conectadas e por fim a função de ativação *SoftMax*.

A rede começa com uma convolução 7×7 e *stride* 2, seguida de *max pooling* 3×3 e *stride* 2, convolução 1×1 e 3×3, ambas com *stride* 2, e novamente *max pooling* 3×3 com *stride* 2. Após essas operações inicias, os dados são passados por conjuntos de blocos *inceptions* com redução de dimensionalidade. Em duas etapas intermediárias são incluídos

classificadores auxiliares, descritos anteriormente. Uma ilustração da rede é apresentada na figura 14.

Figura 14 – Arquitetura GoogLeNet



Fonte: (Szegedy et al. 2015)

2.4.5 *Transfer Learning*

Modelos de aprendizagem de máquina são construídos utilizando um conjunto de dados para treinamento e outro para testes. Tradicionalmente, esses conjuntos possuem distribuições similares, sendo a performance do modelo prejudicada quando esses diferem. Porém, em alguns casos, a quantidade de dados disponíveis é limitada, nessa situação, uma alternativa é o uso de *transfer learning*.

Transfer learning, ou transferência de aprendizado, é uma técnica de aprendizado de máquina que permite utilizar um modelo treinados com dados pertencentes a um subdomínio para realizar previsões em dados de um subdomínio diferente, desde que ambos pertençam a um mesmo domínio comum (Weiss et al. 2016). Um exemplo dessa aplicação seria o uso de um modelo de classificação de imagens de animais como cães e gatos, para a classificação de outras espécies, pois, mesmo se tratando de diferentes imagens das utilizadas durante o treinamento, elas possuem aspectos semelhantes, como o formato, dimensões, números de camadas, posição de algumas características importantes para classificação (olhos, boca, orelhas).

O objetivo de *transfer learning* é melhorar o treinamento de um modelo para uma dada tarefa utilizando aprendizados obtidos para outra tarefa semelhante. Dessa forma, existem três maneiras que esse método pode impactar no treinamento, sendo a primeira fornecendo um melhor estimador inicial, considerando que, para um modelo tradicional, esse é criado com valores aleatórios sendo alterados para se adequar aos dados ao longo do treinamento, já com *transfer learning* é utilizado um modelo já treinado.

Outro possível impacto é a redução do tempo necessário para o aprendizado, já que parte do conhecimento é transferido no início do treinamento, este é mais rápido. Por último, *transfer learning* também pode contribuir com uma melhora na performance, já que o modelo conta com informações adicionais que não possuiria com o conjunto de dados utilizado (Torrey e Shavlik 2010).

Apesar das vantagens apresentadas, o uso de *transfer learning* também pode ser responsável por pioras no modelo, esses casos são conhecidos como transferência negativa. Sendo *transfer learning* o uso de um modelo desenvolvido para uma tarefa específica, para a solução de outra tarefa, é necessário que essas sejam bastantes relacionadas, de forma que o aprendizado obtido pela primeira possa auxiliar na realização da segunda (transferência positiva), de forma que os ganhos com seu uso sejam superiores a possíveis divergências entre as tarefas que possam prejudicar o treinamento (transferência negativa) (Torrey e Shavlik 2010).

2.5 Segmentação de Imagem

Segmentação de imagem é uma técnica de visão computacional que tem como objetivo dividir uma imagem em grupos discretos de pixels. Esse método pode ser visto como um problema de classificação, onde, em vez de buscar determinar a que classe uma dada imagem pertence, a finalidade é realizar a classificação a nível de pixel, definindo a que grupo cada um desses pertence (Minaee et al. 2021). A imagem 15 apresenta um exemplo de segmentação de imagem.

Figura 15 – Segmentação de Imagem

Fonte: (Lei e Nandi 2022)

O termo segmentação de imagem se refere ao campo que envolve uma série de aplicações diferentes, cada uma atendendo uma necessidade específica. Dentre as técnicas contidas nesse termo estão: Detecção de objetos, segmentação semântica, segmentação de instância e segmentação panóptica.

Detecção de objeto pode ser vista como a mais trivial das técnicas citadas, servindo, muitas vezes, de base para as demais. Esse método tem como objetivo identificar, em uma imagem, instâncias de objetos pertencentes a determinadas classes, como, por exemplo, pessoas, carros ou animais. Dessa forma, a detecção de objeto busca responder a pergunta: onde está cada objeto.

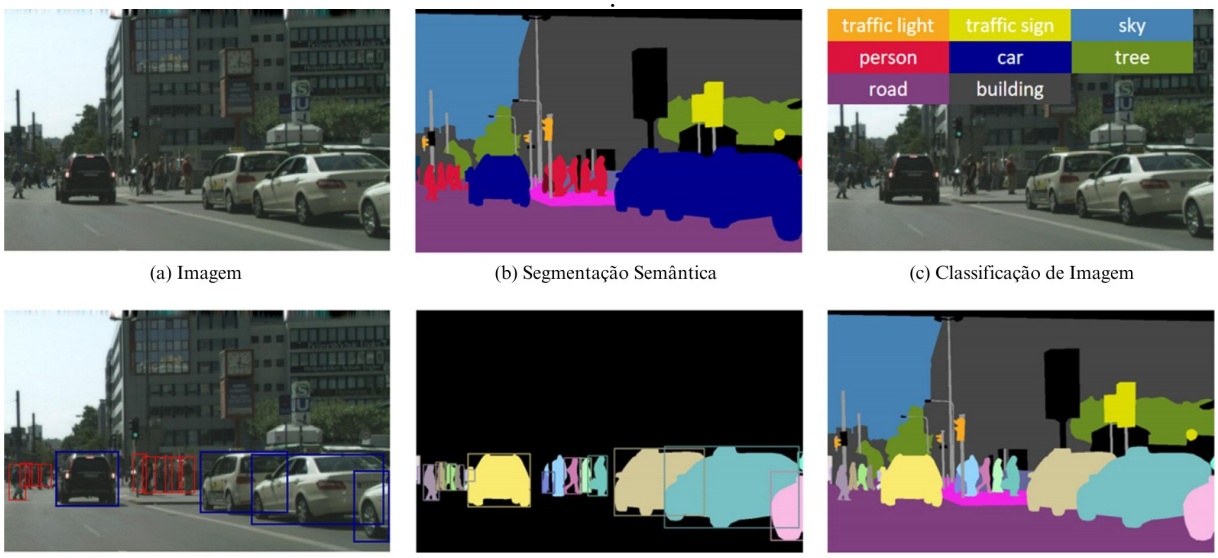
Do ponto de vista de aplicação, detecção de objeto pode ser dividida em dois grupos de pesquisa, detecção de objeto geral e aplicações de detecção. A primeira se refere a exploração de métodos de detecção de diferentes tipos de objetos sob um *framework* único que simule o comportamento da visão e cognição humana. Já o segundo se concentra no estudo de aplicações específicas, como detecção de pedestre ou de texto (Zou et al. 2023).

Segmentação semântica é um método de segmentação de imagem que busca classificar cada pixel entre uma das classes predefinidas, com todos os pixels da imagem pertencente a uma dada categoria formando uma única entidade. Já a segmentação de instância tem objetivo similar, porém, esta diferencia entre múltiplas ocorrências da mesma classe, considerando cada uma dessas uma entidade separada (Ghosh et al. 2019).

A segmentação panóptica busca combinar a aplicação da segmentação semântica e de instância. Para tanto, o modelo classifica cada pixel de acordo com a sua classe e atribui um valor para identificar a qual instância dessa categoria o pixel pertence. Porém, para algumas classes não existe a necessidade de diferenciar entre ocorrências separadas, nesses casos o valor de identificação da instância é irrelevante (Kirillov et al. 2019).

A imagem 16 ilustra as quatro técnicas de segmentação de imagem descritas.

Figura 16 – Técnicas de Segmentação de Imagem



(a) Imagem (b) Segmentação Semântica (c) Classificação de Imagem

(d) Detecção de Objeto (e) Segmentação de Instância (f) Segmentação Panóptica

Fonte: Adaptado de (Hao et al. 2020)

2.5.1 Arquiteturas de Segmentação Semântica

Para a tarefa de segmentação semântica, foram utilizadas seis arquiteturas estabelecidas na literatura, sendo elas: Fully Convolutional Networks (FCN), UNet, LinkNet, DeepLab-V3, Attention UNet e TransUNet. Essas redes foram selecionadas por apresentarem aplicações de diferentes técnicas para o problema de segmentação semântica, desde de soluções iniciais como FCN e UNet, que estabeleceram a base para arquiteturas futuras, como também abordagens mais avançadas com uso de mecanismos de atenção e *transformers*, como as redes Attetion UNet e TransUNet, respectivamente. Nesta seção, serão detalhadas as arquiteturas selecionadas, com ênfase nas principais inovações introduzidas por cada uma.

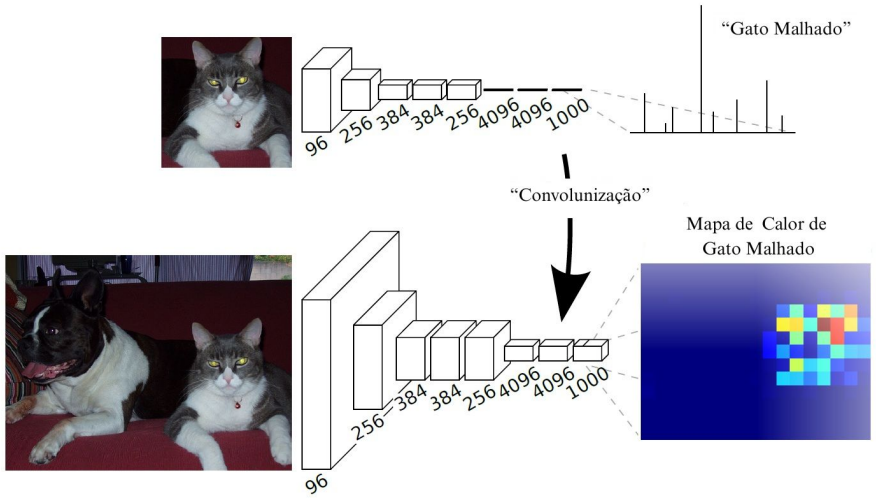
2.5.1.1 *Fully Convolutional Networks* (FCN)

Fully convolutional networks, ou FCNs, são um tipo de CNN que podem ser adaptadas para que, ao invés do seu uso mais comum de classificar uma imagem, essas redes tenham como objetivo realizar a segmentação, dessa forma, no lugar de computar uma função não linear, a rede irá calcular um filtro não linear. Com essa mudança, a FCN pode receber como entrada uma imagem de qualquer tamanho, e retornará uma de

dimensões espaciais proporcionais (Long et al. 2015).

Para ser capaz de realizar essa operação, as FCNs transformam um dos três componentes comuns de uma CNN, a camada completamente conectada é alterada para uma de convolução. Da forma que é comumente utilizada, a FCL recebe um valor fixo de entradas e, após o processamento, o seu resultado não possui dimensões espaciais. Porém, de forma análoga, essa operação pode ser vista como uma convolução onde o filtro cobre todas as regiões da entrada, se aproveitando dessa adaptação, o modelo pode ser construído para retornar o mapeamento da imagem segmentada, com a marcação correta disponível para cada célula. A figura 17 apresenta a construção dessa arquitetura.

Figura 17 – Transformação FCL para camada convolucional



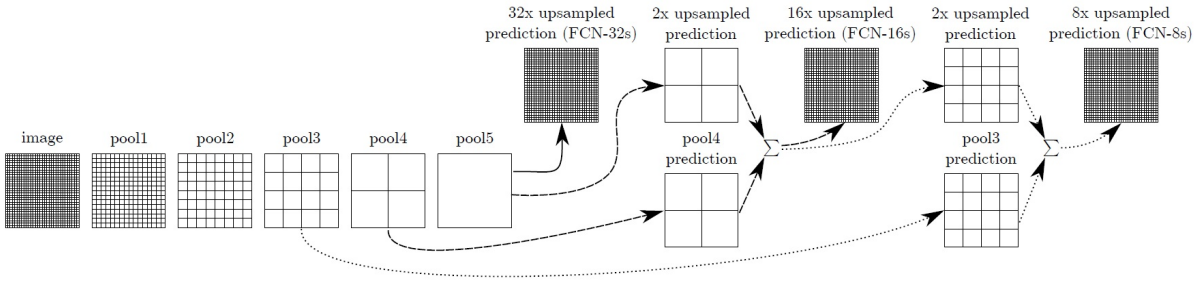
Fonte: Adaptado de (Long et al. 2015)

Porém, apesar dessa adaptação, as dimensões espaciais da imagem sofrem reduções em cada camada, por meio das operações de convolução e *pooling*. Essa redução ocorre a medida que aos dados passam por cada uma das camadas da rede, resultando em uma imagem final de tamanho inferior a original. Para solucionar essa questão, é empregado um processo de *upsampling*, que consiste em realizar uma convolução com valores fracionais para o *stride*, por exemplo onde em uma operação comum teria como *stride* o valor s, no *upsampling* seria usado $\frac{1}{s}$, por esse motivo essa técnica também é chamada de desconvolução.

Apesar de modelos para segmentação semântica construídos a partir da adaptação, por meio das técnicas descritas acima, de arquiteturas estabelecidas de CNN para

classificação de imagem apresentarem performance satisfatória para métricas de avaliação, os resultados dos modelos não possuem um refino esperado para a tarefa. Essa limitação ocorre porque a medida que a rede se aprofunda ela é capaz de obter informações mais profundas, porém ao custo de informações localização espacial, que são mais ricas nas camadas inciais. Com esse fator em vista, são criadas ligações que combinam os resultados da rede com o das camadas mais rasas, dessa forma o modelo é capaz de fazer predições locais que respeitam a estrutura global. Essa arquitetura está ilustrada na figura 18

Figura 18 – Arquitetura FCN



Fonte: (Long et al. 2015)

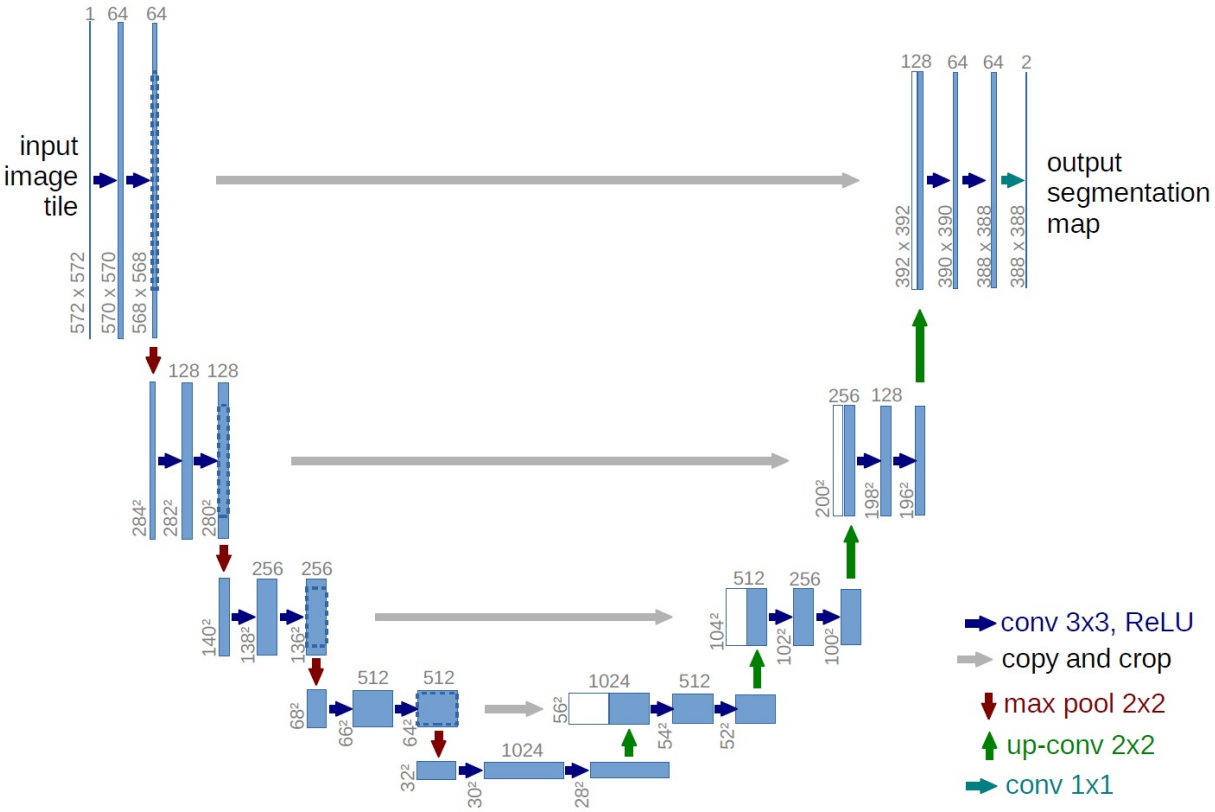
2.5.1.2 UNET

UNet é uma arquitetura desenvolvido para segmentação semântica, tendo como base o modelo FCN desenvolvido por (Long et al. 2015), modificando este para alcançar melhores resultados utilizando uma quantidade reduzida de imagens para o treinamento. A principal mudança da UNet em relação ao seu antecessor, é o emprego de grande número de canais para o processo de *upsampling*, permitindo que a rede propague informações de contexto para camadas de maior resolução (Ronneberger et al. 2015). Por conta dessa estrutura, as etapas da UNet são quase simétricas, resultando em uma arquitetura em formato de U, podendo ser visto na imagem 19, o que dá o nome do modelo.

Assim como a FCN, a UNet não possui nenhuma camada completamente conectada, contendo apenas operações de convolução e *pooling*, utilizando apenas a parte válida de cada convolução, ou seja, o mapa de segmentação somente possui pixel que o contexto está disponível na imagem original. Para classificar pixels nas bordas da imagem, o contexto faltante é extrapolado a partir do espelhamento da imagem de entrada, essa estratégia permite o uso da rede para imagem maiores reduzindo limitações devido ao uso de memória

de GPUs.

Figura 19 – Arquitetura UNet



Fonte: (Ronneberger et al. 2015)

A arquitetura da rede UNet pode ser dividida em duas partes, contração (do lado esquerdo) e expansão (do lado direito). A etapa de contração é semelhante a uma típica CNN, excluindo a camada completamente conectada, sendo aplicadas repetidamente o conjunto de duas convoluções 3×3 consecutivas, cada uma seguida da função de ativação ReLu, finalizando com *max pooling* 2×2 com *stride* 2 para *downsampling*. Após cada operação de *max pooling* o número de canais é dobrado.

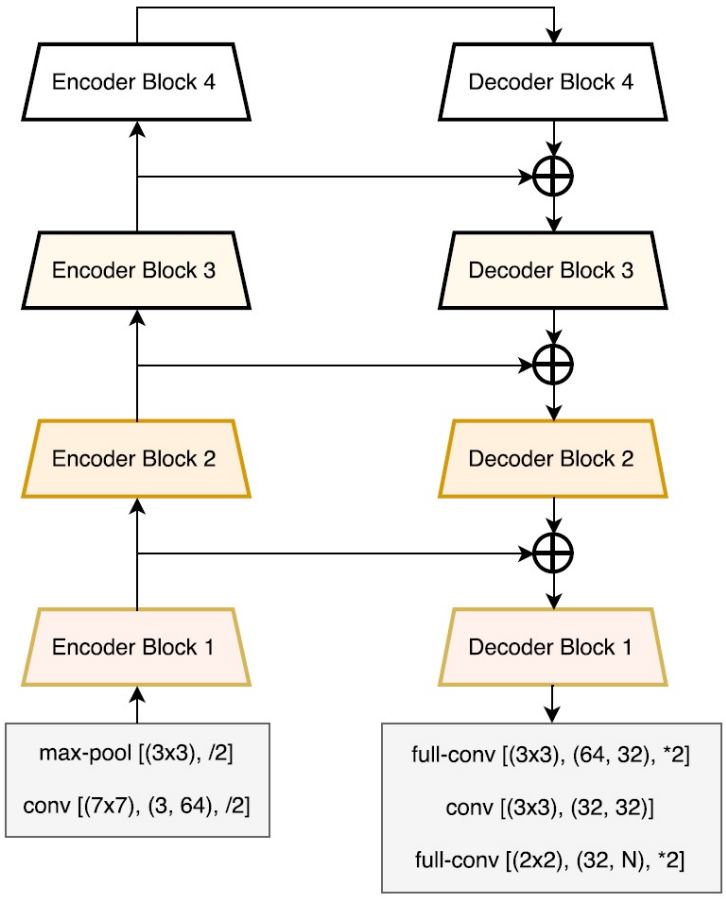
A etapa de expansão segue a mesma lógica que a de contração, porém invertida. Para essa parte da arquitetura, os conjuntos são formados por uma desconvolução 2×2, operação descrita em 2.5.1.1, que reduz o número de canais pela metade. O resultado desse procedimento é então concatenado com mapa da parte correspondente da etapa de contração, essa junção é realizada para que as informações das camadas mais rasas não sejam perdidas a medida que a rede se aprofunda, de maneira similar a construção da FCN. Após esse cálculo, o mapa concatenado passa por duas convoluções 3×3, ambas

seguidas pela função de ativação ReLu. No final da etapa de expansão, o resultado passa por uma última camada de convolução 1×1 que mapeará os 64 canais para a quantidade de classes desejadas.

2.5.1.3 LinkNet

A arquitetura LinkNet, similar a UNet, possui duas etapas sequenciais, aqui chamadas de *encoder* e *decoder*, respectivamente. A ilustração dessa estrutura é apresentado na imagem 20, onde *conv* indica uma operação de convolução e *full-conv* é uma convolução completa. Além disso, a notação /2 significa *downsampling* por um fator de 2 através de uma *strided convolution*, e *2 sinaliza o *upsampling* com *stride* 2. Após a execução de cada convolução indicada, é realizado um batch normalization seguido pela função ReLu. A parte esquerda da imagem representa o *encoder* e a direita o *decoder* (Chaurasia e Culurciello 2017).

Figura 20 – Arquitetura LinkNet

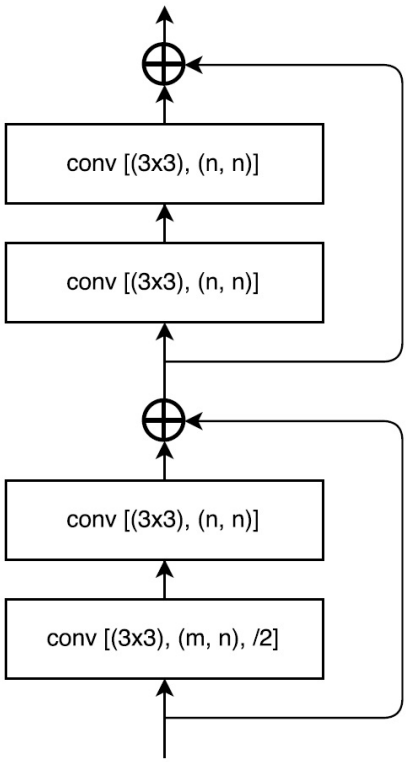


Fonte: (Chaurasia e Culurciello 2017)

A LinkNet busca melhorar a capacidade de predição da rede sem grande acréscimo ao tempo de processamento, desvantagem presente em modelos para detecção de objetos em tempo real, como por exemplo YOLO e Fast RCNN. Para tanto, informações são passadas diretamente do *encoder* para o *decoder* correspondente, fazendo com que dados que seriam perdidos ao longo da discriminação sejam preservados, e reduzindo o tempo de processamento, pois não serão necessários novos parâmetros e operações para reaprender essas informações (Chaurasia e Culurciello 2017).

Uma das limitações desse tipo de arquitetura é a perda de informações espaciais ao longo do *encoder*, devido as convoluções e *pooling*, geralmente sendo corrigida pela recuperação desses dados pelo *decoder*, através dos índices de *pooling* armazenados ou por *upsampling*.

Figura 21 – Encoder Block LinkNet



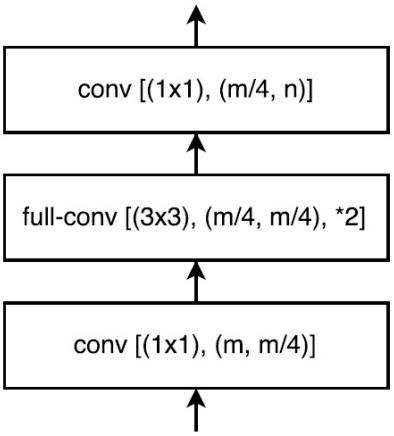
Fonte: (Chaurasia e Culurciello 2017)

O *encoder* é iniciado com um bloco onde é realizado uma convolução com filtro 7×7 e *stride* 2 na imagem de entrada, seguido por um *max pooling* espacial com área 3×3 e *stride* 2. Após essa etapa inicial, o *encoder* consiste de uma sequência de *encoder blocks*, que possuem a mesma estrutura, demostrada na imagem 21. Buscando reduzir o

custo computacional do modelo, sem afetar a performance, a rede ResNet18 é utilizada como *encoder*, essa rede foi escolhida pois é consideravelmente mais leve que outras comumente empregadas por outras arquiteturas para essa tarefa, como por exemplo VGG16 e ResNet101, e alcança resultados de segmentação superiores aos *benchmarks*.

O *decoder* segue um processo similar ao *encoder*. Após finalizada esta etapa, a imagem é passada por uma série de *decoder blocks*, apresentados na imagem 22. A rede é então finalizada com um bloco com uma convolução 3×3 entre duas convoluções completas 3×3.

Figura 22 – Decoder Block LinkNet



Fonte: (Chaurasia e Culurciello 2017)

A contribuição da LinkNet está na ligação direta entre etapas. A rede é construída de forma que as informações de entrada de cada bloco sejam repassadas para seus equivalentes. Mas não apenas nesse caso, por usar uma arquitetura ResNet para o *encoder*, o modelo usa de conexões residuais, que liga o valor ativado na camada $n-1$ com o valor resultante da camada $n+1$, combinando esses valores. Dessa forma as informações espaciais são preservadas ao longo desse processo.

2.5.1.4 DeepLab-V3

Um dos principais desafios para modelos de *deep learning* para segmentação de imagem, como visto pelas arquiteturas apresentadas anteriormente, é a perda de resolução da imagem a medida que esta se aprofunda na rede, afetando a performance para predições

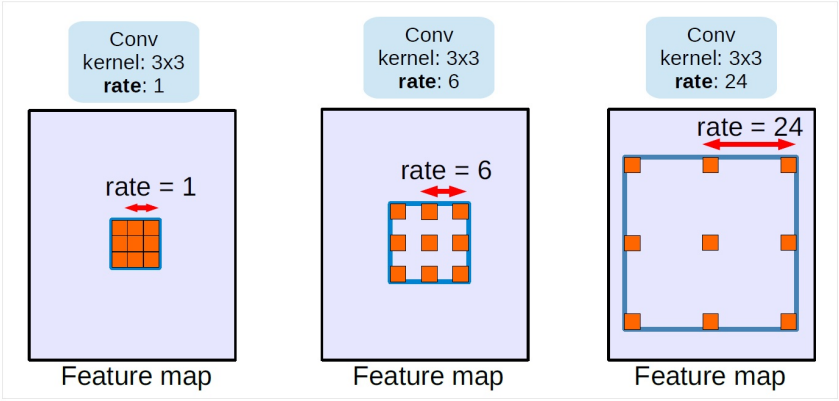
que dependem de informações espaciais. O DeepLab-V3 busca solucionar essa limitação fazendo o uso de convolução *atrous*, ou convolução dilatada.

A convolução dilatada é descrita pela expressão 2.7, onde para cada posição i do resultado y e filtro w, a convolução é aplicada ao mapa de entrada x, com k representando a posição relativa ao centro do filtro que será aplicado. O fator que diferencia essa operação da convolução comum é o grau de dilatação, indicado pelo elemento r, que representa o deslocamento das posições utilizadas na convolução. Esse processo é o equivalente de convolucionar o mapa de entrada x inserindo $r \breve{} 1$ zeros entre elementos consecutivos do filtro, dessa forma, a convolução usual seria um caso especial com $r = 1$ (Chen et al. 2017).

$$y[i] = \sum_{x} x[i + r \cdot k]w[k] \tag{2.7}$$

O termo *atrous* é originado do francês, *à trous*, que significa com buracos, que descreve o processo de adicionar intervalos entre os elementos selecionados para a convolução, adaptando o campo de visão do filtro através da modificação do fator r, como pode ser visto na imagem 23. Outra vantagem dessa técnica é que possibilita controlar explicitamente o quão profundo a rede computará as informações, através do termo *output stride* que indica a proporção do tamanho entre a imagem de entrada na camada e a saída da rede, por exemplo, se em uma dada camada a imagem seja 32 vezes menor do que o resultado final do modelo, o valor do *output stride* será 32. Esse termo pode ser utilizado para limitar a redução das dimensões espaciais da imagem, sendo observado um valor especificado, as camadas subsequentes são substituídas por convoluções dilatadas com r maior que 1. Dessa forma a rede consegue extrair informações mais densas sem a necessidade de aprender novos parâmetros (Chen et al. 2017).

Figura 23 – Convolução Atrous

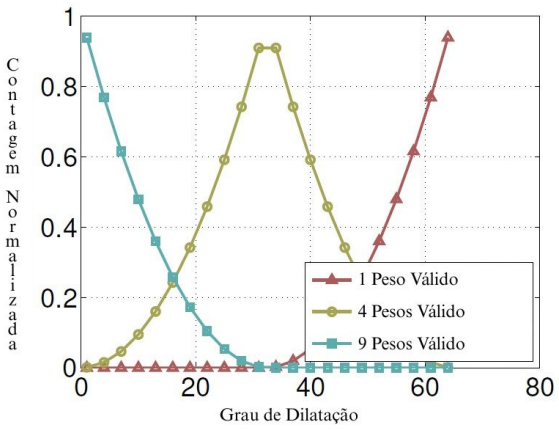


Fonte: (Chen et al. 2017)

Para a implementação de convolução dilatada na rede, DeepLab-V3 emprega o uso de *Atrous Spatial Pyramid Pooling* (ASPP), uma adaptação de *Spatial Pyramid Pooling*, técnica que combina o resultado informações resampleadas em escalas diferentes para classificação de uma área da imagem em uma escala arbitrária. A modificação feita pela ASPP é o uso de diferentes graus de dilatação, também é realizado um *batch normalization*.

Uma limitação encontrada para ASPP, é a redução da quantidade de valores válidos dos filtros, valores contidos dentro da imagem que não correspondem a valores nulos adicionado além das bordas (*padding*), a medida que o grau de dilatação é elevado. Para os casos onde esse valor se aproxima das dimensões da imagem, a operação tem resultado praticamente iguais a convoluções 1×1, pois apenas o valor central do filtro será válido, como apresentado pela figura 24.

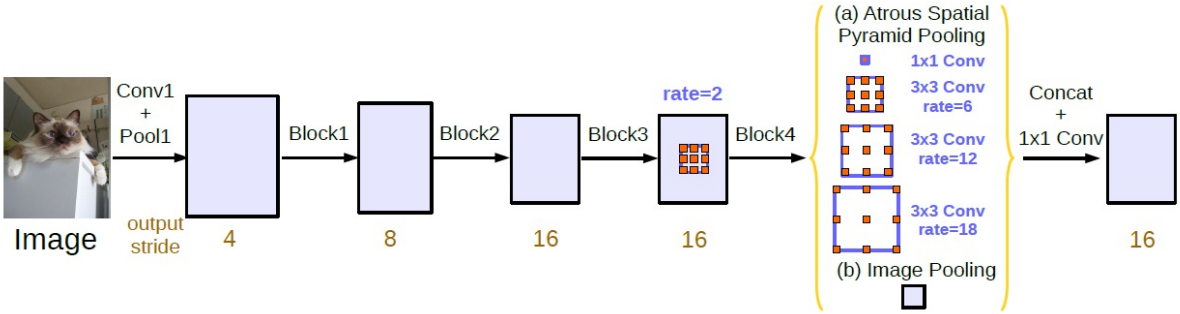
Figura 24 – Gráfico quantidade de pesos válidos por grau de dilatação para um filtro 3 X 3 aplicado a uma imagem 65 X 65



Fonte: Adaptado de (Chen et al. 2017)

Buscando corrigir essa limitação incorporando informações de contexto global ao modelo, foi realizada uma operação de *global average pooling* que é combinado com o resultado ASPP, que consiste em uma convolução 1×1 e três 3×3 com grau de dilatação 6, 12 e 18 (para output *stride* igual a 16). O valor retornado é passado por uma convolução 1×1 com batch normalization, seguido de outra convulção 1×1 que gerará o resultado final da rede. Esse processo é ilustrado pela figura 25.

Figura 25 – Módulos paralelos com convolução dilatada



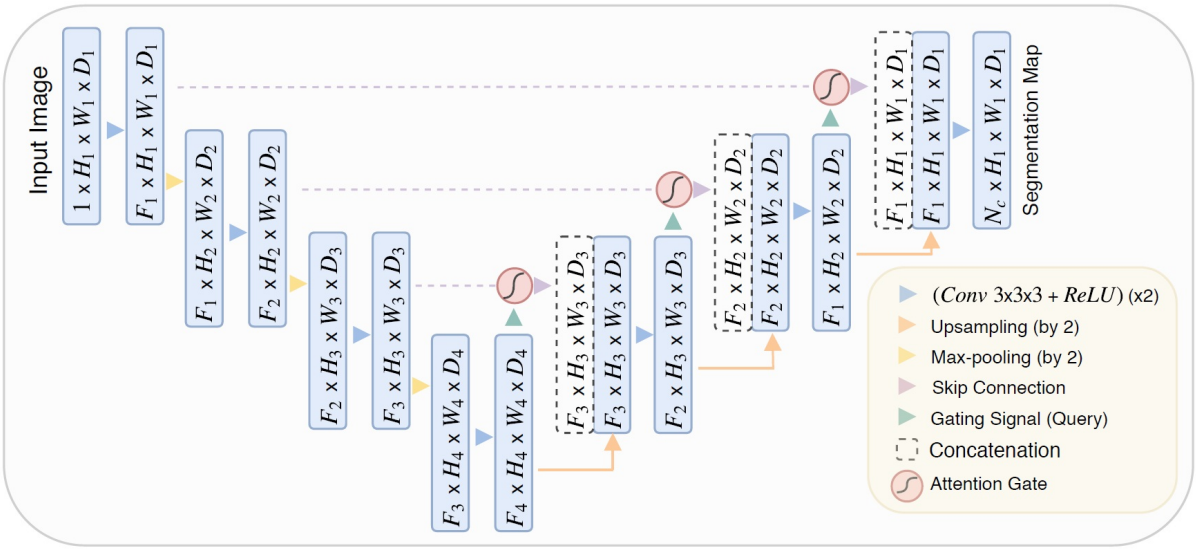
Fonte: (Chen et al. 2017)

Realizando todas essas operações, a arquitetura DeepLab-V3 é capaz de alcançar uma boa performance para a tarefa de segmentação de imagem. Porém, para tanto, é necessário um elevado custo computacional, sendo essa sua principal desvantagem em relação aos demais modelos apresentados.

2.5.1.5 Attention UNet

A arquitetura Attention UNet é uma abordagem para o problema de segmentação semântica, buscando otimizar o modelo para casos onde o objeto de interesse para a segmentação apresenta variações em relação ao tamanho e formato dentro da imagem. Esse algoritmo é uma adaptação da rede UNet, e assim como ela, foi desenvolvida para tarefa de segmentação em imagens médicas, onde as variações citadas são comuns (Oktay et al. 2018).

Figura 26 – Arquitetura attention UNet

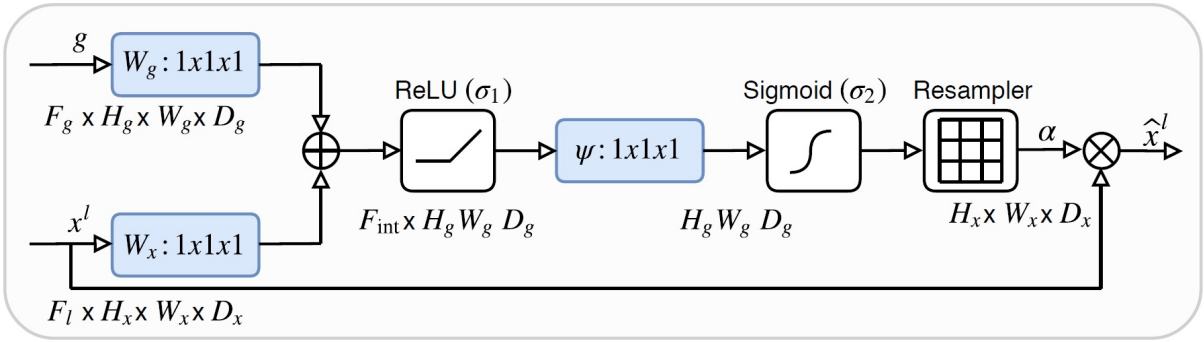


Fonte: (Oktay et al. 2018)

A contribuição da attention UNet para sua predecessora é, como o nome indica, o uso *attention gates* (AGs) na arquitetura, que atuam nas ligações das camadas da etapa de contração com a sua correspondente na etapa de expansão da rede UNet, como indicado na figura 26. Essa mudança busca alcançar uma melhora na performance do modelo, reduzindo a quantidade de processamento de informações com baixa contribuição para predição. Sendo os dados das camadas mais rasas menos trabalhados, e, consquentemente, sem o mesmo poder de discriminação, que os das camadas mais profundas, o uso de AGs destaca as regiões da imagem que mais interessam para o modelo, suprimindo a ativação de informações em áreas identificadas como irrelevantes. Com isso, apenas informações espacias que efetivamente irão complementar os resultados das camadas mais profundas são passados através de *skip connection*.

O uso de modelo de atenção em CNNs pode ser classificado em duas categorias: *hard (stochastic) attention* e *soft (deterministic) attention*. O primeiro utiliza de recortes de áreas da imagem para inferência, é uma operação não diferenciável e para o treinamento depende de uma técnica baseada em *sampling* chamada *REINFORCE* (Jetley et al. 2018). Já a *soft attention*, método utilizado pela Attention UNet, associa pesos as regiões da imagem, dessa forma é diferenciável e pode ser treinada em conjunto com o modelo (Xu et al. 2015).

Figura 27 – Attention Gate



Fonte: (Oktay et al. 2018)

A figura 27 demostra o funcionamento do AG, onde x^l representa o resultado da camada correspondente da etapa de contração e g os valores recebidos da camada anterior, ambos mapas sofrem uma convolução 1×1, com *stride* 1 para g e 2 para x^l, dessa forma ambos terão as mesmas dimensões, o que possibilita a soma executada em sequência, destacando os valores alinhados em ambos. Após a adição dos valores dos mapas, o resultado passa pela função de ativação ReLu, seguido por uma convolução 1×1. Por fim, é aplicada a função Sigmoide e o resultado sofre um *upsample* para as dimensões originais de x^l, resultando em uma matriz com os coeficientes de atenção α. Após todas as operações, o mapa obtido pela multiplicação de x^l pelos coeficientes de atenção α é passado para a próxima camada de maneira semelhante a como x^l é passado na rede UNet. A medida que a rede é treinada, os valores de α vão sendo atualizados destacando as áreas de interesse e suprimindo as demais.

2.5.1.6 TransUNet

TransUNet é um modelo de segmentação para imagens médicas que emprega o uso de mecanismos de auto atenção da perspectiva de predição sequencial, adicionando *vision transformers* (ViT) à arquitetura UNet, já estabelecida para essa tarefa (Chen et al. 2021). ViT é uma adaptação de *transformers*, estrutura desenvolvida para modelos de redes neurais para análise de texto, para o uso na classificação de imagens.

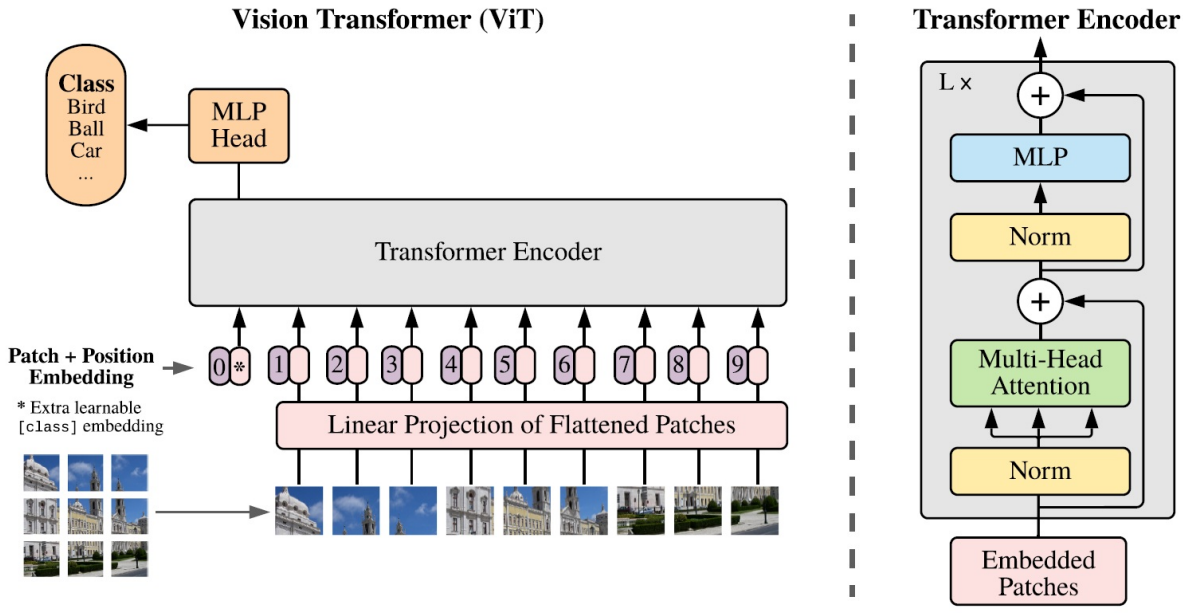
Sendo criado para trabalhar com texto, é necessário algumas mudanças para adequar o uso de *transformers* para imagens. Originalmente, esses modelos recebem, como entrada, uma sequência de *embeddings* de 1 dimensão, dessa forma para ser utilizada, a imagem

sofre uma mudança antes do transformer, das suas dimensões iniciais $x \in \mathbb{R}^{H \times W \times C}$ para uma sequência de patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, onde ($H$, W representam a resolução original da imagem, C é a quantidade de canais, P^2 é a resolução de cada patch e $N = HW/P^2$ é o número de patches, e, consequentemente, o tamanho da sequência de entrada do transformer. Como o transformer trabalha com um vetor de tamanho D constante por todas as suas camadas, os patches são achatados e mapeados para essas D dimensões com uma projeção linear treinável, o resultado dessa etapa é chamado de *patch embeddings* (Dosovitskiy et al. 2020).

Após essa transformação, *embeddings* posicionais são adicionados aos *patch embeddings*, de forma que os dados mantenham suas informações de contexto ao longo da rede. Além disso, também é adicionado o *embeddings* treinável a sequência de *patches*, $z_0^0 = x_{class}$, que ao fim da rede, z_L^0, será associada com uma MLP *head*, consistindo de um MLP com uma camada escondida no pré treinamento e uma camada linear no afinamento.

Os embedding são passados pelo *encoder* do transformer, que contém um bloco de *multi-headed self-attention* (MSA) seguido de um bloco MLP. Antes de cada bloco é realizado uma normalização e após são aplicadas conexões residuais, combinando com os dados antes do bloco. Essa sequência é executada L vezes. Todo o processo descrito pode ser visualizado na figura 28.

Figura 28 – Arquitetura Vision Transformer (ViT)



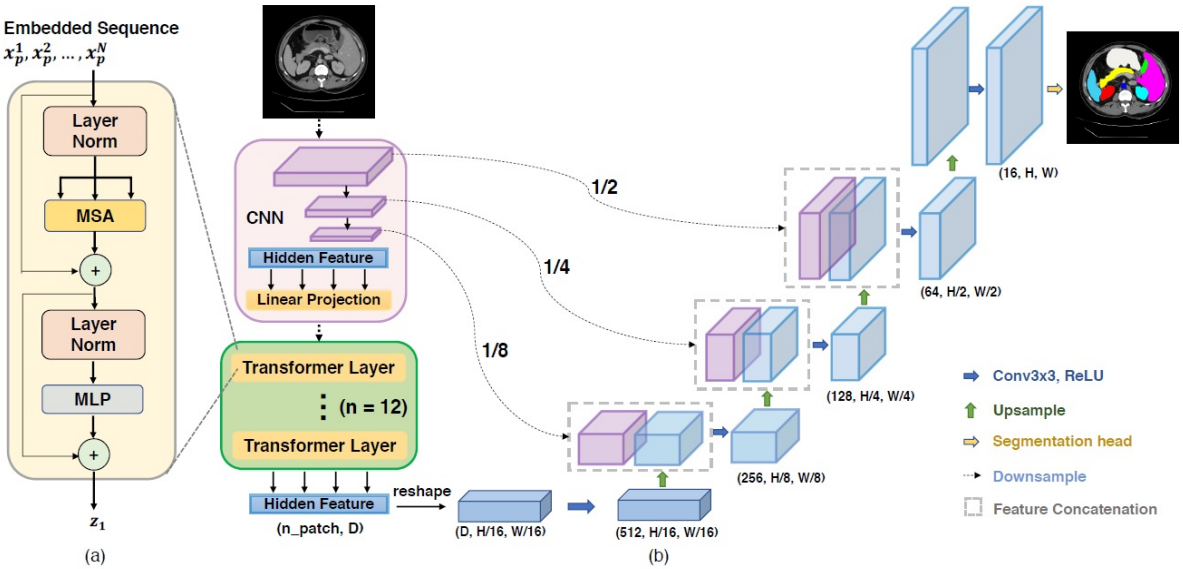
Fonte: (Dosovitskiy et al. 2020)

Com o intuito de implementar o uso de ViT para segmentação de imagens, o primeiro experimento foi utilizar transformer para extrair informações densas dos *patches*, como é feito para classificação, e após realizar o *upsample* para reconstruir a imagem. Porém os resultados obtidos com esse método não foram satisfatórios, pois informações relacionadas a localização vão perdendo o nível de detalhamento à medida que os dados se aprofundam no *transformer*, que não é recuperado com o *upsample*.

Para compensar essa limitação, a TransUNet possui uma estrutura híbrida, que emprega tanto *transformers* como arquiteturas de segmentação com CNN, combinando informações espaciais detalhadas de alta resolução da CNN com o contexto global extraído pelos *transformers* (Chen et al. 2021).

O modelo é construído da seguinte forma, a imagem de entrada primeiro passa por uma sequência de camadas convolucionais que vão aprendendo *features* mais profundas. Além disso, em cada camada convolucional, os dados são enviados para a camada correspondente na etapa de expansão através de *skip connections*.

Figura 29 – Arquitetura TransUNet



Fonte: (Chen et al. 2021)

Após as camadas de convoluções iniciais, os dados adaptados sofrem uma transformação que resulta na projeção linear e depois são divididos em *patches* e passados para o *transformer*. A estrutura do *transformer* segue a apresentada pelo ViT, a rede TransUNet aplica 12 dessas camadas e com o resultado é feito um ajuste das dimensões,

seguido de *upsampling* em cascata (CUP). O CUP consiste em uma sequência de blocos de *upsampling* para que a imagem alcance as dimensões originais. Cada bloco é formado por uma operação de *upsampling* de duas vezes, sendo o resultado dessa combinado com os valores recebidos das camadas iniciais por *skip connection*, uma camada de convolução 3 x 3 e a função de ativação ReLu. Todo esse processo é ilustrado na figura 29.

2.6 IA Explicável

IA explicável, ou explainable AI, é um campo de aprendizagem de máquina que busca auxiliar no entendimento de decisões tomadas por modelos de inteligência artificial. Os algoritmos modernos são construídos utilizando uma grande quantidade de parâmetros, podendo chegar a milhões, com um alto nível de complexidade, fazendo que sejam modelos "caixa-preta", tornando a compreensão dos resultados extremamente difícil para seres humanos. Na última década, esses modelos vem sendo utilizados cada vez mais em diferentes áreas da sociedade, com mais impacto e, também, riscos. Dessa forma, o uso de IA explicável é importante para o uso responsável, ético e aderente à legislação desses modelos, especialmente para áreas sensíveis, demostrando que este não apresenta viés e não faz discriminação em relação a gênero, raça, entre outros (Holzinger et al. 2020).

Existem vários métodos de IA explicável utilizados, porém grande parte deles podem se classificados de acordo com dois aspectos, sendo o primeiro se a explicação é local ou global. É considerada local a solução que busca esclarecer os motivos por trás de uma decisão específica do modelo, um dos principais exemplos desse tipo é a atribuição de variáveis (*feature attribution*), que busca determinar quais partes das entradas recebidas pelo modelo são responsáveis pelo resultado final. Para algoritmos de classificação de imagem, essa técnica se dar através de mapas de calor que destacam as áreas mais influentes no resultado. Já as soluções globais são aquelas que explicam comportamentos de componentes dos modelos, como camadas ou ativações. Uma técnica bastante comum dessa classe é a visualização de variáveis (*features visualization*), apresentando imagens, reais ou geradas artificialmente, que mais são impactadas pelos componentes considerados.

A segunda forma de classificar esses métodos é em relação ao momento da explicação em relação ao modelo, podendo ser *ante-hoc* ou *post-hoc*. As técnicas *ante-hoc* são aquelas em que é necessário inserir camadas de explicabilidade na arquitetura do algoritmo em

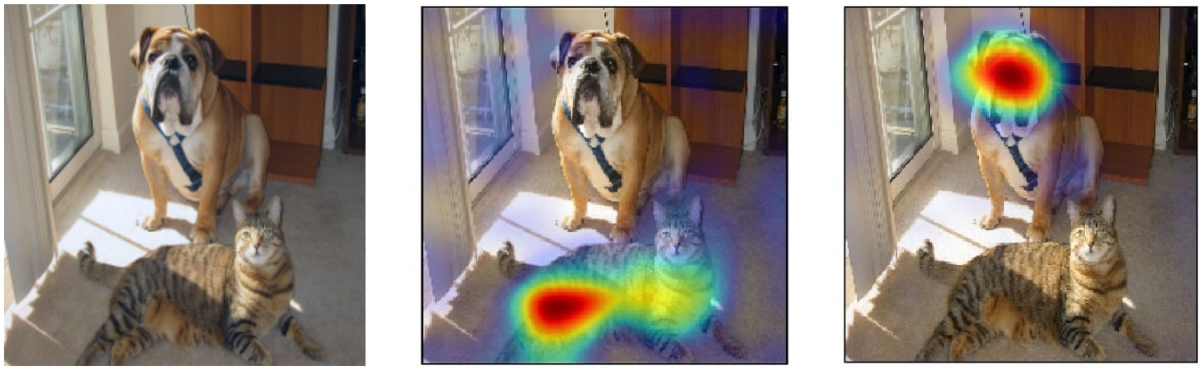
questão antes do treinamento, para que esse modelo retorne além do seu resultado habitual, a explicação para este. Já os métodos *post-hoc*, atuam após todo o processo de treinamento, não sendo necessária alterações na arquitetura.

2.6.1 SegGradCam

SegGradCam, é um método de interpretação dos resultados de um modelo de segmentação de imagem. Essa técnica busca adicionar explicabilidade através de mapas de calor indicando a relevâncias de áreas ou pixels individuais da imagem original para a classe segmentada. O SegGradCam é uma adaptação do método Grad Cam para o uso em problemas de segmentação, sendo esse último uma generalização do Cam, que tem uso limitado a uma arquitetura específica (Vinogradova et al. 2020).

Grad Cam adiciona explicabilidade a tarefas de classificação de imagem através dos gradientes de camadas convolucionais de uma CNN. Para modelos de classificação, as camadas convolucionais possuem a capacidade de reter informações espaciais que são perdidas quando passadas por camadas completamente conectadas no fim da rede, assim, Grad Cam utiliza os gradientes dessas camadas para determinar a importância de um dado neurônio para determinar uma classe específica, como ilustra a figura 30. Esse método pode ser utilizado em qualquer camada convolucional da rede, porém, normalmente é selecionada a última camada, pois essa possui a melhor relação entre semântica de alto nível e informações espaciais. Em relação as categorias descritas anteriormente, Grad Cam é um método que desenvolve explicações locais, apresentando mapas de calor para áreas de interesse, e é *post-hoc*, sendo utilizado após o processo de treinamento dos modelos em questão (Selvaraju et al. 2020).

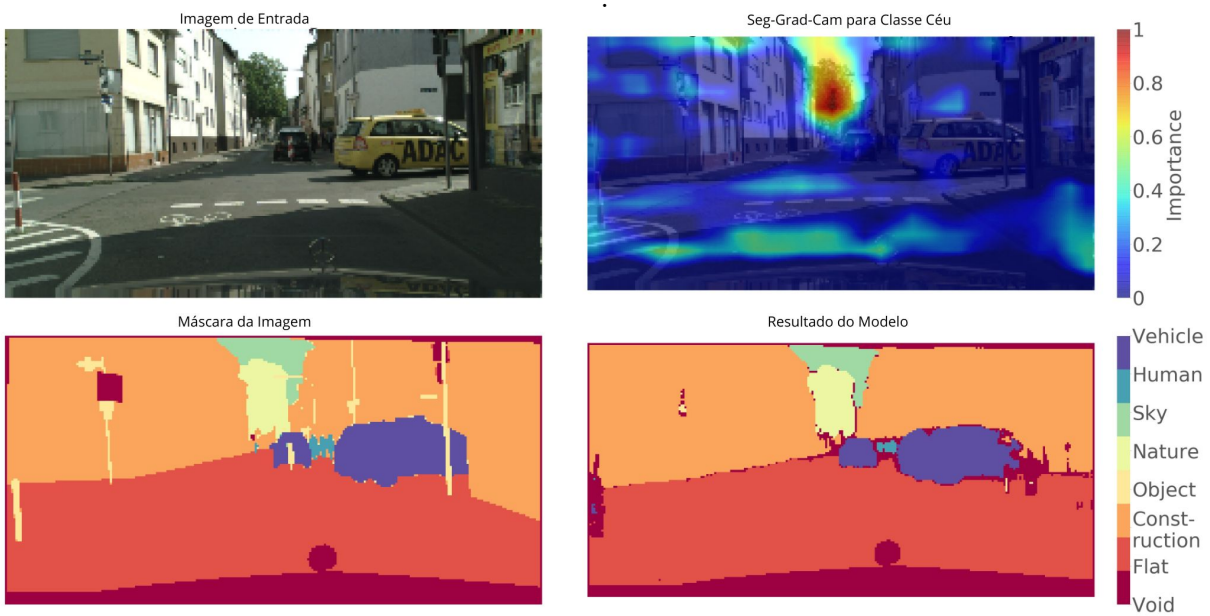
Figura 30 – Exemplo uso de Grad Cam para explicar classificação para as classes gato e cachorro



Fonte: (Selvaraju et al. 2020)

Para a implementação do algoritmo para um problema de segmentação de imagens é necessário algumas modificações, dado que, para classificação, o método busca destacar as regiões que mais influenciaram para o valor atribuído para a classe selecionada pelo modelo. Já para segmentação, o objetivo é destacar todo o mapa para a segmentação de determinada classe, demostrado na figura 31.

Figura 31 – Exemplo uso SegGradCam



Fonte: Adaptado de (Vinogradova et al. 2020)

3 Revisão da Literatura

Segmentação de imagens é uma técnica que possui diversas aplicações na academia, entre elas podemos citar o seu uso na área da biomedicina. Entre os trabalhos desenvolvidos para esse campo se encontram algumas das arquiteturas apresentadas como a UNet (Ronneberger et al. 2015), desenvolvida com base no desafio ISBI para segmentação de estruturas neurais. Outros trabalhos desenvolvidos para problemas similares são 3D U-Net (Çiçek et al. 2016) e V-Net (Milletari et al. 2016).

Outra importante área que utiliza de segmentação de imagem é o desenvolvimento de carros autônomos. Para essa aplicação é necessário que os modelos consigam realizar a separação da imagem em tempo real, dessa forma os modelos são desenvolvidos com ênfase na velocidade. Entre os trabalhos nesse campo estão MobileNetV2 (Sandler et al. 2018), FasterSeg (Chen et al. 2019) e ContextNet (Poudel et al. 2018).

Analisando mais especificamente o uso de modelos de segmentação de imagem para auxiliar atividades de aquacultura 4.0, a maior parte dos trabalhos encontrados tratam do emprego para detecção e identificação de diferentes espécies de peixes. Entre esses estudos podem ser destacados o de (Li et al. 2023) que desenvolveu um modelo para segmentação de imagens de peixes debaixo d'água dividido em três etapas, pré-processamento, *enconder* e *decoder*. (Fernandes et al. 2020) aplicou segmentação semântica para extrair a área do corpo de tilápia-do-nilo, excluindo as barbatanas, para determinar o peso do corpo, o peso da carcaça e qual o rendimento desta. De forma similar, (Munoz-Benavent et al. 2018) experimentou diferentes modelos de segmentação com o objetivo de melhorar a qualidade de sistema de estimação de pesos de atum-rabilho debaixo d'água.

Olhando para os trabalhos voltados para classificação e segmentação de camarão, (Liu et al. 2019) desenvolveu uma rede original voltada para a tarefa de classificação de diferentes espécies camarão chamada ShrimpNet. Esta rede é baseada na estrutura da LeNet-5, contando com duas camadas de convolução iniciais, que sintetizam informações estruturais complexas, essa etapa é seguida de uma camada *max pooling*, capturando partes deformadas e reduzindo as dimensões da imagem. Após, o resultado da operação anterior é passado por duas camadas completamente conectadas e pela camada de combinação, que recebe esse nome pois agrega três tipos de classificadores, Support Vector Machine (SVM), SoftMax e Random Forest, para aprimorar o resultado. Por fim, a rede retorna a espécie

de camarão obtida pelas operações descritas. As principais contribuições da ShrimpNet são: o treinamento em paralelo em três unidades de GPU, adição de filtros 3×3 e 1×1 (baseados no design da SqueezeNet (Iandola et al. 2016)), a junção de diferentes *strides* de convoluções nessas camadas, a combinação de três classificadores diferentes, ajuste da rede para encontrar a estrutura ótima e incorporação de técnicas de *dropout*. O modelo foi avaliado em uma base com nove espécies diferentes de camarão, obtendo uma acurácia de 96,84%. Apesar do bom resultado, sendo a diferenciação entre espécies de camarão o objetivo do modelo, este não é aplicável a presente pesquisa, sendo utilizada nesta apenas uma espécie.

(Poonnoy e Asavasanti 2021) propõem um modelo que utiliza a combinação de reconhecimento de padrões com regressão, através de uma rede neural artificial, para extrair a área e perímetro de imagens bidimensionais de camarões em variadas posições e, utilizando essas informações, estimar a massa do animal. A base de imagens utilizada foi construída com camarões obtidos em um supermercado na Tailândia e fotografados pelos autores. Para a estimação do peso, primeiro, foi empregado um script de MATLAB para calcular a área e o perímetro do animal. Após, os valores foram passados para um modelo *Pattern Recognition Artificial Neural Network* (P-ANN) de forma a determinar a o grau de semelhança entre a imagem avaliada e cada uma das quatro posturas principais previamente estabelecidas. Finalmente, a combinação de todas essas informações é inserida em um modelo *Enhanced Regression Artificial Neural Network* (E-ANN) que estimará a massa do camarão.

(Koushik et al. 2021) utilizou do modelo Faster R-CNN, de segmentação de instância, para identificar imagens de camarão (*prawn*), desenhar uma caixa delimitadora ao redor do perímetro do animal, e, por fim, extrair o tamanho a partir dos limites encontrados. Com as informações de comprimento, é estimado o peso utilizando a fórmula 3.1, onde W representa o peso e L o comprimento. Para o desenvolvimento do modelo foi construída uma base de dados com imagens de camarão e de outros animais coletadas do Google Images, as imagens foram agrupadas em duas classes distintas, camarão e não camarão. Para obter as caixas delimitadoras para o treinamento da rede foi utilizada a ferramenta LabelImg.

$$W = 0,0108 * L^2.6978 \tag{3.1}$$

Outro trabalho que buscou estimar o peso de camarões através de técnicas de segmentação de imagens foi (Setiawan et al. 2022). Com o objetivo de obter essa medida para imagens de camarão debaixo de água, foi utilizada uma câmera submersa filmando ao longo de cinco minutos. *Frames* foram extraídos do vídeo completo e convertidos de RGB para *grayscale* e depois para imagens binárias com o emprego do algoritmo de Otsu. Após essa etapa, as bordas foram traçadas com o uso *Canny Edge Detector*. Para o calculo do peso, foram utilizados diferentes algoritmos de regressão, entre eles Regressão Linear Múltipla (MLR), Support Vector Machines (SVM) e Back Propagation Neural Networks (BPNN).

Similar ao trabalho de (Setiawan et al. 2022), (Lai et al. 2022) desenvolveu um sistema de vídeo para fotografar camarões debaixo de água, no fundo do tanque de aquicultura, durante sua alimentação, com uma câmera fixa. Devido ao uso câmeras com lentes amplas, foi necessário a calibragem das imagens, para evitar distorções. Para tanto, foi utilizando um tabuleiro de xadrez como referência, com esse processo, também foi possível obter a relação pixel-comprimento para uma distância fixa. As figuras foram capturadas nos formatos RGB, *grayscale* e infravermelho e depois classificadas em visíveis ou não, e mensuráveis ou não, com apenas as ocorrências mensuráveis sendo utilizadas para o experimento. O algoritmo YOLOv4-tiny foi empregado pelos autores para a detecção dos animais nas figuras, onde esse desenha uma caixa delimitadora ao redor da área de interesse, separando individualmente cada camarão. Após, utilizado o algoritmo de Otsu para transformação dos recortes de cada animal obtidos na etapa anterior em imagens binárias. Por fim, é desenhada uma caixa delimitadora contendo todos os pixels do camarão, com a mínima área possível, através da transformação Hotelling. Dessa forma, o tamanho do animal é determinado como igual a dimensão da maior aresta da caixa.

(Zhou et al. 2023) buscou mitigar a dependência de dados rotulados para o desenvolvimento de modelo de segmentação semântica para camarão. Para tanto foi aplicado *contrastive learning* para o treinamento de um modelo ResNet-50 de forma a obter a representação de camarões utilizando dados não rotulados. Finalizado o treinamento desse modelo, este substitui a sua contraparte contida na arquitetura do Mask R-CNN e passa por um novo treinamento, dessa vez com dados rotulados.

Nos trabalhos apresentados, foi observada a ausência de metodologias experimentais bem estruturadas e replicáveis, comprometendo a comparação entre diferentes abordagens

e a reprodutibilidade dos resultados. Além disso, não foi realizada uma exploração de diferentes métodos para segmentação de imagem, sendo geralmente utilizada apenas uma técnica para cada etapa do processo.

A tabela 1 apresenta uma comparação resumida entre os trabalhos apresentados nessa seção. Os aspectos detalhados nessa são o número de amostras utilizados (coluna 2), o problema proposto pelo trabalho (coluna 3) e os modelos aplicados na investigação (coluna 4).

Referência	Número de amostras	Problema	Modelo
Liu, Jia and Xu	1731	Classificação de Camarão	ShrimpNet
Poonnoy and Asavasanti	103	Estimar massa e comprimento de camarões em diversas posturas	P-ANN e E-ANN
Koushik et al.	500	Segmentação e extração do comprimento	Faster R-CNN
Setiawan, Hadiyanto and Widodo	20	Estimar o peso a partir de imagem submersas	Algoritmo de Otsu e Canny edge Detector
Lai et al.	3109	Sistema de vídeo submerso para obter imagens durante a alimentação	YOLOv4-tiny, Algoritmo de Otsu e transformação Hotelling
Zhou et al.	10000	Mitigar a dependência de dados rotulados para segmentação de camarão	Mask R-CNN pré-treinado com ResNet-50
Proposta	2600	Aplicação de diferentes arquiteturas de classificação e segmentação para imagens de camarão e o uso de técnicas de IA explicável para análise qualitativa da segmentação	VGGNet, ResNet, GoogLeNet ,FCN, UNet, LinkNet, DeepLab-v3, Attention Unet e TransUNet

Tabela 1 – Comparação Entre os Trabalhos Relacionados

4 Metodologia

Este capítulo descreve, de maneira sistemática, os procedimentos metodológicos adotados para o desenvolvimento e a avaliação dos modelos propostos neste trabalho. A abordagem contempla desde a seleção e preparação da base de dados até os experimentos realizados com os algoritmos de segmentação e classificação de imagens.

Inicialmente, apresenta-se a base de dados utilizada, com destaque para suas características, justificativas para a escolha e os métodos empregados para o pré-processamento das imagens. Em seguida, são detalhados os algoritmos desenvolvidos, divididos entre aqueles voltados à tarefa de segmentação e classificação de imagens. A escolha das arquiteturas foi fundamentada em sua relevância na literatura e na diversidade de abordagens técnicas que oferecem, como redes convolucionais profundas, mecanismos de atenção e *transformers*.

Posteriormente, são discutidos os procedimentos empregados para o treinamento dos modelos e a configuração dos hiperparâmetros. Além disso, também são detalhados os métodos utilizados para a avaliação dos modelos, com a definição das métricas adotadas. Por fim, é descrito o ambiente computacional no qual os experimentos foram executados, incluindo os recursos de *hardware* e as bibliotecas de *software* utilizadas.

A organização deste capítulo visa assegurar a reprodutibilidade dos experimentos e a transparência metodológica, permitindo uma análise crítica e fundamentada dos resultados apresentados nos capítulos subsequentes.

4.1 Base de Dados

Para o treinamento e avaliação dos algoritmos selecionados para as tarefas de classificação e segmentação de imagens de camarão e peixe, foi utilizado a base de dados *A Large Scale Fish Dataset*, disponível na URL [1]. A base de dados contem 8.000 imagens (*augmented*), com dimensões espaciais originais de 590 x 445 pixels, de 8 espécies diferentes de peixes e mais 1.000 de camarão, totalizando 9.000 imagens, sendo 1.000 para cada espécie. As fotografias foram tiradas de animais adquiridos em um supermercado na cidade de Izmir na Turquia. Além das imagens originais, a base de dados também possui a máscara, área

[1] https://www.kaggle.com/datasets/crowww/a-large-scale-fish-dataset

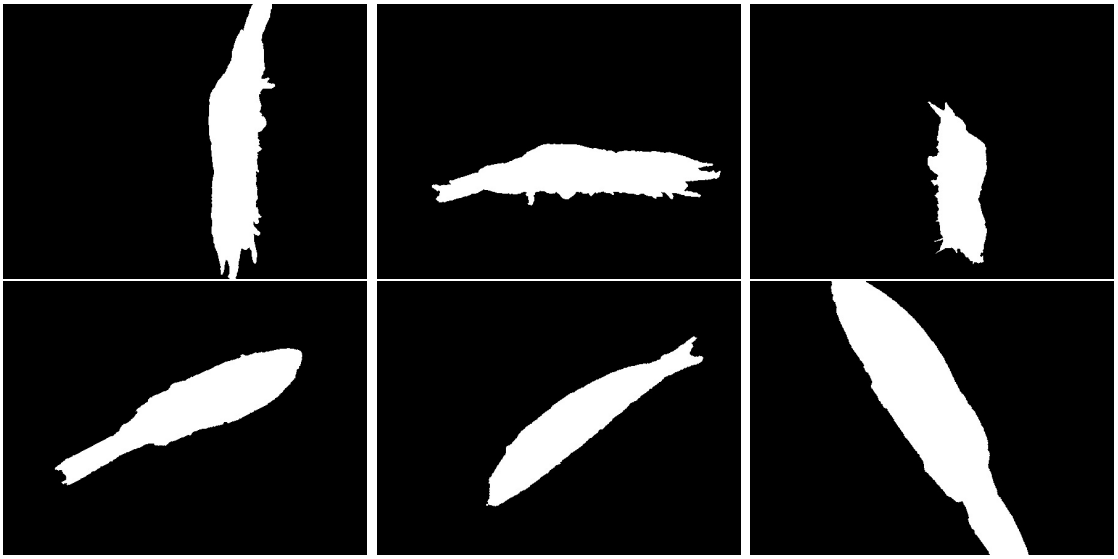
segmentada do animal, correspondente para cada uma. A Figura 32 apresenta exemplos de imagens contidas na base de dados, com as suas máscaras correspondentes disponíveis na Figura 33.

A escolha da base foi realizada devido a quantidade de imagens de camarão, e outros animais, com sua respectiva máscara, permitindo o treinamento dos modelos de segmentação semântica escolhidos. Além disso, como apresenta uma variedade de espécies, a base permite o desenvolvimento de modelos de classificação binária, com o objetivo de determinar se uma dada imagem representa o animal de interesse, o camarão ou qualquer outra espécie. A seleção de imagens foi amplamente utilizada em diferentes pesquisas, (Aziz et al. 2023) a utilizou para o desenvolvimento de uma arquitetura dedicada a classificação de diferentes espécies de peixes. De forma semelhante, (Malik et al. 2023) desenvolveu um modelo de detecção e classificação baseado no algoritmo YOLOv7. Já (Wang et al. 2024), utilizou essa base em conjuntos com outras, de fins diversos, para testar a adaptação do Segment Anything Model (SAM) para diferentes tipos de imagens.

Figura 32 – Imagens de camarão e peixe

Fonte: (Ulucan et al. 2020)

Figura 33 – Máscaras das imagens de camarão e peixe



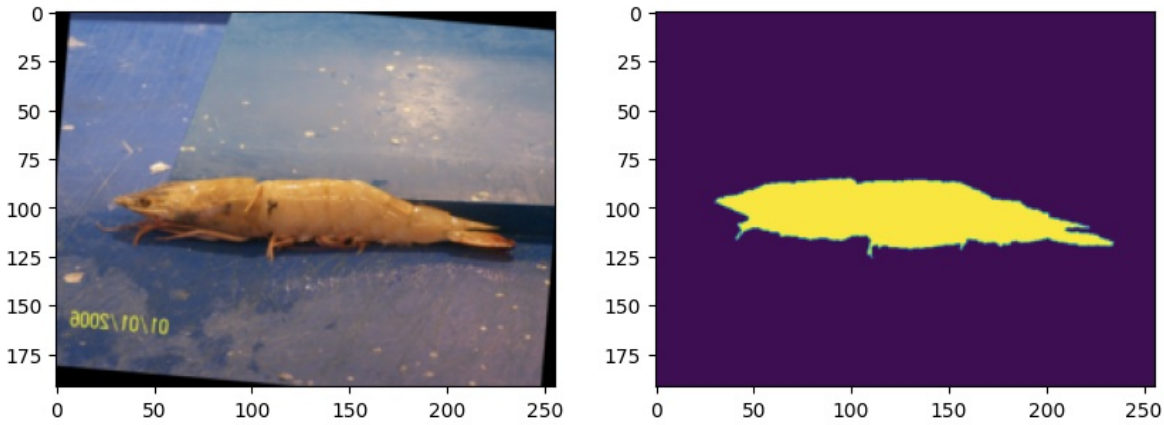
Fonte: (Ulucan et al. 2020)

Considerando que o foco deste trabalho é o uso das técnicas de segmentação e classificação para auxiliar no cultivo de camarão, foram selecionadas da base apresentada as 1.000 imagens correspondente a esse animal. Além dessas, também foi selecionada uma parcela de 20% das demais espécies, com 200 imagens para cada uma das classes. Além das imagens, também foram utilizadas as suas máscaras correspondentes. Ao todos foram utilizadas 2.600 imagens retiradas da base de dados.

Para o desenvolvimento dos modelos, as imagens selecionadas foram divididas em 3 grupos: treinamento, validação e teste. A divisão foi realizada de maneira aleatória, apenas mantendo a proporção de imagens por especies encontrada no conjunto completo. Dessa forma, as 2.600 imagens foram divididas seguindo os seguintes percentuais em relação ao total, 60% (1.560) para treinamento, 20% (520) para validação e 20% (520) para teste.

Além dos recortes descritos, também foram realizadas manipulações nas imagens, utilizando a biblioteca OpenCV (cv2), para melhor adequá-las para o uso nos modelos a serem desenvolvidos. Primeiro, foi feito uma operação de resize, onde as dimensões espaciais das imagens foram alteradas de 590×445 para 256×192 pixels, mantendo a proporção original. Após, o esquema de cor foi convertido de BGR para RGB para as imagens e binarização para as máscaras, onde qualquer pixel com valor superior a 0 foi alterado para 1. A Figura 34 demonstra uma imagem, e sua máscara correspondente, após as transformações descritas.

Figura 34 – Imagem Manipulada de Camarão e Máscara



Fonte: O Autor

4.2 Algoritmos

Para o desenvolvimento dos modelos foi utilizada uma série de algoritmos que podem ser divididos em dois grupos: segmentação e classificação.

Para a implementação de modelos de segmentação de imagens, foram selecionados algoritmos estabelecidos na academia, que se utilizam de diferentes técnicas em sua composição, como *skip connections*, blocos *enconder* e *decoder*, convolução dilatada, *attention gates* e *transformers*. Essa seleção de algoritmos foi escolhida para fornecer uma ampla gama de cenários para o teste da adaptação de diferentes técnicas para o problema proposto. Ao fim, os algoritmos implementados são: FCN (Long et al. 2015), UNet (Ronneberger et al. 2015), LinkNet (Chaurasia e Culurciello 2017), DeepLabv3 (Chen et al. 2017), Attention Unet (Oktay et al. 2018) e TransUnet (Chen et al. 2021). O detalhamento dessas arquiteturas foi descrito na seção 2.5.1.

Para a tarefa de classificação de imagem, o objetivo é realizar uma classificação binária, onde será identificado se uma dada imagem tem a presença de camarão ou não. Dessa forma, sendo um problema mais amplamente estudado, e se tratando de uma tarefa mais simples quando comparada a segmentação, foram selecionadas três arquiteturas amplamente utilizadas e estabelecidas na acadêmia. São elas: VGGNet (Simonyan e Zisserman 2014), ResNet (He et al. 2016) e GoogLeNet (Szegedy et al. 2015).

4.3 Configuração de Experimentação

4.3.1 Configuração dos Modelos de Segmentação de Imagem

Para o treinamento dos diferentes modelos de segmentação de imagem citados anteriormente, foi utilizada a seguinte metodologia. Cada experimento foi executado por 20 *epochs*, com *batch size* de 32, com um total de 49 *batches*. O *learning rate* utilizado foi de $1e^3$, com a função de perda `nn.BCEWithLogitsLoss`, implementada na biblioteca PyTorch, sua equação é descrita na equação 4.1, onde N representa o *batch size*. Para cada um dos algortimos, foram realizadas múltiplas execuções independentes, buscando fornecer robustez e confiabilidade aos resultados obtidos através de testes de significância estatística. Para tanto, a quantidade de execuções estabelecida foi 30, almejando um equilíbrio entre a confiabilidade que maiores amostras possibilitam com o custo computacional de execuções extras. Por fim foi obtida a média das métricas resultantes. Todos os parâmetros descritos são listados na tabela 2. Para os demais hiperparâmetros dos diversos modelos que não foram citados, foram utilizados seus valores padrões.

$$\ell(x, y) = \frac{1}{N} \sum_{n=1}^{N} \ell_n \tag{4.1}$$

$$\ell_n = -w_n \left[y_n \cdot \log(\sigma(x_n)) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right]$$

Parâmetro	Valor
Execuções	30
Epochs	20
Batch Size	32
Learning Rate	1e-3
Valor Limite	0.5
Função de Perda	BCEWithLogitsLoss

Tabela 2 – Parâmetros de Treinamento Modelos de Segmentação

4.3.2 Configuração dos Modelos de classificação de Imagem

Diferente dos modelos de segmentação de imagem, os algoritmos selecionados para classificação foram implementados com o uso de *transfer learning*, sendo utilizado o modelo pré-treinado completo, fornecido pela biblioteca PyTorch, apenas sendo substituída a última camada de cada um para se adequar a classificação entre camarão e não camarão. Após essa modificação, o treinamento dos modelos de classificação seguiram os mesmos princípios utilizados para segmentação, com diferenças apenas para a função de perda e as métricas de avaliação.

Como função de perda, foi utilizada a implementação da função `nn.CrossEntropyLoss` da biblioteca PyTorch, a fórmula dessa função é apresentada na equação 4.2.

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{4.2}$$

A tabela 3 apresenta todos os paramêtros especificados para o treinamento desses modelos, com os demais hiperparâmetros não citados sendo utilizados seus valores padrão.

Parâmetro	Valor
Execuções	30
Epochs	20
Batch Size	32
Learning Rate	1e-3
Função de Perda	Cross Entropia Cruzada

Tabela 3 – Parâmetros de Treinamento Modelos de Classificação

4.4 Avaliação Experimental

4.4.1 Avaliação dos Modelos de Segmentação de Imagem

Para a avaliação dos algoritmos empregados para segmentação foi definido um valor limite, onde cada pixel do resultado do modelo é comparado com esse, sendo classificado

como pertencente a classe os que apresentarem valor superior ao limite, com os demais transformados em nulo. Essa transformação foi utilizada para aproximar o formato da máscara retornada pelo modelo com a máscara original, já que esta apresenta valores binários para os pixels e os valores resultantes da rede são contínuos variando de 0 até 1. Dessa forma, a máscara resultante é binarizada, processo necessário para o cálculo da métrica Intersection over Union (IoU), sendo utilizado o valor de 0,5 como limite, encontrado através de análises oculares dos resultados obtidos para diferentes valores de limite. A métrica IoU, mensura a área de interseção entre duas imagens, dividida pela área da união destas, dessa forma, é uma medida da proporção de pixels com valor da primeira imagem que possuem correpondencia na segunda, tendo seu valor aumentado nos casos em que um pixel contido na máscara seja incluido na segmentação, já, se um pixel da máscara não esteja contido no resultado do modelos, ou se neste estejam pixels adicionais, o valor da métrica é reduzido. A equação 4.3 apresenta a fórmula para o cálculo da métrica IoU, onde A representa o resultado do modelo e B a máscara real da imagem.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{4.3}$$

4.4.2 Avaliação dos Modelos de Classificação de Imagem

Para a avaliação da performance dos modelos, foram analisadas duas métricas, a acurácia e o F1-Score. As fórmulas das métricas se encontram nas equações 4.4 e 4.5, respectivamente, onde TP representa o verdadeiro positivo, TN o verdadeiro negativo, FP o falso positivo e FN o falso negativo.

$$\text{Acurácia} = \frac{\text{Quantidade de Previsões Corretas}}{\text{Quantidade Total de Previsões}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

$$\text{F1 Score} = \frac{2TP}{2TP + FP + FN} \tag{4.5}$$

4.5 Ambiente de Execução

Os experimentos foram realizados utilizando *Jupyter Notebooks* em um *desktop* com 16GB de memória RAM, processador AMD Ryzen 5 5600G *with Radeon Graphics* e uma placa de vídeo dedicada NVIDIA GeForce RTX 3060. O código dos experimentos foi escrito na linguagem Python, utilizando as seguintes bibliotecas: PyTorch[2], Pandas[3], OpenCV[4] e NumPy[5]

[2] https://pytorch.org/
[3] https://pandas.pydata.org/
[4] https://opencv.org/
[5] https://numpy.org/

5 Resultados

Neste capítulo, são apresentados e analisados os resultados obtidos a partir da aplicação dos modelos de segmentação e classificação propostos, conforme metodologia descrita no capítulo 4. Inicialmente, será abordada a avaliação quantitativa dos algoritmos de segmentação, considerando métricas como Intersection over Union (IoU), tempo de execução, número de parâmetros e tamanho dos modelos. Em seguida, realiza-se uma análise qualitativa dos mapas de segmentação gerados, com o suporte da técnica SegGradCam, buscando compreender o comportamento dos modelos diante de diferentes imagens.

Após, são apresentados os resultados referentes à tarefa de classificação, com a avaliação dos modelos baseando-se nas métricas de acurácia e F1-Score. Também são analisados aspectos como eficiência computacional, tamanho dos modelos e tempo de inferência.

Em ambas as tarefas, são aplicados testes estatísticos, como o teste de Friedman seguido do *post-hoc* de Nemenyi, para verificar a significância das diferenças entre os algoritmos.

5.1 Avaliação dos Modelos de Segmentação

5.1.1 Análise Quantitativa

Todos os algoritmos utilizados foram avaliados em relação a sua performance de segmentação, complexidade e tamanho, tempo para treinamento e teste. A Tabela 4 apresenta os resultados alcançados por cada um dos modelos de segmentação para os aspectos citados. A segunda coluna mostra o valor médio da métrica IoU para as 30 execuções do respectivo algoritmo. A terceira coluna apresenta a quantidade total de parâmetros treináveis de cada um dos modelos, já na quarta coluna está o tamanho em MB ocupado em memória pelo modelo no formato *pkl*. Por fim, a quinta e a sexta colunas possuem os valores tempo médio necessário para o treinamento, em segundos, e o tempo médio para avaliação de uma imagem durante o processo de teste, em milissegundos, respectivamente, utilizando os dados descritos no capítulo 4.

Algorithms	IoU Médio	Número de Parâmetros (M)	Tamanho do Arquivo (MB)	Tempo de Treinamento (s)	Tempo de Teste (ms)
FCN	84,33	18,6	71,1	906	26,2
UNet	74,71	0,1	0,5	609	25,4
LinkNet	86,92	11,9	45,5	584	25,0
DeepLab-V3	88,67	39,6	151,5	1319	30,4
Attention UNet	87,21	2,6	10,1	658	24,4
Trans UNet	85,40	105,2	401,6	1285	29,6

Tabela 4 – Análise Comparativa de Performance.

Analisando os resultados obtidos, é possível identificar que o modelo DeepLab-V3 obteve a melhor performance para a métrica IoU, seguido pelos algoritmos Attention UNet e LinkNet.

Em relação as demais medidas analisadas, temos uma inversão nos resultados, com o UNet sendo a arquitetura mais leve, possuindo o menor número de parâmetros treináveis e o menor tamanho médio para os seus arquivos, esse modelo também obteve um bom resultado em relação a sua velocidade, apresentando o segundo menor tempo para treinamento e terceiro menor para avaliação, porém essas medidas são obtidas com um custo para a métrica IoU, com o modelo apresentando valores significantemente abaixo dos demais desenvolvidos, com valor médio mais de dez pontos percentuais abaixo do desempenho do segundo menor resultado. Outros algoritmos que se destacaram positivamente foram o Attention Unet e LinkNet, em especial o primeiro, sendo o único próximo do UNet em relação ao número de parâmetros e tamanho do arquivo, e apresentando o menor tempo necessário para avaliação. Por outro lado, o modelo DeepLab-V3 foi o mais lento tanto para o treinamento quanto para o teste, esse resultado pode ser explicado pela complexidade do modelo que possui o segundo maior número de parâmetros treinaveis com 39,6 milhões.

Para complementar análise quantitativa dos resultados encontrados para os modelos desenvolvidos, foi realizado um teste de significância estatística, sendo a hipótese nula é que as médias dos valores da métrica IoU para os seis algoritmos não são estatísticamente diferentes. Para tanto, foi empregado o teste não paramétrico de Friedman, o nível de

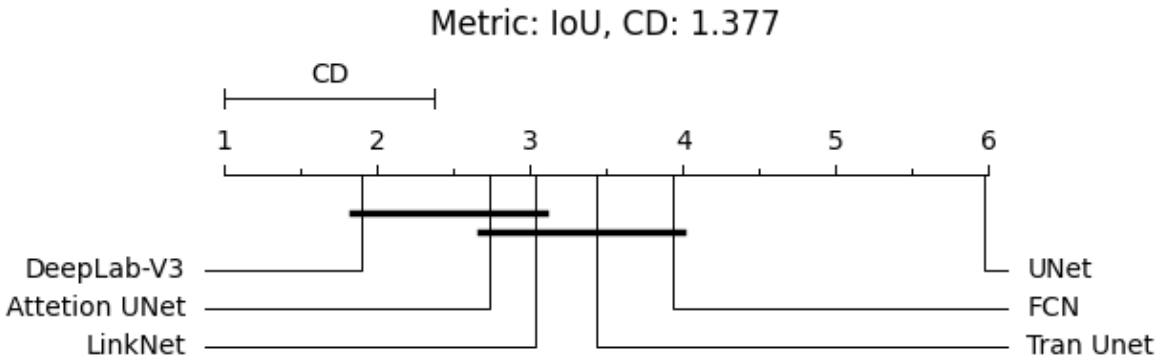
significância alfa utilizado foi 0,05. Com isso a hipótese nula foi rejeitada, com valor-p de $2,3 \times 10^{-16}$, pois foi identificado que pelo menos um dos resultados é estatísticamente diferentes de outro. Para além dessa análise, também foi realizado o teste *post-hoc* de Nemenyi para identificar os grupos de modelos que possuem semelhanças estatística entre os integrantes. Esse teste calcula o *ranking* médio de cada um dos algoritmos, como também o valor de distância crítica (CD), onde resultados com diferença menor que esse são considerados estatisticamente semelhantes. A Tabela 5 apresenta os rankings médios de cada um dos modelos desenvolvidos, já a Figura 35 ilustra a comparação desses resultados destacando os que são estatisticamente semelhantes.

Algoritmo	Ranking
DeepLab-V3	1,90
Attention UNet	2,73
LinkNet	3,03
Trans UNet	3,43
FCN	3,93
UNet	5,97

Tabela 5 – Ranking Médio.

Como esperado pelos resultados encontrados na Tabela 4, o modelo DeepLab-v3 apresentou o melhor valor entre os algoritmos desenvolvidos, com ranking médio de 1,9, ou seja, considerando os 30 experimentos feitos para cada modelo, quando comparado com os demais o DeepLab-V3, fica em média ligeiramente acima da segunda posição, mesmo apresentando a melhor performance, o valor médio do ranking indica que esse modelo não é consistentemente o melhor, o que sugere um certo equilibrio entre os algoritmos. Por outro lado, com o rank médio de 5,97, a arquitetura UNet constantemente apresentou o pior resultado quando comparada com as demais.

Figura 35 – Diagrama de Diferença Crítica para o teste estatístico Friedman-Nemenyi.

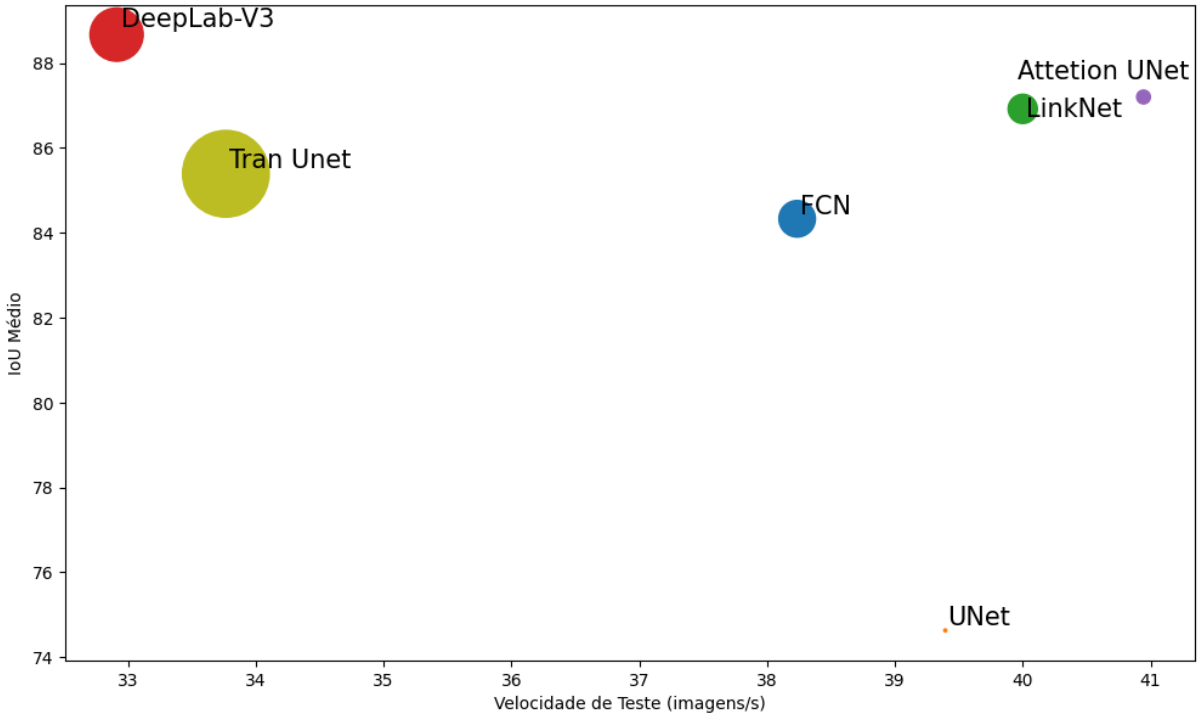


Fonte: O Autor.

Analisando o diagrama apresentado na Figura 35, temos que o valor da distância crítica calculada é de 1,377, ou seja, modelos que possuem diferença de rank médio inferior a esse valor são consideradas estatísticamente semelhantes, essa relação é representada no diagrama por uma linha horizontal atravessando os elementos do grupo. Dessa forma, o resultado obtido para o modelo DeepLab-v3 é estatísticamente semelhante aos inferidos para os modelos Attention Unet e LinkNet, que por sua vez também são estatísticamente semelhantes as arquiteturas Trans Unet e FCN. Já para o algoritmo UNet, foi encontrado que este possui uma distancia em relação aos demais superior ao valor crítico, tendo resultados diferentes de todos os outros.

Para uma análise mais completa das métricas descritas na Tabela 4, foi desenvolvido um gráfico de dispersão apresentado na Figura 36, onde o eixo X representa a velocidade de processamento de imagens para teste, em imagens por segundo, valor obtido a partir da divisão de 1000 pelo valor do tempo de teste de cada modelo, essa transformação foi realizada para melhorar o a visualização do gráfico, posicionando os algoritmos mais rápidos do lado direito, ao invés do lado esquerdo caso fosse o utilizado o tempo decorrido. Já o eixo Y indica o valor médio da métrica IoU. Por fim, o tamanho do círculo repressentando cada um dos modelos é proporcional à quantidade de parâmetros treináveis deste.

Figura 36 – Comparação de Performance dos Modelos Desenvolvidos.



Fonte: O Autor.

Explorando o total de informações obtidas para os algoritmos desenvolvidos não é possível afirmar que existe um com performance superior a todos os demais, pois, mesmo obtendo o melhor *rank* para métrica IoU, pelo teste de Nemenyi, não podemos afirmar que o modelo DeepLab-V3 é estatísticamente diferente do Attention UNet e LinkNet, que por sua vez se mostraram mais leve e rápido em relação a esse. Dessa forma, esses dois algoritmos, analisando pelas métricas apresentadas no gráfico da Figura 36, se mostraram superiores aos demais, sendo os dois modelos com melhores valores para a métrica IoU, e, apesar do modelo LinkNet ser estatisticamente semelhante aos dois, este apresentou valores inferiores em relação ao Attention UNet para as demais métricas.

5.1.2 Análise Qualitativa

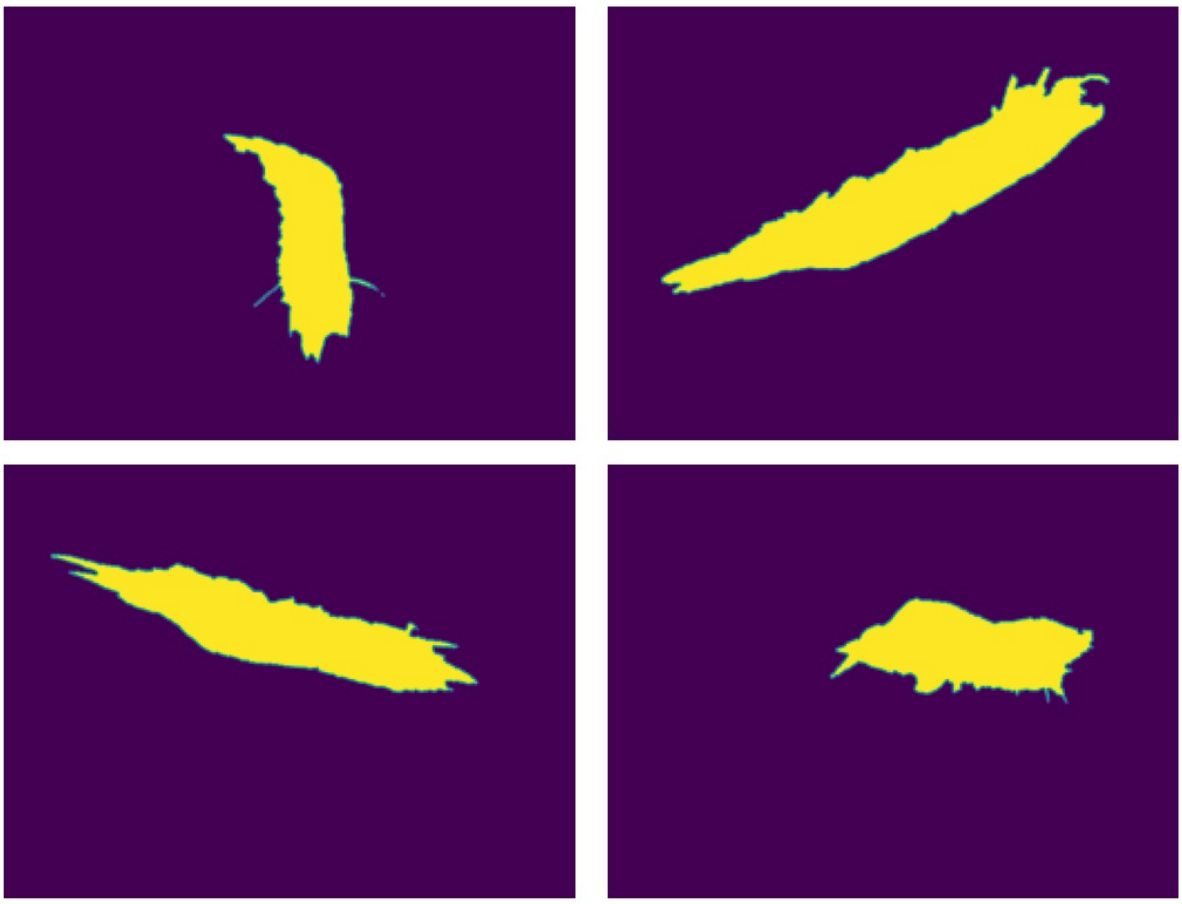
Para a análise qualitativa dos modelos de segmentação desenvolvidos foram selecionadas quatro imagens de camarão da base de teste, apresentadas na Figura 37, com suas respectivas máscaras na Figura 38, e para cada uma dessas foi aplicado o método SegGradCam aos resultados de cada um dos algortimos utilizados. Para seleção de qual

execução entre as trintas realizadas, foi escolhida a que possui valor de IoU mais próximo da média para cada uma das arquiteturas. Para cada um dos modelos será ilustrado o resultado da segmentação e a explicação obtida pelo SegGradCam considerando as imagens de teste apresentadas.

Figura 37 – Imagens de Camarão Selecionadas Aleatoriamente 1, 2, 3 e 4.

Fonte: (Ulucan et al. 2020).

Figura 38 – Máscaras das Imagens de Camarão Selecionadas Aleatoriamente 1, 2, 3 e 4.

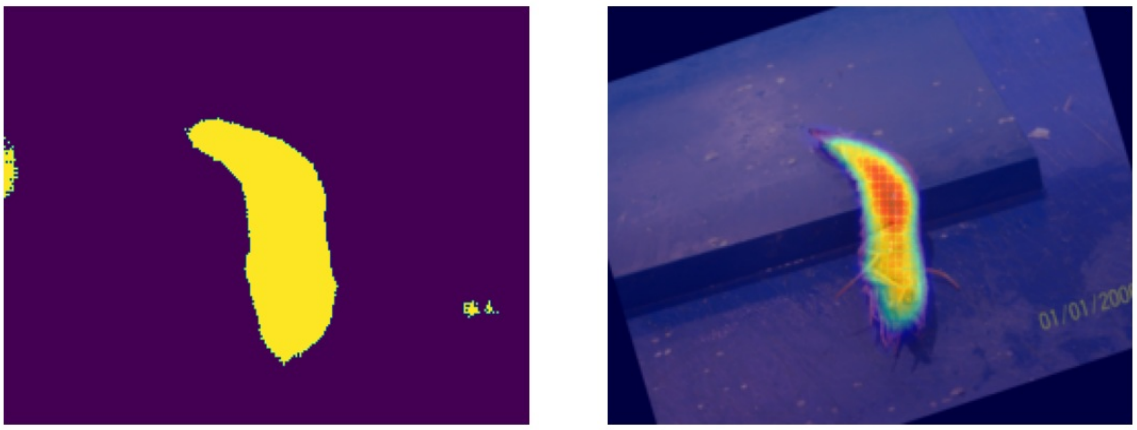


Fonte: (Ulucan et al. 2020).

5.1.2.1 FCN

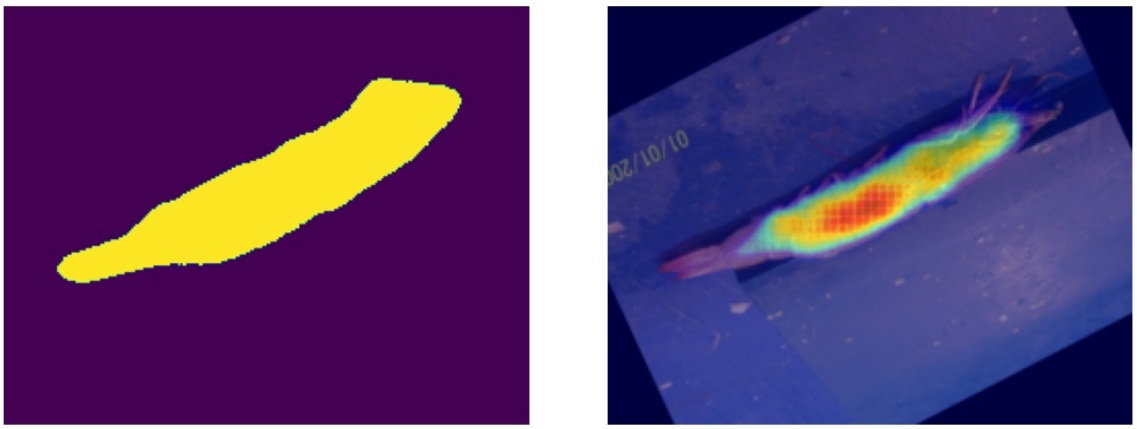
Os resultados do modelo FCN para os exemplos 1, 2, 3 e 4, são apresentados nas Figuras 39, 40, 41 e 42, respectivamente. Analisando as imagens geradas é observado que o modelo possui resultados mais intensos para a área central do corpo do camarão que vai enfraquecendo a medida que se aproxima das extremidades. Outro ponto a destacar é o que aparenta ser um comportamento de agrupamento de alguns pixels em blocos, com pequenas linhas entre esses, se assemelhando a um *grid*.

Figura 39 – Segmentação e Explicação do exemplo 1 para o modelo FCN.



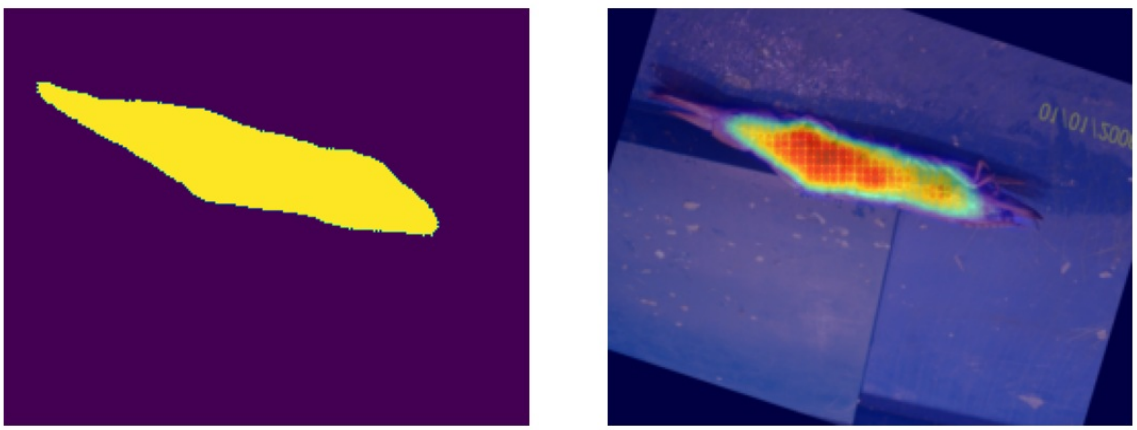
Fonte: O Autor.

Figura 40 – Segmentação e Explicação do exemplo 2 para o modelo FCN.



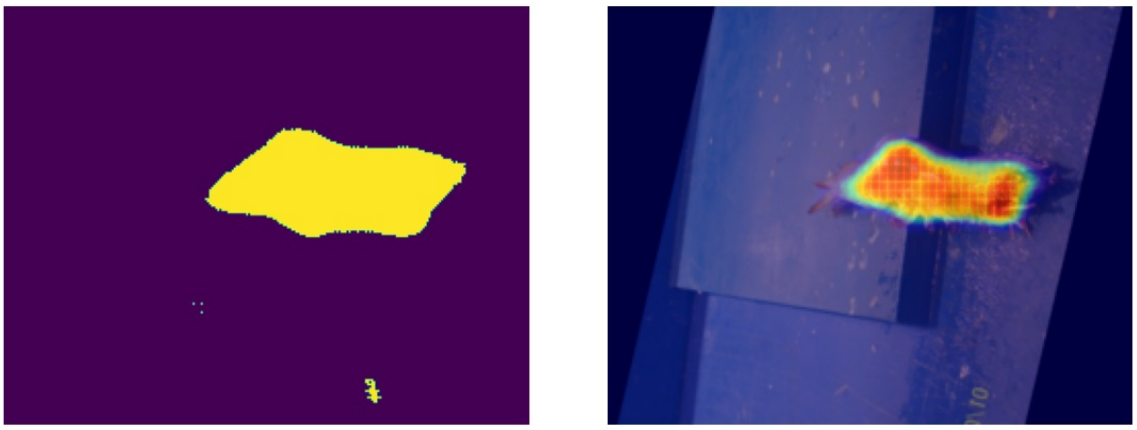
Fonte: O Autor.

Figura 41 – Segmentação e Explicação do exemplo 3 para o modelo FCN.



Fonte: O Autor.

Figura 42 – Segmentação e Explicação do exemplo 4 para o modelo FCN.

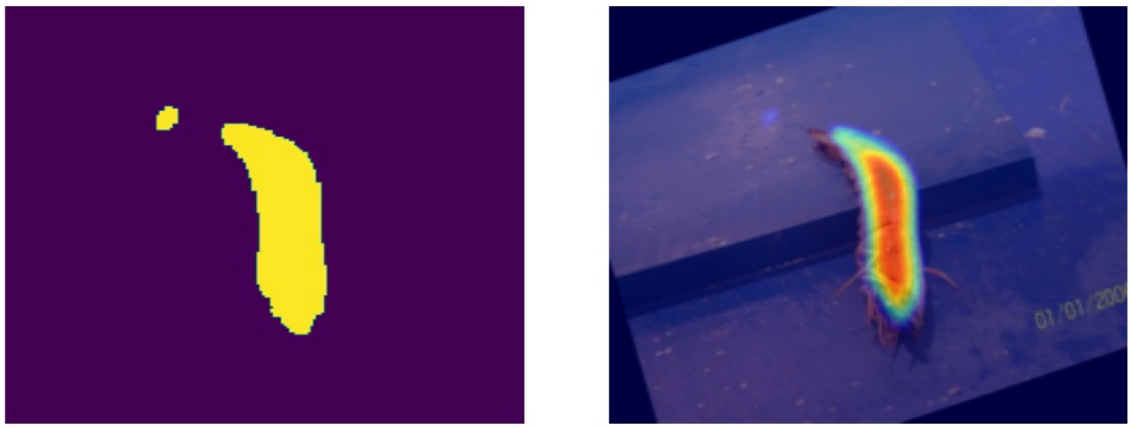


Fonte: O Autor.

5.1.2.2 UNet

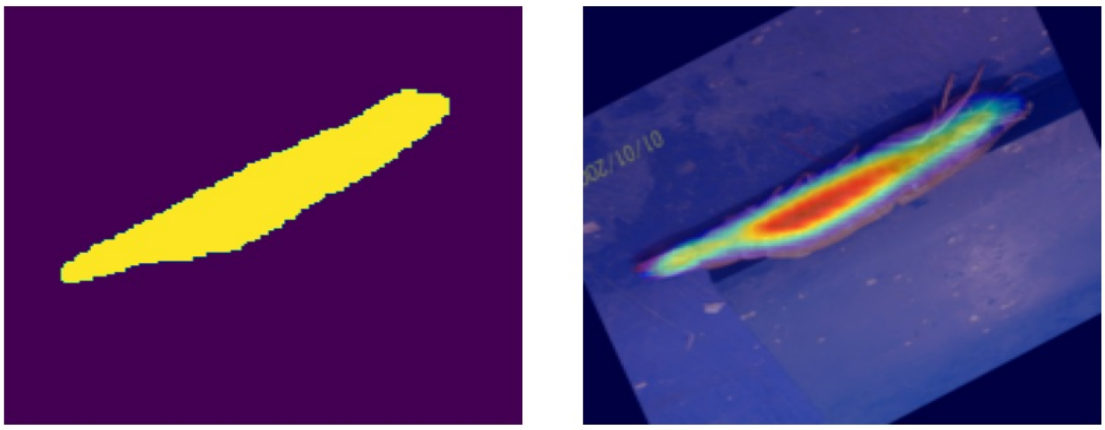
O mapa de segmentação para o modelo UNet apresenta alguns *insights* para o motivo da sua performance ser abaixo dos demais. Similar ao modelo FCN, esse algoritmo identifica com valores mais elevados a área central do corpo do camarão, porém não possui bom desempenho para delimitar as bordas desse, em especial a parte superior do animal, com essa parte recebendo menos atenção do modelo nas Figuras 44 e 45 do que ruídos contidos nas Figuras 43 e 46.

Figura 43 – Segmentação e Explicação do exemplo 1 para o modelo Unet.



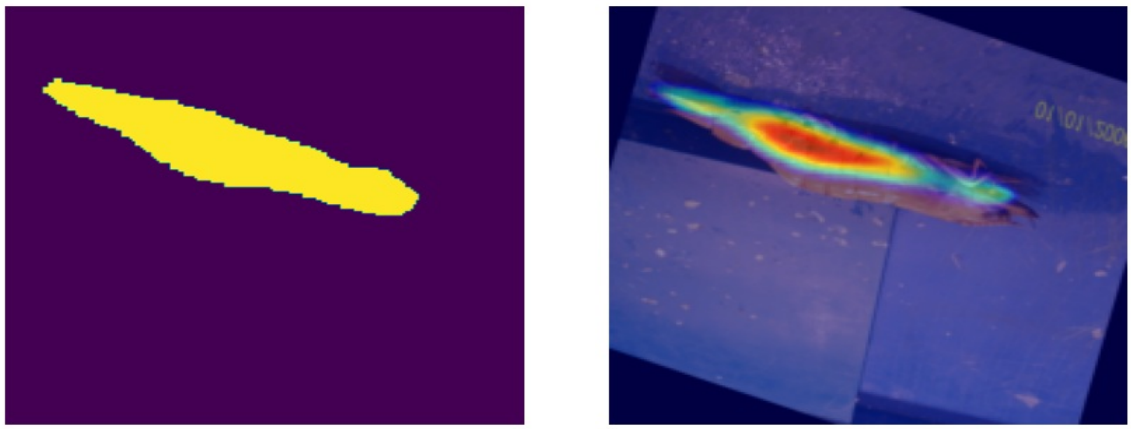
Fonte: O Autor.

Figura 44 – Segmentação e Explicação do exemplo 2 para o modelo Unet.



Fonte: O Autor.

Figura 45 – Segmentação e Explicação do exemplo 3 para o modelo Unet.



Fonte: O Autor.

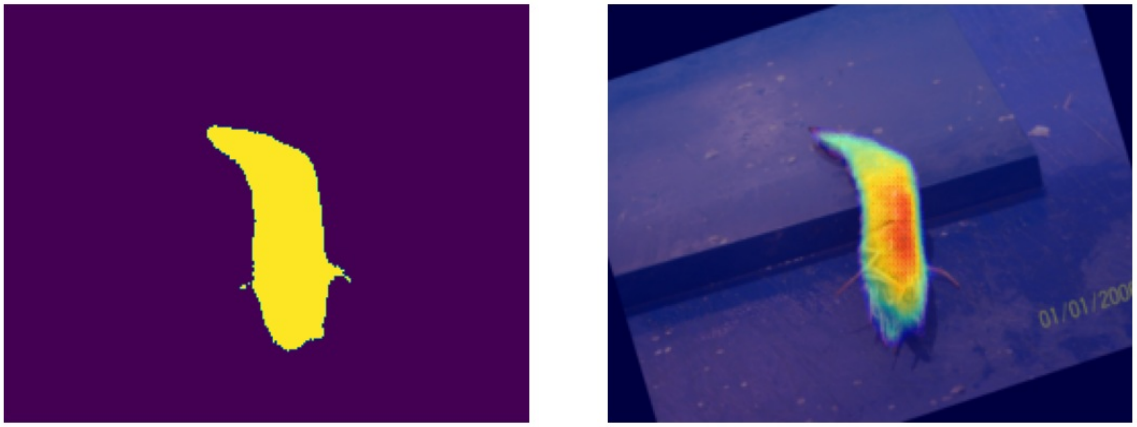
Figura 46 – Segmentação e Explicação do exemplo 4 para o modelo Unet.

Fonte: O Autor.

5.1.2.3 LinkNet

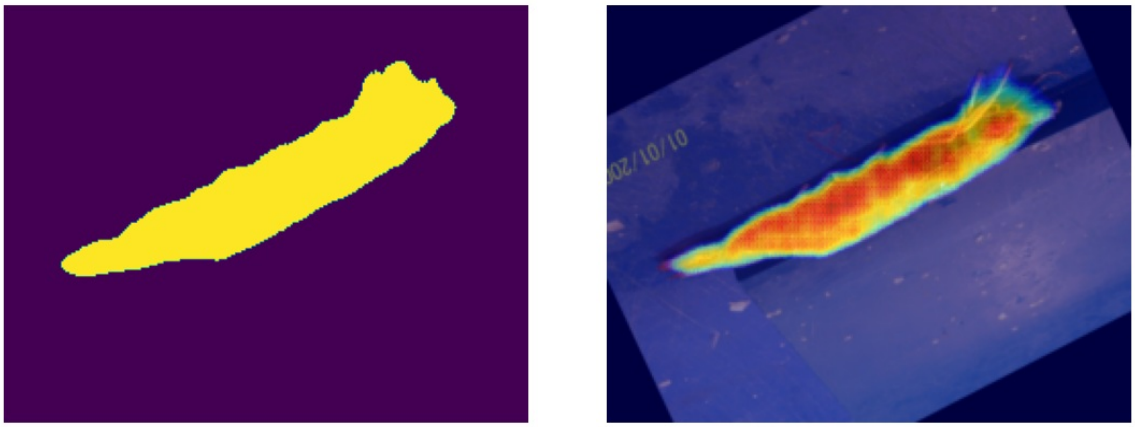
O algorítimo LinkNet por sua vez apresentou uma boa performance em relação aos demais modelos quando analisado pela métrica IoU, sendo estatisticamente semelhante aos dois algoritmos que possuem os maiores valores para esta métrica, e pela observação dos resultados para as imagens selecionadas é possível visualizar sua superioridade em relação aos modelos anteriores. Pela explicação obtida pelo método SegGradCam, nos casos observados, o modelo consegue identificar com valores relevantes grande parte do corpo, sendo encontrada valores menores para as extremidades deste, consequentemente, esta área possui um contorno irregular na segmentação.

Figura 47 – Segmentação e Explicação do exemplo 1 para o modelo Linknet.



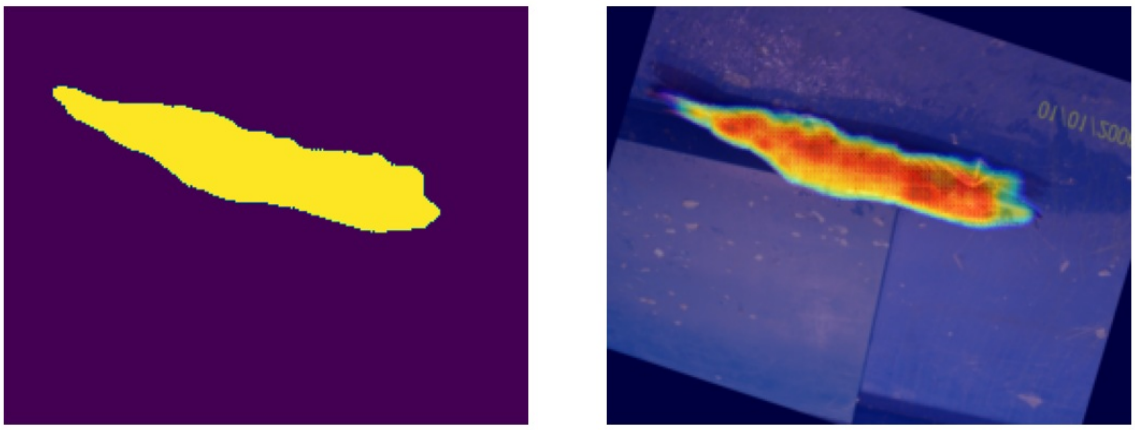
Fonte: O Autor.

Figura 48 – Segmentação e Explicação do exemplo 2 para o modelo Linknet.



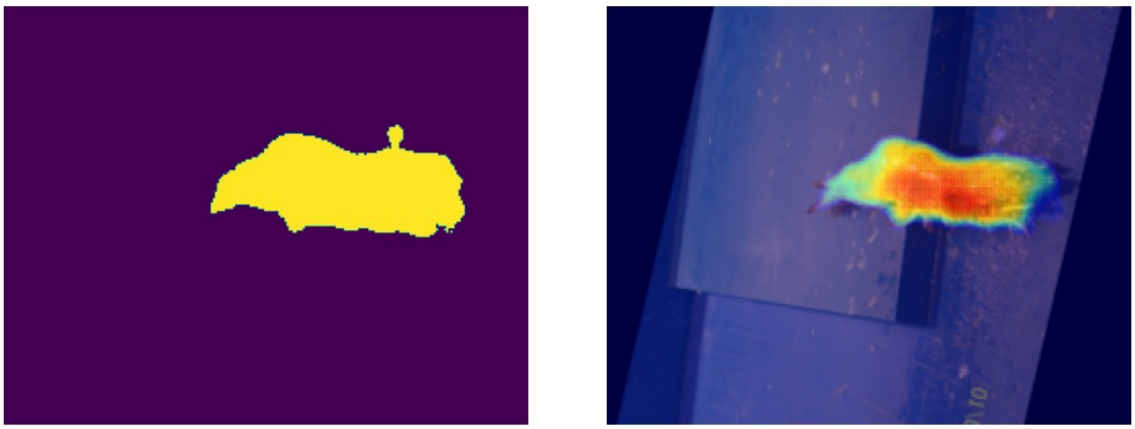
Fonte: O Autor.

Figura 49 – Segmentação e Explicação do exemplo 3 para o modelo Linknet.



Fonte: O Autor.

Figura 50 – Segmentação e Explicação do exemplo 4 para o modelo Linknet.

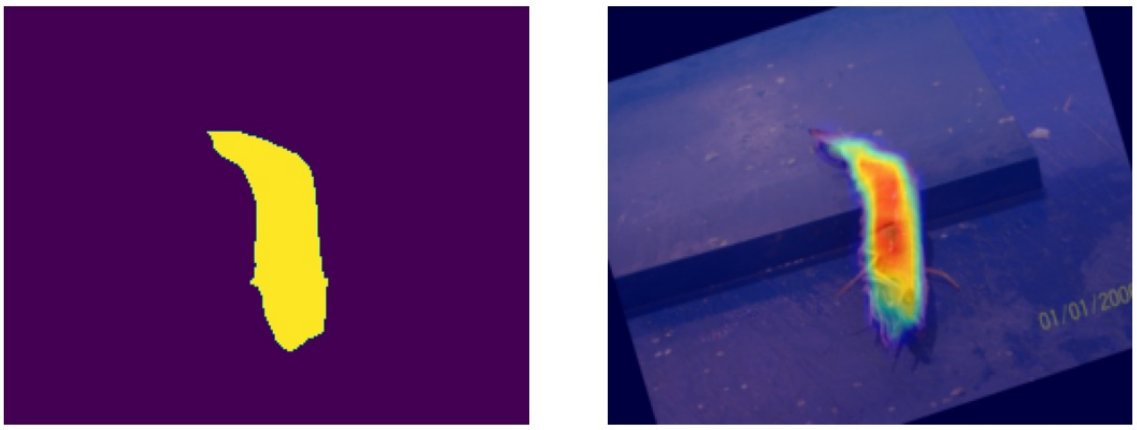


Fonte: O Autor.

5.1.2.4 DeepLab-V3

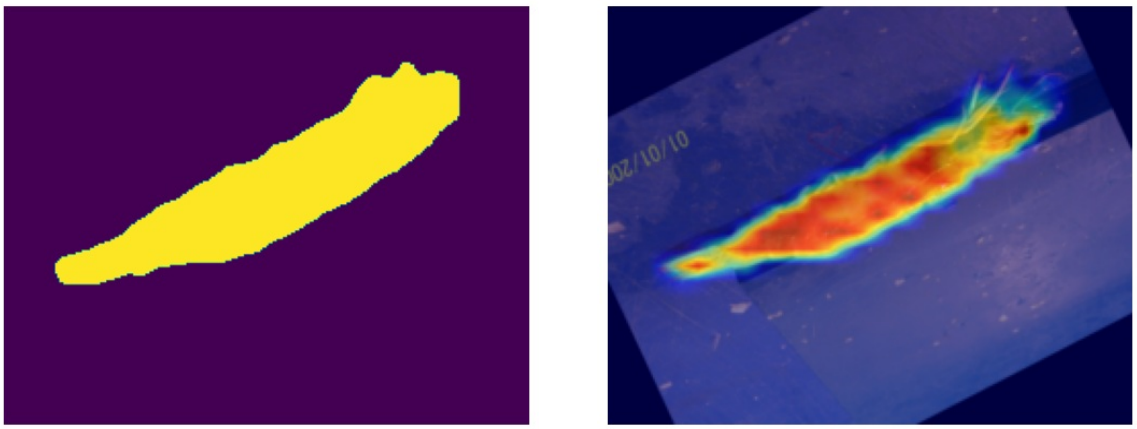
O modelo DeepLab-V3 apresentou comportamento semelhante ao LinkNet para a sua explicação, sendo a principal diferença entre esses o resultado para a extremidades, sendo o DeepLab-V3 em alguns casos mais preciso, porém como observado para a Figura 53, também existem casos em que este algoritmo não detecta uma parte relevante do corpo.

Figura 51 – Segmentação e Explicação do exemplo 1 para o modelo DeepLab-V3.



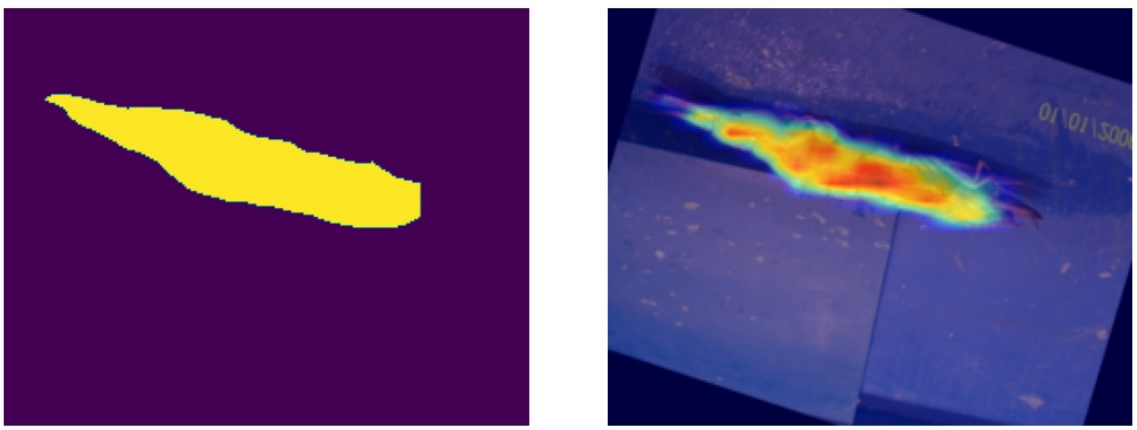
Fonte: O Autor.

Figura 52 – Segmentação e Explicação do exemplo 2 para o modelo DeepLab-V3.



Fonte: O Autor.

Figura 53 – Segmentação e Explicação do exemplo 3 para o modelo DeepLab-V3.



Fonte: O Autor.

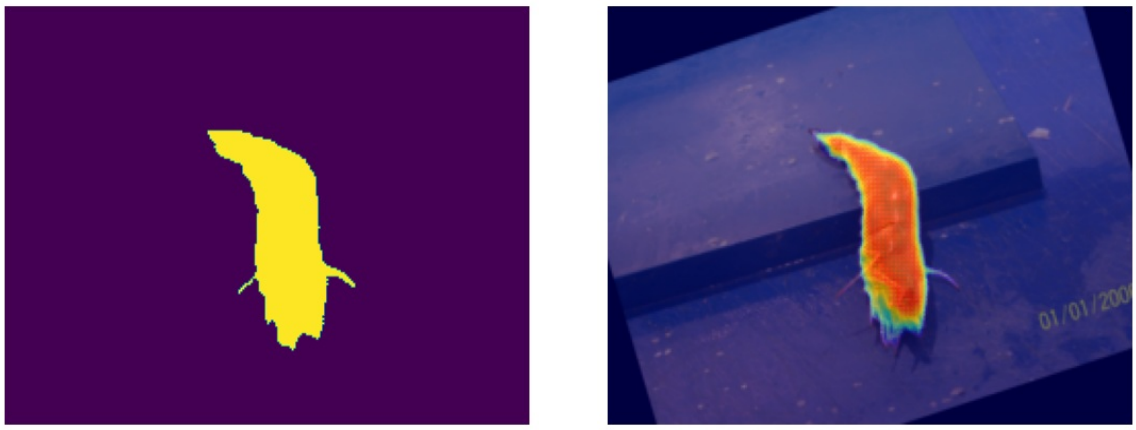
Figura 54 – Segmentação e Explicação do exemplo 4 para o modelo DeepLab-V3.

Fonte: O Autor.

5.1.2.5 Attention UNet

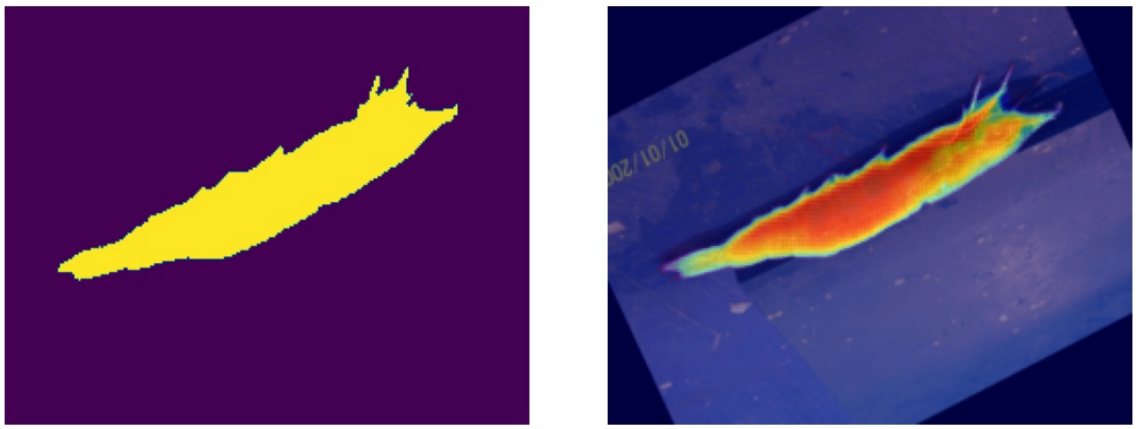
O resultado da segmentação para o algoritmo Attention UNet se mostrou o mais próximo das máscaras para os exemplos selecionados. Observando a explicação obtida é possível identificar uma alta ativação para quase a totalidade do corpo do camarão, sendo capaz de traçar uma delimitação mais precisa para as bordas, quando comparado aos demais modelos. Porém, nas Figuras 57 e 58, foi incluída no resultado da segmentação uma pequena área pertencente ao fundo da imagem, quando analisado em conjunto com os valores da métrica IoU, temos que, em média, o Attention UNet apresentou resultados ligeiramente abaixo do DeepLab-V3, uma possível explicação para esses seria que o Attention UNet, por oferecer uma delimitação mais próxima do real, também está mais suscetível a incluir pixeis adicionais à segmentação, ocasionando em uma redução para métrica de avaliação. Para averiguar essa hipótese, seria necessário uma análise ocular da totalidade de imagens incluídas na base de testes para todas as 30 execuções de cada um dos algoritmos.

Figura 55 – Segmentação e Explicação do exemplo 1 para o modelo Attention Unet.



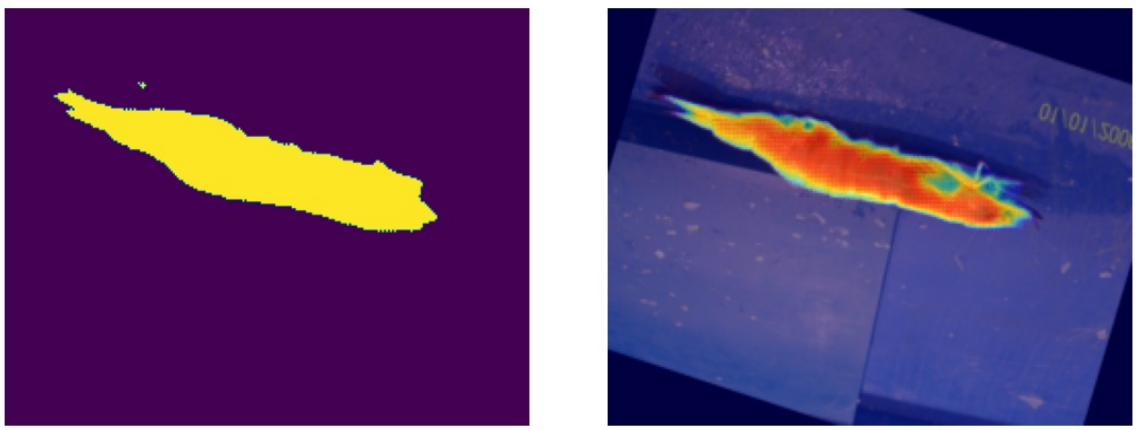
Fonte: O Autor.

Figura 56 – Segmentação e Explicação do exemplo 2 para o modelo Attention Unet.



Fonte: O Autor.

Figura 57 – Segmentação e Explicação do exemplo 3 para o modelo Attention Unet.



Fonte: O Autor.

Figura 58 – Segmentação e Explicação do exemplo 4 para o modelo Attention Unet.

Fonte: O Autor.

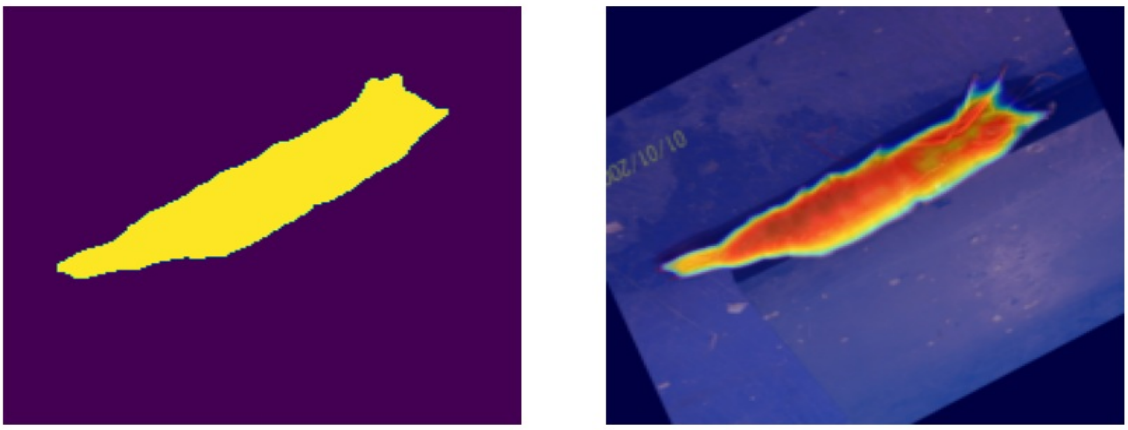
5.1.2.6 Trans Unet

O modelo Trans UNet apresentou um comportamento semelhante ao Attention UNet, porém ainda mais sensível a ruídos contidos no fundo da imagem, como observado na Figura 59.

Figura 59 – Segmentação e Explicação do exemplo 1 para o modelo TransUnet.

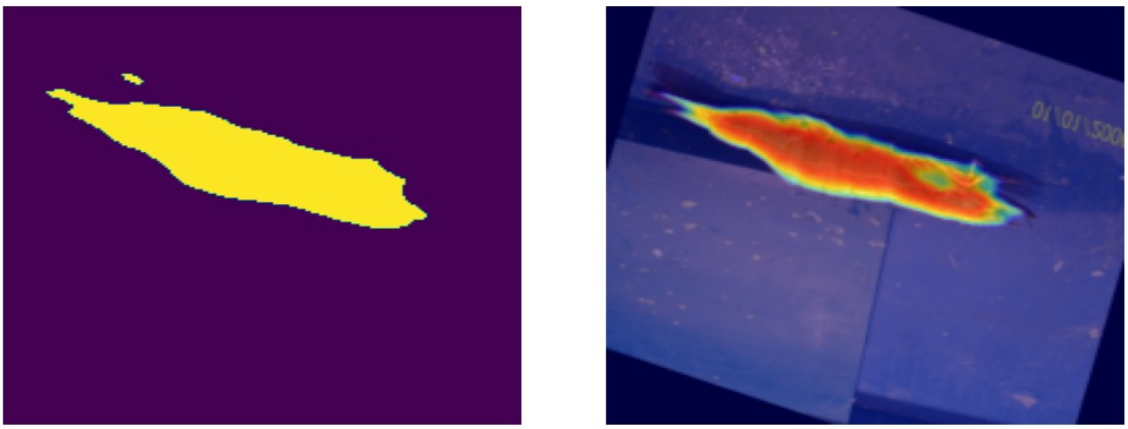
Fonte: O Autor.

Figura 60 – Segmentação e Explicação do exemplo 2 para o modelo TransUnet.



Fonte: O Autor.

Figura 61 – Segmentação e Explicação do exemplo 3 para o modelo TransUnet.



Fonte: O Autor.

Figura 62 – Segmentação e Explicação do exemplo 4 para o modelo TransUnet.

Fonte: O Autor.

Avaliando todos os exemplos gerados para cada um dos algoritmos, os que apresentaram os melhores resultados visualmente coincidem com os indicados pela métrica IoU, sendo esses LinkNet, DeepLab-V3 e Attention UNet. Em relação ao padrão de ativação do gradiente, o comportamento dos dois primeiros modelos citados se assemelham, já o último apresenta uma estrutura diferente, por conta disso esses se mostram mais capacitado de identificar uma maior parte do corpo do camarão, porém, ao mesmo tempo é mais sensível a ruídos no fundo da imagem, incluindo essas áreas na segmentação. Com essa análise é adicionada mais uma diferenciação aos modelos com melhores performances para a análise quantitativa, com o DeepLab-V3 apresentando bordas menos precisas, porém sem extrapolar os limites do corpo (para os exemplos avaliados).

5.2 Avaliação dos Modelos de Classificação

A avaliação dos modelos de classificação de imagem foi realizada utilizando duas métricas para mensurar a performance, sendo elas a acurácia e o F1-Score. Além dessas medidas, assim como feito com os algoritmos de segmentação, também foram comparados o tempo de treinamento e teste, a quantidade de parâmetros treináveis e o tamanho do arquivo. Dessa forma, a Tabela 6 apresenta os dados para análise comparativa entre os modelos utilizados, seguindo a mesma estrutura utilizada na Tabela 4, porém com a substituição da métrica IoU pela acurácia e F1-Score.

Algorithms	Acurácia Média	F1-Score Médio	Número de Parâmetros (M)	Tamanho do Arquivo (MB)	Tempo de Treinamento (s)	Tempo de Teste (s)
VGG	94,53	90,55	134,3	512,3	804	7,63
ResNet	95,63	92,23	11,2	42,7	431	6,20
GoogLeNet	98,76	98,62	5,6	21,5	445	6,21

Tabela 6 – Análise Comparativa de Performance - Modelos de Classificação.

Comparando os resultados obtidos, o algoritmo GoogLeNet obteve o melhor desempenho tanto para a acurácia quanto para o F1-Score, além disso também apresentou o menor número de parâmetros treináveis e, consequentemente, o menor tamanho de arquivo, porém apesar de ser mais leve que o ResNet, este foi ligeiramente mais rápido

tanto no processo treinamento quanto no de teste. Já o modelo VGG apresentou o pior desempenho em relação a todas as métricas analisadas.

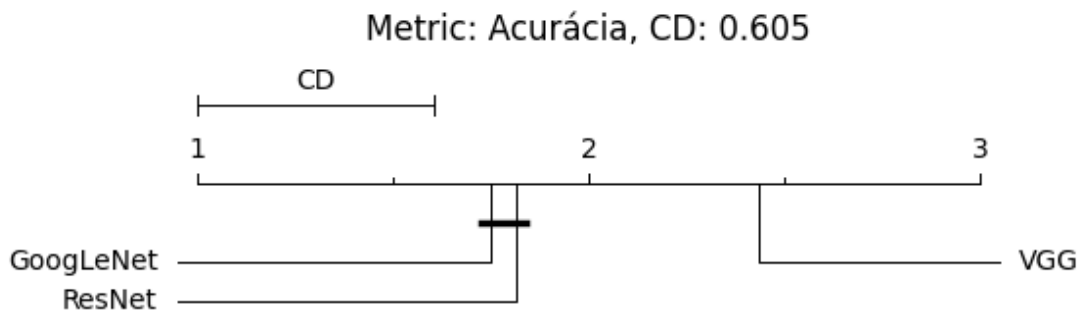
Similar aos modelos de segmentação, foi realizado um teste de significância estatística Friedman-Nemenyi, para determinar se os resultados obtidos para os algoritmos de classificação são estatisticamente diferentes para as métricas acurácia e F1-Score. Assim como no anterior, nesse caso a hipótese nula foi rejeitada com valor-p de $8,3 \times 10^{-4}$. Analisando a distância crítica (CD), foi encontrado o valor de 0.605, determinando como estatisticamente semelhantes os modelos com diferença de rank médio inferior a esse valor. Os ranks médios para cada algoritmo é apresentado na tabela 7, já a comparação gráfica dos resultados estão nas imagens 63 e 64.

Algoritmo	Ranking
GoogLeNet	1,75
ResNet	1,82
VGG	2,43

Tabela 7 – Ranking Médio para as métricas Acurácia e F1-Score.

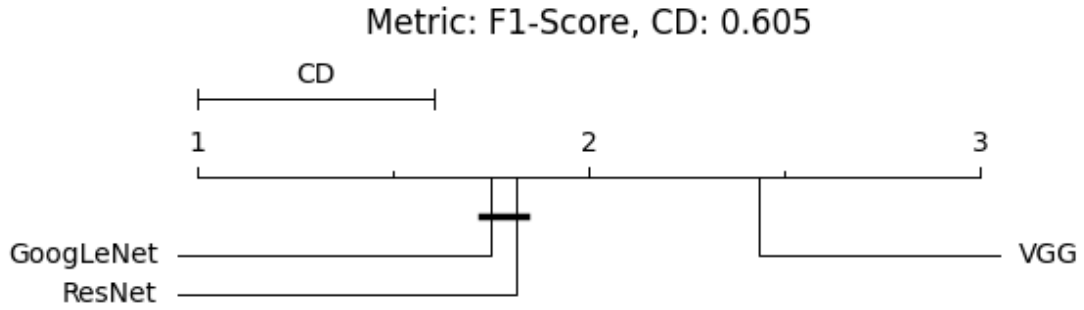
Os resultados obtidos apontam a superioridade do modelo GoogLeNet em relação aos demais, porém essa não é bastante para afirmar que este algoritmo é superior ao ResNet, sendo a diferença entre eles inferior a distância crítica. Por outro lado, o modelo VGG ficou abaixo dos demais, com distância suficiente para estabelecer sua diferença em relação aos outros.

Figura 63 – Diagrama de Diferença Crítica para o teste estatístico Friedman-Nemenyi para métrica Acurácia.



Fonte: O Autor.

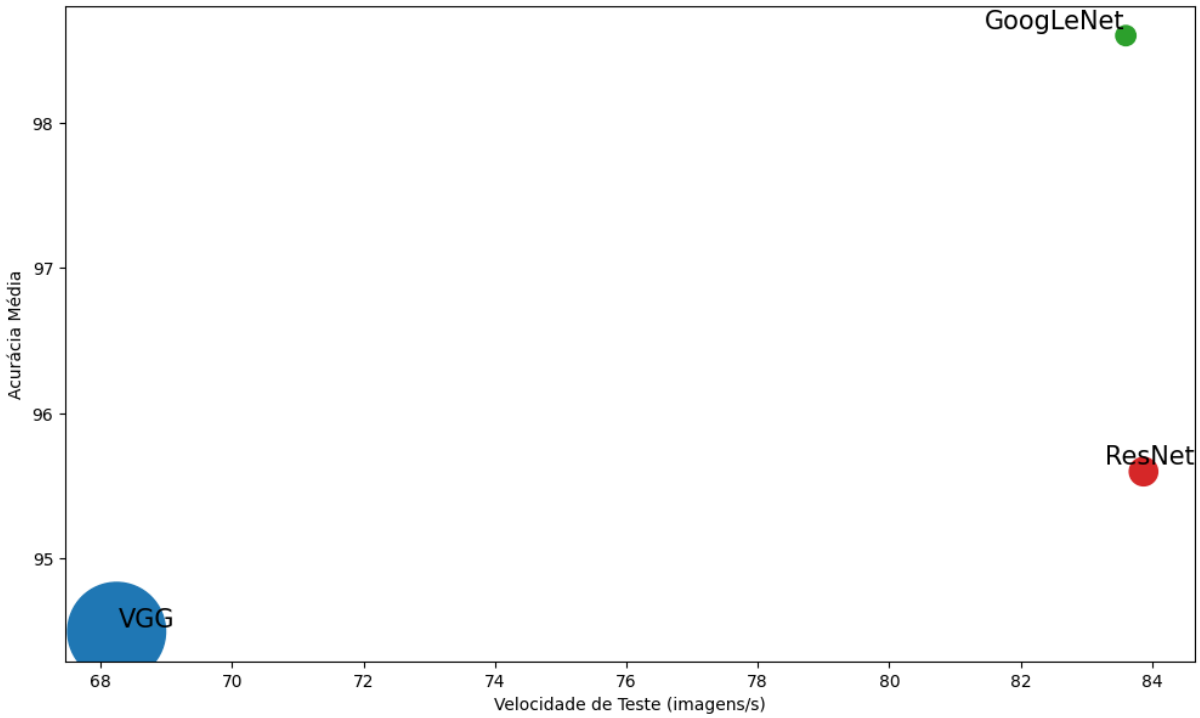
Figura 64 – Diagrama de Diferença Crítica para o teste estatístico Friedman-Nemenyi para métrica F1-Score.



Fonte: O Autor.

Além da análise estatística, também foi construído um gráfico de dispersão, apresentando a relação da acurácia (eixo Y), com a velocidade de teste, em imagens por segundo (eixo X), com a quantidade de parâmetros treináveis (área do círculo). Dada a semelhança da distribuição das métricas acurácia e F1-Score, apenas a primeira foi selecionada para a representação gráfica.

Figura 65 – Comparação de Performance dos Modelos de Classificação Desenvolvidos.



Fonte: O Autor.

O gráfico acima ilustra o que foi encontrado com a análise dos resultados, o modelo

VGG apresenta uma performance abaixo em todos os aspectos avaliados, se localizando no canto inferior esquerdo do diagrama e com maior área. Já em relação aos demais modelos, fica destacado a diferença entre eles, com o algoritmo GoogLeNet acima em relação a acurácia média e o ResNet sendo ligeiramente mais rápido.

5.3 Seleção do Modelo

Dado os resultados apresentados acima, para a tarefa de segmentação, dois algoritmos se mostraram superiores aos demais, são eles o DeepLab-V3 e o Attention UNet. Quando comparados entre si, mesmo o DeepLab-V3 obtendo um valor superior para a métrica IoU, esse não foi considerado um valor estatisticamente relevante pelo teste de significância estatística Friedman-Nemenyi. Adicionalmente, o modelo DeepLab-V3 apresentou um maior custo computacional em relação ao Attetion UNet, possuindo uma quantidade de parâmetros e tamanho do seu arquivo cerca de 15 vezes maior quando comparado. Outro ponto que o coloca abaixo é o tempo necessário para a avaliação das imagens, sendo 25% superior ao do algoritmo Attention UNet. Com esses resultados, caso não exista limitação de recursos computacionais e o processamento das imagens seja realizado *offline*, pode ser escolhido tanto o modelo DeepLab-V3 quanto o Attention UNet, a depender do tipo de erro que seja considerado mais crítico para uma dada situação, sendo o DeepLab-V3 menos sensível a ruídos fora do corpo do animal, porém com bordas menos precisas em relação ao Attention UNet.. Porém, em uma situação em que o uso eficiente de recursos seja necessário, por exemplo o processamento seja realizado *live*, deve ser escolhido o modelo desenvolvido com a arquitetura Attention UNet para a tarefa de segmentação das imagens de camarão.

Já para a classificação das imagens, o algoritmo GoogLeNet foi o escolhido, pois, apesar de não se mostrar estatisticamente diferente ao modelo ResNet, esse se mostrou superior aos demais em outros aspectos como na quantidade de parâmetros treináveis e tamanho do arquivo resultante, necessitando de um menor custo computacional. A única métrica em que o GoogLeNet foi superado é o tempo necessário para avaliação, com o ResNet sendo ligeiramente mais rápido, porém, essa diferença não é suficiente para justificar uma alteração na escolha do modelo.

6 Conclusão

6.1 Considerações Finais

Com o crescimento da produção em aquicultura no Brasil (Nations 2024), oferecendo uma alternativa aos processos de pescas tradicionais, surge, em conjunto, a oportunidade de otimizar alguns dos procedimentos laboriosos e, dessa forma, melhorar todo o processo produtivo. Considerando o cultivo de camarão, um desses procedimentos é a captura dos dados biométricos do animal, que além de custoso em tempo, também contribui para o estresse do camarão, que por sua vez tem efeitos danosos ao metabolismo deste, podendo resultar no desencadeamento de doenças e a mortalidade total da produção.

Este estudo se propõem a desevolver uma metodologia estruturada e replicável para construção de modelos de segmentação de imagens de camarões, de forma que possam ser utilizadas para estimação de dados biométricos como peso e comprimento, sendo uma extensão do artigo intitulado *"A Comparative Study of Deep Learning Approaches for Shrimp Segmentation in Images"* submetido para publicação no *Engineering Applications of Artificial Intelligence*. Para tanto foram treinados modelos de segmentação e classificação de imagens, utilizando diversas arquiteturas estabelecidas, com objetivo de avaliar a compatibilidade de cada uma com a tarefa em questão.

A avaliação dos modelos é um processo tão importante quanto o treinamento, para essa foi feita uma análise tanto quantitativa, para segmentação e classificação, quanto qualitativa, apenas para segmentação. Para a análise quantitativa foram observadas métricas de performance em relação à tarefa, IoU para segmentação, e acurácia e F1-Score para classificação, como também medidas de performance computacional, como tempo de teste e quantidade de parâmetros treináveis. Já a análise qualitativa foi realizada pela avaliação das áreas relevantes para a segmentação, obtidas pela aplicação do método SegGradCam.

Com a análise de resultados foi encontrado que os algoritmos que apresentaram a melhor performance foram DeepLab-V3 e Attention UNet, para segmentação, e GoogLeNet, para classificação. Para os modelos desenvolvidos para segmentação, a superioridades dos citados se deu pela métrica IoU com o DeepLab-V3 apresentando os melhores valores, com Attention UNet em seguida, apesar dessa superioriade, ao aplicar o

teste de significância estatística Friedman-Nemenyi, foi encontrado que os modelos são estatísticamente semelhantes. Por outro lado, ao avaliar pela perspectiva da performance computacional, foi encontrado que o modelo DeepLab-V3 não obteve um bom resultado relativo aos demais, já o Attention UNet se mostrou o mais eficiente entre os algoritmos competitivos. Dessa forma, o Attention UNet se mostrou mais equilibrado combinando um valor médio para o IoU ligeiramente abaixo com uma redução no tempo de processamento e quantidade de parâmetros treináveis.

Em relação ao modelos de classificação, o algoritmo GoogLeNet apresentou o melhor resultado tanto para a acurácia quanto para o F1-Score, junto com isso, também foi o modelo com a menor quantidade de parâmetros treináveis entre os analisados, sendo superado apenas no quesito de tempo de processamento, onde o ResNet se mostrou um pouco mais rápido.

Com esses resultados, pode-se concluir que algoritmos baseados em redes neurais convolucionais se mostraram aptas para as tarefas de classificação e segmentação para o conjunto de imagens de camarão utilizadas para o seu desenvolvimento.

6.2 Trabalhos Futuros

No planejamento inicial desta pesquisa foi definido o uso de uma base de dados original, com imagens e dados biométricos capturados durante o processo realizado. A construção dessa base não só possibilitaria o desenvolvimento de modelos mais adequados aos problemas específicos do cultivo de camarão em Pernambuco, como também, por conta dos dados biométricos associados aos animais, a estimativa de peso e tamanho em conjunto com o processo de classificação e segmentação. A construção dessa base seria uma das etapas do projeto maior da colaboração do DC com o DEPAQ, porém, devido a limitações tanto na captura das imagens quanto para a criação das máscaras necessárias para o treinamento dos modelos, esse planejamento foi alterado para utilizar uma base de dados de terceiros.

Para o desenvolvimento de trabalhos futuros o principal aspecto necessário para completar essa pesquisa seria o desenvolvimento de modelos capazes de, a partir da imagem segmentada do camarão, estimar as medidas biométricas. Para tanto, será necessário uma base de dados que possua imagens do animal em conjunto com esses dados. A conclusão

do desenvolvimento da base forneceria o conjunto adequado para essa tarefa.

Referências

AGHDAM, H. H.; HERAVI, E. J. et al. Guide to convolutional neural networks. **New York, NY: Springer**, Springer, v. 10, n. 978-973, p. 51, 2017.

AZIZ, R. M.; MAHTO, R.; DAS, A.; AHMED, S. U.; ROY, P.; MALLIK, S.; LI, A. Co-woa: novel optimization approach for deep learning classification of fish image. **Chemistry & Biodiversity**, Wiley Online Library, v. 20, n. 8, p. e202201123, 2023.

BONDAD-REANTASO, M. G.; SUBASINGHE, R. P.; JOSUPEIT, H.; CAI, J.; ZHOU, X. The role of crustacean fisheries and aquaculture in global food security: Past, present and future. **Journal of Invertebrate Pathology**, v. 110, n. 2, p. 158–165, 2012. ISSN 0022-2011. Diseases in Aquatic Crustaceans: Problems and Solutions for Global Food Security. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0022201112000699>.

CHAURASIA, A.; CULURCIELLO, E. Linknet: Exploiting encoder representations for efficient semantic segmentation. In: IEEE. **2017 IEEE visual communications and image processing (VCIP)**. [S.l.], 2017. p. 1–4.

CHEN, J.; LU, Y.; YU, Q.; LUO, X.; ADELI, E.; WANG, Y.; LU, L.; YUILLE, A. L.; ZHOU, Y. Transunet: Transformers make strong encoders for medical image segmentation. **arXiv preprint arXiv:2102.04306**, 2021.

CHEN, L.-C.; PAPANDREOU, G.; SCHROFF, F.; ADAM, H. Rethinking atrous convolution for semantic image segmentation. **arXiv preprint arXiv:1706.05587**, 2017.

CHEN, W.; GONG, X.; LIU, X.; ZHANG, Q.; LI, Y.; WANG, Z. Fasterseg: Searching for faster real-time semantic segmentation. **arXiv preprint arXiv:1912.10917**, 2019.

CHROUSOS, G. P.; GOLD, P. W. The concepts of stress and stress system disorders: overview of physical and behavioral homeostasis. **Jama**, American Medical Association, v. 267, n. 9, p. 1244–1252, 1992.

ÇIÇEK, Ö.; ABDULKADIR, A.; LIENKAMP, S. S.; BROX, T.; RONNEBERGER, O. 3d u-net: learning dense volumetric segmentation from sparse annotation. In: SPRINGER. **Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19**. [S.l.], 2016. p. 424–432.

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S. et al. An image is worth 16x16 words: Transformers for image recognition at scale. **arXiv preprint arXiv:2010.11929**, 2020.

EDWARDS, P.; DEMAINE, H. Rural aquaculture: overview and framework for country reviews. 1997.

FERNANDES, A. F.; TURRA, E. M.; ALVARENGA, É. R. de; PASSAFARO, T. L.; LOPES, F. B.; ALVES, G. F.; SINGH, V.; ROSA, G. J. Deep learning image segmentation for extraction of fish body measurements and prediction of body weight and carcass traits

in nile tilapia. **Computers and electronics in agriculture**, Elsevier, v. 170, p. 105274, 2020.

GHOSH, S.; DAS, N.; DAS, I.; MAULIK, U. Understanding deep learning techniques for image segmentation. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 52, n. 4, p. 1–35, 2019.

GU, J.; WANG, Z.; KUEN, J.; MA, L.; SHAHROUDY, A.; SHUAI, B.; LIU, T.; WANG, X.; WANG, G.; CAI, J. et al. Recent advances in convolutional neural networks. **Pattern recognition**, Elsevier, v. 77, p. 354–377, 2018.

HAO, S.; ZHOU, Y.; GUO, Y. A brief survey on semantic segmentation with deep learning. **Neurocomputing**, Elsevier, v. 406, p. 302–321, 2020.

HAYKIN, S. **Neural networks and learning machines, 3/E**. [S.l.]: Pearson Education India, 2009.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HOLZINGER, A.; GOEBEL, R.; FONG, R.; MOON, T.; MÜLLER, K.-R.; SAMEK, W. xxai-beyond explainable artificial intelligence. In: SPRINGER. **International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers**. [S.l.], 2020. p. 3–10.

IANDOLA, F. N.; HAN, S.; MOSKEWICZ, M. W.; ASHRAF, K.; DALLY, W. J.; KEUTZER, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. **arXiv preprint arXiv:1602.07360**, 2016.

JETLEY, S.; LORD, N. A.; LEE, N.; TORR, P. H. Learn to pay attention. **arXiv preprint arXiv:1804.02391**, 2018.

KIRILLOV, A.; HE, K.; GIRSHICK, R.; ROTHER, C.; DOLLÁR, P. Panoptic segmentation. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. [S.l.: s.n.], 2019. p. 9404–9413.

KOUSHIK, C. V.; KAMAL, R. V.; TARUN, C.; TEJA, K. D.; MANNE, S. An efficient algorithm for prawn detection and length identification. In: SPRINGER. **Proceedings of International Conference on Computational Intelligence and Data Engineering: ICCIDE 2020**. [S.l.], 2021. p. 457–467.

LAI, P.-C.; LIN, H.-Y.; LIN, J.-Y.; HSU, H.-C.; CHU, Y.-N.; LIOU, C.-H.; KUO, Y.-F. Automatic measuring shrimp body length using cnn and an underwater imaging system. **Biosystems Engineering**, Elsevier, v. 221, p. 224–235, 2022.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.

LEI, T.; NANDI, A. K. **Image segmentation: principles, techniques, and applications**. [S.l.]: John Wiley & Sons, 2022.

LI, D.; YANG, Y.; ZHAO, S.; YANG, H. A fish image segmentation methodology in aquaculture environment based on multi-feature fusion model. **Marine environmental research**, Elsevier, v. 190, p. 106085, 2023.

LIU, Z.; JIA, X.; XU, X. Study of shrimp recognition methods using smart networks. **Computers and electronics in agriculture**, Elsevier, v. 165, p. 104926, 2019.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 3431–3440.

MALIK, H.; NAEEM, A.; HASSAN, S.; ALI, F.; NAQVI, R. A.; YON, D. K. Multi-classification deep neural networks for identification of fish species using camera captured images. **Plos one**, Public Library of Science San Francisco, CA USA, v. 18, n. 4, p. e0284992, 2023.

MERCIER, L.; PALACIOS, E.; CAMPA-CÓRDOVA, Á. I.; TOVAR-RAMÍREZ, D.; HERNÁNDEZ-HERRERA, R.; RACOTTA, I. S. Metabolic and immune responses in pacific whiteleg shrimp litopenaeus vannamei exposed to a repeated handling stress. **Aquaculture**, Elsevier, v. 258, n. 1-4, p. 633–640, 2006.

MICHELUCCI, U. **Advanced applied deep learning: convolutional neural networks and object detection**. [S.l.]: Apress, 2019.

MILLETARI, F.; NAVAB, N.; AHMADI, S.-A. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: IEEE. **2016 fourth international conference on 3D vision (3DV)**. [S.l.], 2016. p. 565–571.

MINAEE, S.; BOYKOV, Y.; PORIKLI, F.; PLAZA, A.; KEHTARNAVAZ, N.; TERZOPOULOS, D. Image segmentation using deep learning: A survey. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 44, n. 7, p. 3523–3542, 2021.

MUNOZ-BENAVENT, P.; ANDREU-GARCÍA, G.; VALIENTE-GONZÁLEZ, J. M.; ATIENZA-VANACLOIG, V.; PUIG-PONS, V.; ESPINOSA, V. Automatic bluefin tuna sizing using a stereoscopic vision system. **ICES Journal of Marine Science**, Oxford University Press, v. 75, n. 1, p. 390–401, 2018.

NATIONS, F. **The State of World Fisheries and Aquaculture 2024: Blue Transformation in action**. FAO, 2024. (The State of World Fisheries and Aquaculture (SOFIA) ;). ISBN 9789251387634. Disponível em: <https://books.google.it/books?id=ytAREQAAQBAJ>.

OKTAY, O.; SCHLEMPER, J.; FOLGOC, L. L.; LEE, M.; HEINRICH, M.; MISAWA, K.; MORI, K.; MCDONAGH, S.; HAMMERLA, N. Y.; KAINZ, B. et al. Attention u-net: Learning where to look for the pancreas. **arXiv preprint arXiv:1804.03999**, 2018.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. **arXiv preprint arXiv:1511.08458**, 2015.

PANICHELLA, A. A systematic comparison of search-based approaches for lda hyperparameter tuning. **Information and Software Technology**, Elsevier, v. 130, p. 106411, 2021.

POONNOY, P.; ASAVASANTI, S. Implementation of coupled pattern recognition and regression artificial neural networks for mass estimation of headless-shell-on shrimp with random postures. **Journal of Food Process Engineering**, Wiley Online Library, v. 44, n. 8, p. e13747, 2021.

POUDEL, R. P.; BONDE, U.; LIWICKI, S.; ZACH, C. Contextnet: Exploring context and detail for semantic segmentation in real-time. **arXiv preprint arXiv:1805.04554**, 2018.

RASAMOELINA, A. D.; ADJAILIA, F.; SINČÁK, P. A review of activation function for artificial neural network. In: IEEE. **2020 IEEE 18th world symposium on applied machine intelligence and informatics (SAMI)**. [S.l.], 2020. p. 281–286.

REBALA, G.; RAVI, A.; CHURIWALA, S.; REBALA, G.; RAVI, A.; CHURIWALA, S. Machine learning definition and basics. **An introduction to machine learning**, Springer, p. 1–17, 2019.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. **Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18**. [S.l.], 2015. p. 234–241.

SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 4510–4520.

SELVARAJU, R. R.; COGSWELL, M.; DAS, A.; VEDANTAM, R.; PARIKH, D.; BATRA, D. Grad-cam: visual explanations from deep networks via gradient-based localization. **International journal of computer vision**, Springer, v. 128, p. 336–359, 2020.

SETIAWAN, A.; HADIYANTO, H.; WIDODO, C. E. Shrimp body weight estimation in aquaculture ponds using morphometric features based on underwater image analysis and machine learning approach. **Revue d'Intelligence Artificielle**, International Information and Engineering Technology Association (IIETA), v. 36, n. 6, p. 905, 2022.

SHELDON, M. R.; FILLYAW, M. J.; THOMPSON, W. D. The use and interpretation of the friedman test in the analysis of ordinal-scale data in repeated measures designs. **Physiotherapy Research International**, Wiley Online Library, v. 1, n. 4, p. 221–228, 1996.

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

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 1–9.

TORREY, L.; SHAVLIK, J. Transfer learning. In: **Handbook of research on machine learning applications and trends: algorithms, methods, and techniques**. [S.l.]: IGI global, 2010. p. 242–264.

ULUCAN, O.; KARAKAYA, D.; TURKAN, M. A large-scale dataset for fish segmentation and classification. In: IEEE. **2020 Innovations in Intelligent Systems and Applications Conference (ASYU)**. [S.l.], 2020. p. 1–5.

VALENTI, W. C.; BARROS, H. P.; MORAES-VALENTI, P.; BUENO, G. W.; CAVALLI, R. O. Aquaculture in brazil: past, present and future. **Aquaculture Reports**, Elsevier, v. 19, p. 100611, 2021.

VINOGRADOVA, K.; DIBROV, A.; MYERS, G. Towards interpretable semantic segmentation via gradient-weighted class activation mapping (student abstract). In: **Proceedings of the AAAI conference on artificial intelligence**. [S.l.: s.n.], 2020. v. 34, n. 10, p. 13943–13944.

WANG, S.-C. Artificial neural network. In: _____. **Interdisciplinary Computing in Java Programming**. Boston, MA: Springer US, 2003. p. 81–100. ISBN 978-1-4615-0377-4. Disponível em: <https://doi.org/10.1007/978-1-4615-0377-4_5>.

WANG, Y.; LI, Y.; SONG, Y.; RONG, X. The influence of the activation function in a convolution neural network model of facial expression recognition. **Applied Sciences**, MDPI, v. 10, n. 5, p. 1897, 2020.

WANG, Y.; ZHAO, Y.; PETZOLD, L. An empirical study on the robustness of the segment anything model (sam). **Pattern Recognition**, Elsevier, v. 155, p. 110685, 2024.

WEISS, K.; KHOSHGOFTAAR, T. M.; WANG, D. A survey of transfer learning. **Journal of Big data**, Springer, v. 3, p. 1–40, 2016.

WU, Y.-c.; FENG, J.-w. Development and application of artificial neural network. **Wireless Personal Communications**, Springer, v. 102, p. 1645–1656, 2018.

XU, K.; BA, J.; KIROS, R.; CHO, K.; COURVILLE, A.; SALAKHUDINOV, R.; ZEMEL, R.; BENGIO, Y. Show, attend and tell: Neural image caption generation with visual attention. In: PMLR. **International conference on machine learning**. [S.l.], 2015. p. 2048–2057.

YAMASHITA, R.; NISHIO, M.; DO, R. K. G.; TOGASHI, K. Convolutional neural networks: an overview and application in radiology. **Insights into imaging**, Springer, v. 9, p. 611–629, 2018.

ZHOU, H.; KIM, S. H.; KIM, S. C.; KIM, C. W.; KANG, S. W.; KIM, H. Instance segmentation of shrimp based on contrastive learning. **Applied Sciences**, MDPI, v. 13, n. 12, p. 6979, 2023.

ZHU, Y. On the statistical significance. **arXiv preprint physics/0507145**, 2005.

ZOU, Z.; CHEN, K.; SHI, Z.; GUO, Y.; YE, J. Object detection in 20 years: A survey. **Proceedings of the IEEE**, IEEE, v. 111, n. 3, p. 257–276, 2023.