



**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA - CEFET/RJ CAMPUS PETRÓPOLIS
CURSO: BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO**

Segmentação e clusterização de anomalias em oleodutos subaquáticos

Caio Emiliano Rodrigues
Marcus Vinicius Rosa de Oliveira

**PETRÓPOLIS
2023**

**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA - CEFET/RJ CAMPUS PETRÓPOLIS
CURSO: BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO**

Segmentação e clusterização de anomalias em oleodutos subaquáticos

Caio Emiliano Rodrigues
Marcus Vinicius Rosa de Oliveira

Trabalho de Conclusão de Curso apresentado ao
CEFET/RJ - *campus* Petrópolis, como parte dos
requisitos para obtenção do título de Bacharel em
Engenharia de Computação.

Orientador: Prof. Diego Barreto Haddad
Co-orientadora: Profa. Fernanda Duarte Vilela Reis de Oliveira

**PETRÓPOLIS
2023**

**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA - CEFET/RJ CAMPUS PETRÓPOLIS
CURSO: BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO**

FOLHA DE APROVAÇÃO

Segmentação e clusterização de anomalias em oleodutos subaquáticos

Caio Emiliano Rodrigues
Marcus Vinicius Rosa de Oliveira

Trabalho de Conclusão de Curso apresentado ao
CEFET/RJ - *campus* Petrópolis, como parte dos
requisitos para obtenção do título de Bacharel em
Engenharia de Computação.

Orientador: Prof. Diego Barreto Haddad
Co-orientadora: Profa. Fernanda Duarte Vilela Reis de Oliveira

Aprovado por:

Prof. Diego Barreto Haddad, D.Sc. (Orientador)

Profa. Fernanda Duarte Vilela Reis de Oliveira (co-orientadora), D.Sc.

Prof. Luís Domingues Tomé Jardim Tarrataca, D.Sc.

Profa. Mariane Rembold Petraglia , D.Sc.

Junho de 2023

DEDICATÓRIA

Aos nossos familiares Paulo de Souza Emiliano e Daniela Emiliano e José de Oliveira Neto e Ildete Ferreira Neto, nossos maiores incentivadores nos estudos.

Aos nossos professores e orientadores.

E aos amigos e familiares que nos acompanharam nesta trajetória

AGRADECIMENTO

Eu, Caio, agradeço primeiramente a Deus, por me capacitar e me guiar durante toda minha jornada acadêmica. Agradeço aos meus familiares por todo apoio e incentivo fornecidos. Agradeço a todos os professores do CEFET-RJ que contribuíram para meu aprendizado e formação, em especial o professor Luís Tarrataca, por ter me auxiliado e incentivado também em um dos momentos mais difíceis que enfrentei. Por fim, agradeço aos meus amigos e colegas de curso, que compartilharam comigo a experiência acadêmica.

Eu, Marcus, agradeço aos meus pais, por terem me dado suporte durante toda a minha trajetória acadêmica. Aos meus familiares, que me possibilitaram mudar de cidade para que eu conseguisse me dedicar integralmente aos estudos. E aos meus colegas de turma, que compartilharam experiências comigo durante toda a faculdade.

Agradecemos em conjunto aos professores Diego Haddad e Fernanda Oliveira por terem nos orientado, incentivado e auxiliado por anos nos projetos de PIBIC e neste trabalho.

RESUMO

A segmentação e clusterização de anomalias em oleodutos subaquáticos contribui para a detecção precisa e eficiente de áreas problemáticas, permitindo a tomada de medidas preventivas e a manutenção adequada dos oleodutos, evitando danos ambientais e prejuízos econômicos. O estudo demonstra a aplicação bem-sucedida de técnicas de segmentação e clusterização de oleodutos, evidenciando seu potencial para outras áreas de análise de imagens subaquáticas. Uma base de dados composta por 34.323 imagens, obtidas por meio de inspeção ROV e anotadas com a posição do oleoduto, foi construída. A segmentação das imagens foi realizada utilizando a ferramenta *Detectron 2*, enquanto a clusterização das anomalias foi conduzida por meio dos algoritmos *k-means* e *fuzzy k-means*. Foram obtidos resultados satisfatórios, com uma precisão média de aproximadamente 93.8% para o conjunto de teste na etapa de segmentação e uma avaliação quantitativa do agrupamento de anomalias foi realizada na clusterização. Esse resultado destaca a eficácia da abordagem proposta no processo de identificação e clusterização das anomalias nos oleodutos subaquáticos.

Palavras-chaves: Oleodutos subaquáticos, Visão computacional, *Detectron 2*, Extração de características, Clusterização.

ABSTRACT

The segmentation and clustering of anomalies in underwater pipelines contribute to the accurate and efficient detection of problematic areas, enabling preventive measures and proper maintenance of the pipelines, avoiding environmental damage and economic losses. The study demonstrates the successful application of segmentation and clustering techniques for pipelines, highlighting their potential for other areas of underwater image analysis. A database composed of 34,323 images, obtained through ROV inspection and annotated with the pipeline's position, was constructed. The image segmentation was performed using the *Detectron 2* tool, while the clustering of anomalies was conducted using the *k-means* and *fuzzy k-means* algorithms. Satisfactory results were achieved, with an average precision of approximately 93.8% for the test set in the segmentation stage, and a quantitative evaluation of anomaly clustering was performed. This result highlights the effectiveness of the proposed approach in the identification and clustering of anomalies in underwater pipelines.

Key-words: Underwater pipelines, Computer vision, Detectron 2, Feature extraction, Clustering.

LISTA DE FIGURAS

1	ROV - Veículo Operado Remotamente.	1
2	Imagens de Oleodutos	2
3	Exemplo da Interface	5
4	Em preto, exemplo de anotação da base de dados	6
5	Modelo neuronal proposto por McCulloch e Pitts.	8
6	Exemplo de linearmente separável e não separável em duas classes de dados.	9
7	Representação gráfica do algoritmo de <u>backpropagation</u>	12
8	Arquitetura CNN simplificada.	13
9	Ativações obtidas da primeira camada convolucional de uma rede neural convolucional simplificada, após o treinamento no banco de dados MNIST de dígitos escritos à mão.	14
10	Exemplo da invariância à translação em uma rede neural convolucional. A entrada no canto inferior esquerdo é uma versão transladada da imagem de entrada no canto superior esquerdo em um pixel para a direita e um pixel para baixo.	15
11	Arquitetura da rede R-CNN.	17
12	Performance de detecção dos modelos YOLO	21
13	Arquitetura da solução proposta.	22
14	Arquitetura da rede VGG16.	27
15	Variância acumulada x Número de componentes	28
16	Precisão média - COCO x Épocas	32
17	Precisão média - COCO (IoU=0.50) x Épocas	33
18	Precisão média - COCO (IoU=0.75) x Épocas	33
19	<i>Total loss</i> da rede (em escala logarítmica) x Épocas	34
20	Precisão média - COCO x Épocas	35
21	Precisão média - COCO (IoU=0.50) x Épocas	35
22	Precisão média - COCO (IoU=0.75) x Épocas	36
23	<i>Total loss</i> da rede (em escala logarítmica) x Épocas	36
24	Precisão média - COCO x Épocas	37
25	Precisão média - COCO (IoU=0.50) x Épocas	38
26	Precisão média - COCO (IoU=0.75) x Épocas	38
27	<i>Total loss</i> da rede (em escala logarítmica) x Épocas	39
28	<i>k-means</i> - Distribuição de amostras por <i>clusters</i>	40
29	<i>fuzzy k-means</i> - Distribuição de amostras por <i>clusters</i>	41

LISTA DE TABELAS

1	Configuração 1 dos hiperparâmetros	31
2	Configuração 2 dos hiperparâmetros	31
3	Configuração 3 dos hiperparâmetros	32
4	Parâmetros usados no algoritmo <i>k-means</i>	39
5	Parâmetros usados no algoritmo <i>fuzzy k-means</i>	40
6	Comparação entre as diferentes configurações apresentadas.	42
7	Análise quantitativa do algoritmo <i>k-means</i>	42
8	Análise quantitativa do algoritmo <i>fuzzy k-means</i>	42

Lista de Siglas

ROV	- <i>Remotely Operated Vehicle</i>
RCNN	- <i>Region-Based Convolutional Neural Network</i>
CNN	- <i>Convolutional Neural Network</i>
SVM	- <i>Support Vector Machines</i>
FCM	- <i>Fuzzy c-means</i>
RPN	- <i>Region Proposal Network</i>
PCA	- <i>Principal Component Analysis</i>
AP	- <i>Average Precision</i>

SUMÁRIO

1	Introdução	1
1.1	Tema	2
1.2	Delimitação	3
1.3	Justificativa	4
1.4	Objetivos	4
1.5	Metodologia	5
1.6	Descrição	6
2	Fundamentação Teórica	7
2.1	Visão computacional	7
2.2	Redes neurais	8
2.2.1	Single-layer neural network	8
2.2.2	Multi-layer neural network	10
2.2.3	Aprendizado em redes feed-forward	10
2.3	Convolutional Neural Network	13
2.3.1	Camadas de convolução e pooling	14
2.3.2	Otimizador	15
2.4	R-CNN	16
2.5	Feature Extraction	17
2.6	Aprendizado não supervisionado e clusterização	18
3	Trabalhos Relacionados	20
4	Solução Proposta	22
4.1	Segmentação do oleoduto	22
4.1.1	Construção da base de dados	23
4.1.2	Pré-processamento das imagens	23
4.1.3	Criação do catálogo dos dados no formato do <i>Detectron 2</i>	24
4.1.4	Treinamento do modelo <i>Detectron 2</i>	25
4.2	Clusterização de Anomalias	26
4.2.1	Extração de Características	27
4.2.2	Principal Component Analysis (PCA)	28
4.2.3	Clusterização de anomalias: <i>k-means</i> e <i>fuzzy k-means</i>	29
5	Resultados	30
5.1	Experimento 1	31

5.2	Experimento 2	34
5.3	Experimento 3	37
5.4	Experimento 4	39
5.5	Considerações finais	41
6	Conclusão	43

1 Introdução

Os combustíveis fósseis, dentre os quais o petróleo e o gás natural são os exemplos mais prominentes, consistem na nossa principal fonte de energia, permitindo o funcionamento dos diferentes meios de transporte e a manutenção do nosso estilo de vida moderno (1). Constituído por uma mescla de compostos orgânicos, especialmente hidrocarbonetos, o petróleo fica armazenado no interior de poros ou espaços vazios de rochas impermeáveis, que na sua grande maioria encontram-se no fundo do mar.

Durante o procedimento de extração do petróleo, o escoamento da produção ocorre através dos oleodutos subaquáticos. Diante da complexidade dessas operações subaquáticas, a higidez das instalações só pode ser garantida mediante um programa de inspeção adequado. Esse programa deve fornecer à empresa informações rápidas sobre possíveis situações perigosas ou danos causados pela mobilidade do fundo do mar, corrosão ou atividades humanas, como tráfego marítimo ou pesca.

Atualmente, para realizar tarefas de vigilância e inspeção em oleodutos subaquáticos, são utilizadas câmeras de vídeo acopladas a veículos operados remotamente - *Remotely Operated Vehicles (ROV)* (vide Figura 1), controlados da superfície por operadores treinados. Esses veículos operam sob a superfície do mar, realizando operações de inspeção, busca, recuperação, reparo e manutenção de estruturas, além de pesquisa oceanográfica e de engenharia (2)

Figura 1: ROV - Veículo Operado Remotamente.

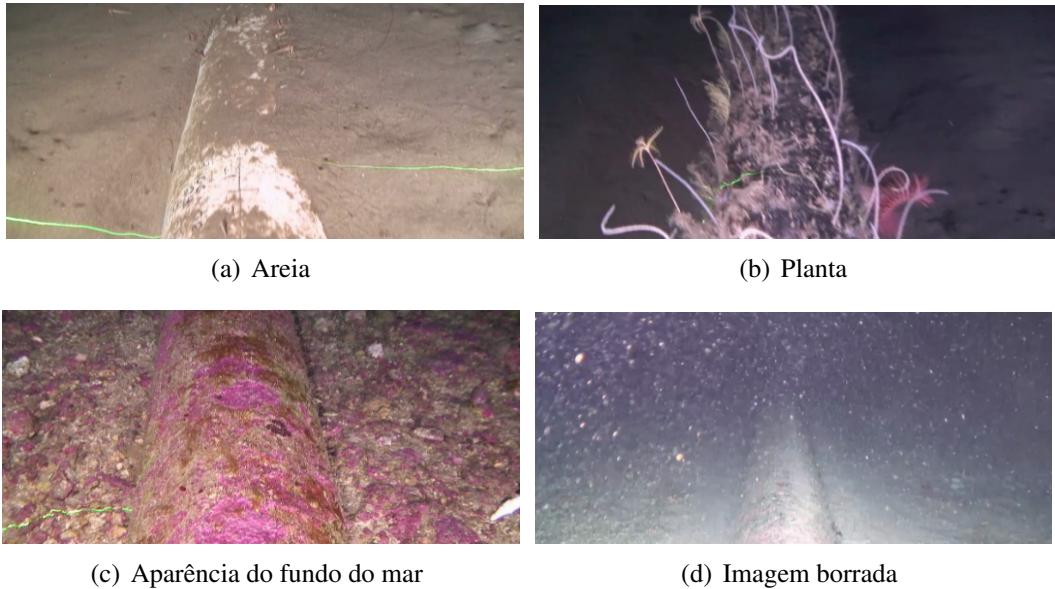


Tradicionalmente, a detecção de danos em dutos subaquáticos é feita por uma pessoa especializada, através da inspeção visual, durante ou após a tarefa de vigilância, o que pode ser um processo tedioso e levar a análises incorretas devido à falta de concentração do operador. Automatizar qualquer parte do processo de inspeção de oleodutos subaquáticos pode trazer melhorias significativas na manutenção dessas instalações, tanto em relação à precisão quanto em relação ao tempo e custos envolvidos. A utilização de tecnologias automatizadas pode reduzir a necessidade de inspeções visuais

realizadas por operadores especializados, o que pode levar a um processo mais eficiente e menos suscetível a erros humanos. Além disso, a automação pode permitir a análise de dados em tempo real e o monitoramento contínuo, contribuindo para a prevenção de falhas e redução de riscos de danos e vazamentos.

Diante desse cenário, torna-se essencial desenvolver processos automatizados para a manutenção de oleodutos subaquáticos. Um grande problema é que a inspeção visual pode ser prejudicada em alguns pontos devido a fatores como: areia cobrindo o duto, vegetação marinha, camuflagem com o fundo do mar e imagens distorcidas causadas pela instabilidade do ROV. Esses obstáculos podem ser observados na Figura 2.

Figura 2: Imagens de Oleodutos



Levando-se em consideração estes aspectos, faz-se necessário uma ferramenta capaz de analisar imagens geradas via ROV, para detectar uma possível falha na estrutura dos dutos, de forma automatizada. Consequentemente, é preciso explorar soluções tecnológicas capazes de superar esses obstáculos e garantir a eficiência e precisão das inspeções.

1.1 Tema

O presente trabalho propõe a utilização de redes neurais para a segmentação de oleodutos subaquáticos e algoritmos de clusterização para identificação de falhas nestes oleodutos, a fim de aprimorar o processo de extração de petróleo na etapa de inspeção subaquática.

Inicialmente, busca-se identificar e segmentar oleodutos subaquáticos utilizando redes neurais convolucionais (CNN), testando diferentes configurações, apresentadas na seção 5, como medida de comparação de desempenho. A partir da etapa de segmentação, a extração de características será realizada para a clusterização de anomalias nos dutos.

O objetivo principal deste trabalho é criar uma arquitetura capaz de auxiliar na detecção de possíveis anomalias que possam afetar os oleodutos, visando à automatização de um processo que hoje é realizado manualmente. Com a implementação dessa arquitetura, espera-se aumentar a eficiência e a eficácia das inspeções subaquáticas, além de reduzir possíveis erros e custos associados a processos manuais.

1.2 Delimitação

O escopo deste trabalho abrange a análise de oleodutos subaquáticos inflexíveis, utilizando conjuntos de dados privados e sigilosos fornecidos pela Petrobras em parceria com a UFRJ. A proposta é desenvolver uma ferramenta baseada em técnicas de visão computacional e aprendizado de máquina capaz de detectar anomalias de forma automatizada, tornando o processo de inspeção mais eficiente e objetivo.

Para abordar o desafio de segmentação dos oleodutos, escolhemos utilizar a rede *Mask R-CNN*, que foi implementada por meio da ferramenta *Detectron 2*. Essa escolha se baseia no fato de que o *Detectron 2* oferece algoritmos *state-of-the-art* de detecção e segmentação de objetos, por meio de uma biblioteca de pesquisa de inteligência artificial do *Facebook*, tornando-o uma opção adequada para lidar com o nosso problema específico. A utilização do *Mask R-CNN* nos permite obter resultados precisos ao identificar os oleodutos e segmentá-los nas imagens subaquáticas. De acordo com (3), em sua seção de *notes* da documentação, o *Detectron 2* é cerca de 4.5 vezes mais eficiente do que a clássica implementação da rede *Mask R-CNN* em termos de *throughput* (em imagens/segundo).

Como métrica de avaliação da qualidade das segmentações do duto, utilizamos *average precision* (AP), calculada em diferentes intervalos de IoU (Interseção sobre União), presente na API do framework COCO (*Common Objects in Context*). Essa métrica nos fornece uma medida objetiva da sobreposição entre as regiões preditas pelo modelo e as regiões verdadeiras presentes na base de dados (4). Ao empregar essa métrica, somos capazes de avaliar o desempenho do modelo de segmentação de forma quantitativa, obtendo uma medida precisa de sua capacidade de detectar corretamente os oleodutos. Essa abordagem nos permite tomar decisões informadas durante o processo de desenvolvimento e ajuste do sistema de detecção, buscando constantemente aprimorar os resultados e otimizar a detecção dos oleodutos em imagens subaquáticas.

Na etapa de clusterização, escolhemos utilizar os algoritmos *k-means* e *fuzzy k-means*. Essa escolha se baseia no fato de que o algoritmo *k-means* é amplamente utilizado em tarefas de clusterização devido à sua simplicidade e eficiência computacional. Como alternativa ao *k-means*, optamos por usar o *fuzzy k-means*, pois permite uma clusterização mais flexível e suave, levando em consideração a incerteza e sobreposição nos dados.

Ao optar pelo uso dos algoritmos de clusterização *k-means* e *fuzzy k-means*, nossa intenção é aproveitar as características distintas desses métodos e sua habilidade em identificar padrões e estruturas na segmentação de oleodutos subaquáticos. Nossa abordagem envolve a utilização de uma métrica

quantitativa para mensurar a porcentagem de imagens, tanto anômalas quanto normais, agrupadas em cada *cluster*. Dessa forma, poderemos obter uma avaliação objetiva e comparativa do desempenho dos algoritmos na clusterização das anomalias nos oleodutos subaquáticos.

1.3 Justificativa

ROVs são amplamente utilizados para a inspeção de oleodutos subaquáticos e cabos de transmissão de energia (5). Embora estejamos conectados por mais de 3,5 milhões de quilômetros de oleodutos sob os oceanos, estima-se que esse número tenha crescido 12,2% em 2022 (6), a maioria dessas instalações é inspecionada apenas uma ou duas vezes por ano, em um cronograma regular (7).

Nesse cenário, qualquer vazamento potencial pode causar danos aos equipamentos utilizados e, especialmente, a áreas ambientalmente sensíveis, seus ecossistemas e espécies marinhas. Um exemplo extremo é o acidente do derramamento de petróleo do Golfo (*Deepwater Horizon*), que ocorreu em 2010, onde estima-se que 3,19 milhões de barris de petróleo vazaram na área do Golfo (8).

Além disso, (9) mostrou que os gastos globais com ROV aumentaram de 1,3 bilhão de dólares em 2013 para 1,8 bilhão de dólares em 2017, com o total de dias de operação de ROV aumentando de 120.000 para 140.000 no mesmo período. O estudo concluiu que a redução no custo de operação do ROV é essencial para garantir a continuação do desenvolvimento de energia *offshore*.

Em virtude dos fatos mencionados, o presente trabalho tem como objetivo analisar e identificar possíveis falhas, em imagens de oleodutos subaquáticos capturadas por ROVs, utilizando técnicas de visão computacional. A escolha pela análise de imagens de oleodutos subaquáticos deu-se pela relevância em se manter a integridade e segurança dessas estruturas críticas. Para detecção de falhas usamos uma abordagem baseada em técnicas de visão computacional, visando a segmentação dos oleodutos e clusterização de eventuais falhas presentes. Por meio dessas técnicas, é possível identificar falhas com eficiência, contribuindo para a prevenção de possíveis danos ambientais e prejuízos financeiros. Portanto, os resultados deste estudo podem ser aplicados pelos operadores de ROVs, técnicos e pesquisadores como base para melhoria da inspeção de oleodutos subaquáticos e para prevenção de possíveis problemas.

1.4 Objetivos

O objetivo deste trabalho é criar uma arquitetura capaz de auxiliar no processo de extração de petróleo, especificamente na etapa de inspeção subaquática, por meio da utilização de técnicas de *machine learning* e visão computacional para identificar possíveis falhas nos oleodutos. O propósito é automatizar o processo que antes era realizado manualmente e aprimorar a eficiência e precisão do mesmo.

Para avaliar o desempenho da arquitetura proposta, serão realizadas duas etapas: segmentação e clusterização. A segmentação consistirá em isolar o oleoduto presente na imagem de entrada

da rede, para que a etapa de extração de características e clusterização possa ser realizada posteriormente. Na clusterização, nosso objetivo é identificar eventuais anomalias presentes no duto. Por fim, esse trabalho foi desenvolvido através de uma base de dados privada, cedida por uma parceria com a Petrobrás, mas que pode ser replicável em qualquer outra base de dados disponíveis (ajustados os devidos parâmetros).

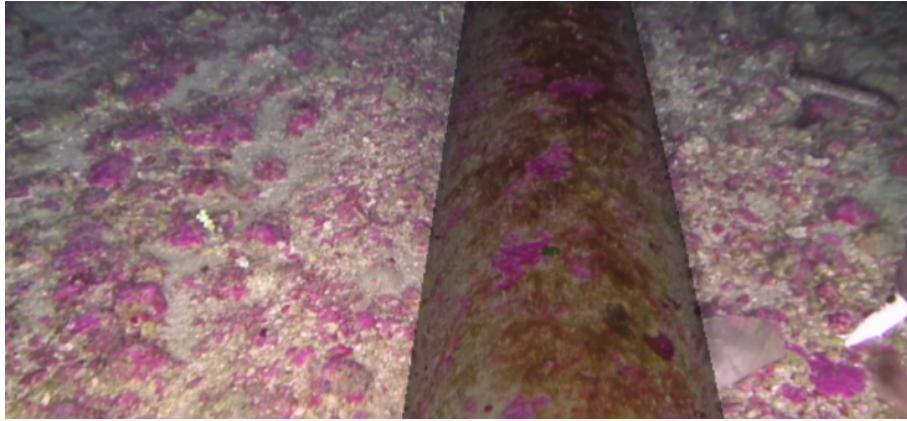
1.5 Metodologia

Para implementar este trabalho, foi necessária a construção de uma ampla base de dados para treinar a rede na etapa de segmentação, composta por 34.323 *frames* anotados manualmente. Para isso, utilizou-se um programa desenvolvido em Python pelo doutorando da COPE, Roberto Esteban Campos Ruiz. Esse programa extrai *frames* do vídeo obtido pelo ROV por meio de métodos do OpenCV e os exibe em uma interface gráfica, ilustrada na Figura 3. Em seguida, o responsável pelas anotações delimita a área onde se encontra o oleoduto por meio de cliques, e as informações são armazenadas em um arquivo CSV. Esse conjunto de anotações representa o *ground truth* da localização dos oleodutos 4, anotados *frame a frame* (10).

Figura 3: Exemplo da Interface



Figura 4: Em preto, exemplo de anotação da base de dados



Com a base de dados construída, iniciamos a etapa de segmentação. As anotações que representam o *ground truth* do oleoduto servem de entrada para o treinamento da ferramenta *Detectron 2* para identificação dos *pixels* que representam o oleoduto. Separamos as anotações em 3 conjuntos: treino, teste e validação.

A partir desse resultado, é feita a extração de características via VGG16 e, por fim, uma clusterização de anomalias via *k-means* e *fuzzy k-means*.

1.6 Descrição

Este trabalho está organizado da seguinte forma: o segundo capítulo possui uma abordagem sobre os principais conceitos envolvidos para a compreensão do trabalho; o terceiro capítulo relata alguns trabalhos relacionados encontrados na literatura; o quarto capítulo demonstra a metodologia de trabalho desenvolvida; o quinto capítulo apresenta os resultados encontrados através da análise realizada; o sexto e último capítulo exibe as conclusões e ideias para os trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo são apresentados os conceitos mais relevantes para o devido entendimento deste trabalho. Serão apresentados: I) os principais conceitos relacionados a visão computacional, II) redes neurais, III) CNNs, IV) *feature extraction* e V) clusterização não supervisionada.

2.1 Visão computacional

De acordo com (11), “a visão computacional denota um novo campo na inteligência artificial, centrado em estudos teóricos do processamento de informações visuais. Seus dois principais objetivos são: desenvolver sistemas de compreensão de imagens, que automaticamente constroem descrições de cenas a partir de dados de entrada de imagens, e compreender a visão humana”. Na prática, isso significa que a tecnologia pode não apenas capturar imagens, mas também separá-las, classificá-las e agrupá-las de acordo com um padrão pré-estabelecido.

Existem diversas aplicações para problemas do mundo real envolvendo visão computacional, e é por isso que as tecnologias se tornaram comuns em muitos campos. Um dos usos mais populares dessas tecnologias é na medicina, em tarefas de assistência de diagnóstico (12), detecção de células vermelhas na corrente sanguínea (13), recuperação de informações de raios-X, ultrassom e ressonância magnética.

Outro caso bem conhecido de aplicação da visão computacional são carros autônomos. De acordo com (14) “Uma das tecnologias mais importantes em veículos autônomos é a visão computacional, a tecnologia que “pode fazer as máquinas enxergarem”. Ela envolve a captura de imagens com diferentes tipos de câmeras, processamento dessas imagens e extração de informações do mundo real, geralmente para análise ou tomada de decisão”.

Dito isso, é importante entender os princípios básicos do processamento de imagem para que possamos compreender como os sistemas de visão computacional funcionam. De acordo com (15), “o processamento de imagens trata principalmente da aquisição de imagens, aprimoramento de imagens, segmentação de imagens, *feature extraction*, classificação de imagens, entre outros”. Essas técnicas podem ser aplicadas em diferentes etapas do *pipeline* de processamento de imagem, que envolve desde a captura da imagem até a extração de informações relevantes.

Ao entender esses princípios básicos de processamento de imagem, é possível entender melhor como as tecnologias de visão computacional funcionam e como elas podem ser aplicadas em diferentes campos, como medicina, transporte, manufatura e outros.

Nesse trabalho, utilizamos segmentação da imagem, que divide a imagem em regiões de interesse e também *feature extraction*, para realizar a extração de informações relevantes para a clusterização de anomalias. As técnicas de segmentação e de extração de características foram feitas através de redes neurais, que são particularmente úteis em tarefas de visão computacional, porque podem aprender representações complexas e hierárquicas dos dados de imagem, o que permite que as redes

sejam capazes de extrair características discriminativas dos dados brutos. A clusterização foi realizada utilizando os algoritmos clássicos da literatura *k-means* e *fuzzy k-means*.

2.2 Redes neurais

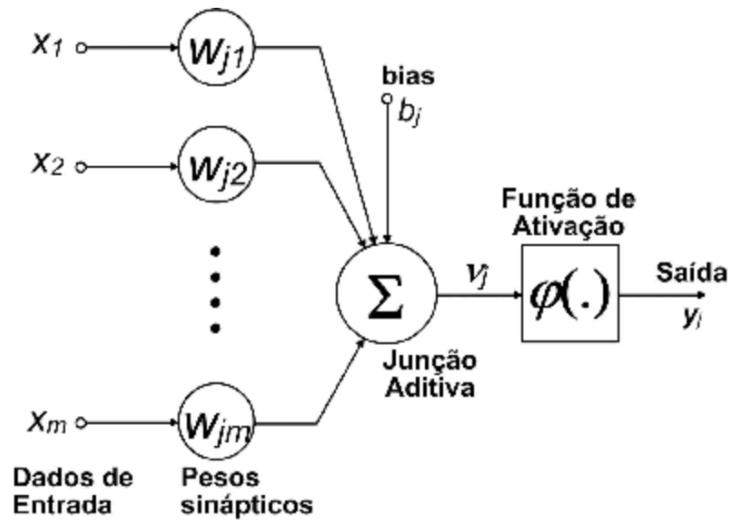
2.2.1 Single-layer neural network

As redes neurais são uma classe de algoritmos de aprendizado de máquina, inspirados na estrutura e funcionamento do cérebro humano. Essas redes são compostas por camadas de neurônios interconectados, nas quais cada neurônio realiza operações matemáticas em seus sinais de entrada e transmite a informação processada para os neurônios da camada seguinte.

De acordo com (16), “uma rede neural artificial, frequentemente chamada apenas de rede neural, é um modelo matemático inspirado nas redes neurais biológicas. Uma rede neural consiste em um grupo interconectado de neurônios artificiais e processa informações usando uma abordagem de computação baseada em conexões.”

Essencialmente, cada neurônio artificial recebe várias entradas, que são multiplicadas por pesos, que representam o fluxo de informação. Esses pesos são calculados por uma função matemática, que determina a ativação do neurônio. Outra função é usada para calcular a saída do neurônio, que pode depender de um determinado limiar. Os neurônios da rede somam suas entradas ponderadas, sendo que os neurônios de entrada possuem apenas uma entrada e sua saída é simplesmente a entrada multiplicada por um peso.

Figura 5: Modelo neuronal proposto por McCulloch e Pitts.



Fonte: extraído de (17) .

No modelo de McCulloch-Pitts 5, o sinal x_i na entrada i é primeiro multiplicado por um parâmetro w_i conhecido como peso sináptico (que é análogo à força sináptica em uma rede biológica)

e depois é somado a todos os outros sinais de entrada ponderados, para obtermos uma saída total na forma

$$v = b_0 + \sum_{i=1}^N w_i x_i \quad (1)$$

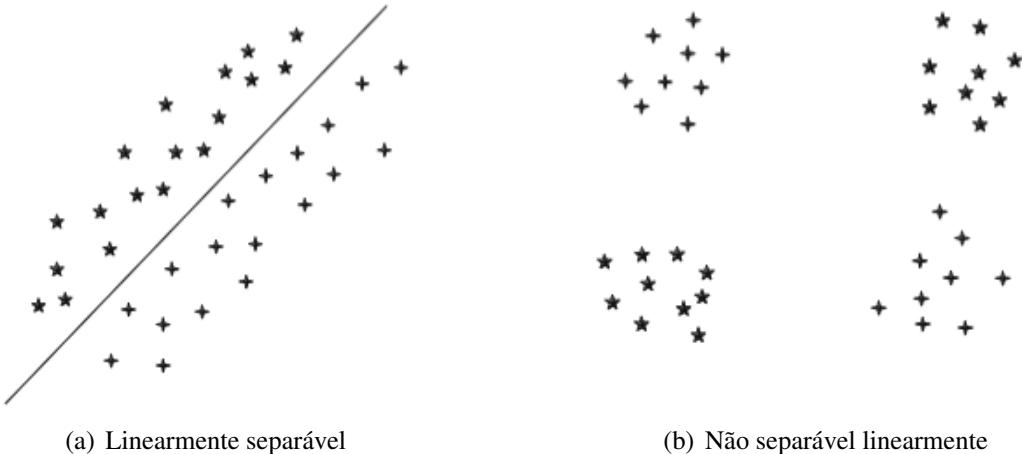
$$Y = \phi(v) \quad (2)$$

na qual o parâmetro b_0 é chamado de *bias* (e corresponde ao limiar de ativação em um neurônio biológico). Em relação ao *bias*, trata-se de um parâmetro extra e invariável, que tem a função de realizar alterar a saída do somador por uma constante, geralmente permitindo que a saída a uma excitação nula seja não nula.

O processamento envolve uma operação linear que combina as entradas, onde cada entrada x_i é atribuída a um peso sináptico w_i . Entretanto, a saída v resultante da Equação 1 deve conter características não lineares. Para esse propósito, é necessário aplicar uma função de ativação na saída Y do neurônio, conforme a Equação 2, para representar como a entrada interna e o estado atual influenciam na definição do próximo estado de ativação. Além disso, quando se trata de uma tarefa com múltiplas saídas, podemos facilmente expandir o modelo de perceptron adicionando várias unidades de saída $y_{k=1}^K$, uma para cada classe, com seus respectivos pesos sinápticos, agora uma matriz de pesos, $\{W_{ki}\}_{i=1,\dots,N; k=1,\dots,K}$.

Essa rede neural artificial é também referida na literatura como *Perceptron* (18). O *Perceptron* contém apenas uma camada de entrada e um nó de saída e é comumente citado como um TLU (*Threshold Logical Unit*) por descrever os dados, dependendo do resultado do somatório dos pesos sinápticos, se é maior ou menor que um determinado limiar (*threshold*) (19). Assim, o *Perceptron* é estritamente equivalente a um discriminante linear e frequentemente usado como um classificador binário (vide Figura 6(a)).

Figura 6: Exemplo de linearmente separável e não separável em duas classes de dados.



2.2.2 Multi-layer neural network

Uma das principais limitações do uso de *Single-layer neural networks* vem de sua separação linear para uma tarefa de classificação, apesar do uso de uma função de ativação não linear (20). O algoritmo do perceptron é bom para classificar conjuntos de dados como o mostrado na Figura 6 (a), quando os dados são linearmente separáveis. Por outro lado, tende a apresentar um desempenho ruim em conjuntos de dados como o mostrado no lado direito da Figura 6 (b). Esse exemplo ilustra a limitação intrínseca do modelo perceptron, que exige o uso de arquiteturas neurais mais complexas.

Essa limitação pode ser evitada com o uso de uma rede *Multi-layer neural network*, proposto por (21). As redes neurais *multi-layers* contêm múltiplas camadas computacionais. As camadas intermediárias adicionais (entre entrada e saída) são chamadas de camadas ocultas (*hidden layers*) porque os cálculos realizados não são visíveis para o usuário. Para uma rede de duas camadas neurais, que também é conhecida como um multi-layer perceptron, nós podemos escrever sua composição como

$$y_k = \varphi^{(2)} \left(\sum_{j=1}^M W_{kj} \varphi^{(1)} \left(\sum_{i=1}^N W_{ji} x_i \right) \right), \quad (3)$$

na qual o índice subscrito denota o índice da camada, M representa o número de neurônios na camada escondida e N , o número de neurônios na camada de entrada. Note que o termo de bias foi omitido por simplicidade. Podemos expandir o número de camadas escondidas até a $(L - 1)$ por meio da composição:

$$y_k = \varphi^{(L)} \left(\sum_l W_{kl}^L \varphi^{(L-1)} \left(\sum_m W_{lm} \varphi^{(L-2)} \left(\dots \varphi^{(1)} \sum_i W_{ji}^1 x_i \right) \right) \right) \quad (4)$$

Em teoria, é possível aplicar diferentes tipos de funções de ativação para diferentes camadas da rede. Entretanto, é comum aplicar o mesmo tipo de função de ativação para as camadas ocultas na literatura. No entanto, tais funções devem ser não lineares, já que, a complexidade de funções lineares é limitada; assim, elas não têm a capacidade de aprender e reconhecer mapeamentos complexos a partir de dados (22).

2.2.3 Aprendizado em redes feed-forward

Em termos de aprendizado de redes, existem dois problemas fundamentais: aprendizado da arquitetura da rede e aprendizado dos parâmetros da rede. Embora o aprendizado da arquitetura da rede ainda seja uma questão em aberto, quase sempre projetado empiricamente (21), existe um algoritmo eficiente para o aprendizado dos parâmetros da rede, conforme circunstanciado abaixo.

O problema de aprendizado de parâmetros em redes neurais do tipo feedforward pode ser formulado como minimização de uma função de erro. Para isso, dado um conjunto de dados de treinamento, $\{x_n, t_n\}_{n=1}^N$, composto por observações e vetores de indicação de classe (com codificação *one-of-K*), onde $x_n \in \mathbb{R}^D$ representa uma observação e $t_n \in \{0, 1\}^K$ representa um vetor indicador de uma classe codificada no sistema *one-of-K*, ou seja, para uma classe k , apenas o k -ésimo elemento do vetor

t_n é 1 e todos os outros elementos são 0. Para classificação de K classes, é comum usar uma função de custo de entropia cruzada (23) definida como:

$$E(\theta) = -\frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk}. \quad (5)$$

Por inspeção, podemos constatar que a função de erro descrita na Equação 5 não é linear e nem convexa, o que significa que não existe uma solução analítica para o conjunto de parâmetros θ que minimiza a Equação 5. Em vez disso, normalmente um algoritmo de *gradient descent* (descida de gradiente) é utilizado, de modo a empreender uma atualização iterativa dos parâmetros. Para utilizar tal algoritmo, é preciso ter um meio de computar o gradiente $\nabla E(\theta)$ avaliado no conjunto de parâmetros θ .

Para uma rede neural feedforward, o gradiente pode ser eficientemente calculado por meio do *backpropagation* de erro (21). O algoritmo de *backpropagation* tem como ideia chave propagar os erros da camada de saída de volta para a camada de entrada, utilizando a regra da cadeia. Em uma rede neural de L camadas, a derivada da função de erro E em relação aos parâmetros da camada l , ou seja, $W(l)$, pode ser estimada da seguinte forma:

$$\frac{\partial E}{\partial W^{(l)}} = \frac{\partial E}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(l+2)}}{\partial a^{(l+1)}} \frac{\partial a^{(l+1)}}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial W^{(l)}}, \quad (6)$$

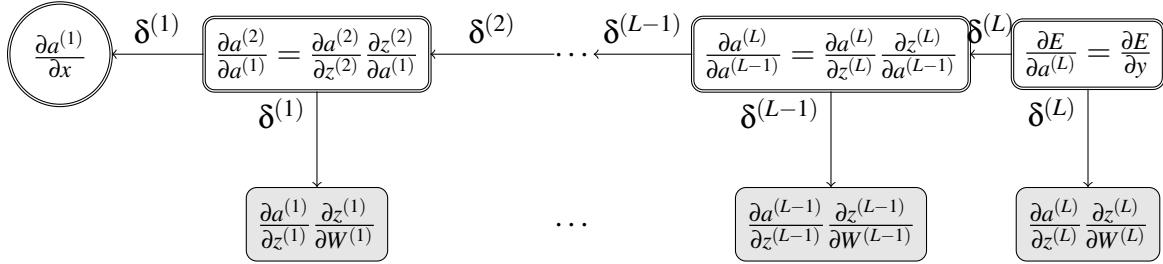
onde $z^{(l)}$ e $a^{(l)}$ respectivamente denotam o vetor de pré-ativação e o vetor de ativação da camada l e $a^{(L)} = y$. Note que $\frac{\partial E}{\partial a^{(l)}}$, ou de forma equivalente $\frac{\partial E}{\partial y}$, corresponde ao erro computado na camada de saída. Para a estimativa do gradiente de uma função de erro E em relação ao parâmetro $W(l)$, é utilizada a propagação do erro da camada de saída através das camadas, na forma de $\frac{\partial a^{(k+1)}}{\partial a^{(k)}}$, $k = l, l+1, \dots, L-1$, juntamente com $\frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial W^{(l)}}$. A fração $\frac{\partial a^{(k+1)}}{\partial a^{(k)}}$ também pode ser calculada de maneira similar como:

$$\frac{\partial a^{(L+1)}}{\partial a^{(l)}} = \frac{\partial a^{(L+1)}}{\partial z^{(L+1)}} \frac{\partial z^{(L+1)}}{\partial a^{(L)}} \quad (7)$$

$$= f' \left(z^{(L)} \right) \left(W^{(L+1)} \right)^T, \quad (8)$$

onde f' representa o gradiente da função de ativação $f(l)$ em relação ao vetor de pré-ativação $z(l)$.

Figura 7: Representação gráfica do algoritmo de backpropagation.



A Figura 7 ilustra um diagrama que representa graficamente como estimar os gradientes de uma função de erro em relação aos parâmetros de uma rede neural de L camadas. Esse processo é baseado nas Equações 7 e 8 e é conhecido como algoritmo de *backpropagation*. No diagrama, cada nó realiza cálculos ao passar mensagens de erro através de setas, $\delta^{(l)}$, começando do nó superior mais à direita. Quando um nó de uma camada l recebe uma mensagem de erro, $\delta^{(l+1)}$, de um nó da camada seguinte, $l + 1$, ele atualiza a sua própria mensagem de erro, $\delta^{(l)}$, computando:

$$\delta^{(l)} = \frac{\partial a^{(L+1)}}{\partial z^{(L+1)}} \odot \left(\frac{\partial z^{(L+1)}}{\partial a^{(l)}} \cdot \delta^{(l+1)} \right) \quad (9)$$

$$= f'(z^{(L)}) \odot \left((W^{(L+1)})^T \cdot \delta^{(l+1)} \right) \quad (10)$$

onde \odot representa uma multiplicação elemento a elemento.

Uma vez que obtemos o vetor gradiente de todas as camadas, ou seja, os nós sombreados do grafo da figura 7, o conjunto de parâmetros $W = [W(1) \dots W(l) \dots W(L)]$ pode ser atualizado de acordo com a *delta rule* ou "*Widrow-Hoff learning rule*" (24):

$$W^{(i+1)} = W^{(i)} - \eta \nabla E(W^{(i)}) \quad (11)$$

onde $\nabla E(W) = [\frac{\partial E}{\partial W^{(1)}} \dots \frac{\partial E}{\partial W^{(l)}} \dots \frac{\partial E}{\partial W^{(L)}}]$ é obtido via retropropagação (*backpropagation*), η é a taxa de aprendizado, e i sinaliza o número da iteração. O processo de atualização é repetido até alcançarmos a convergência ou o número predefinido de iterações.

Quanto à atualização dos parâmetros na Equação 12, existem três variantes do método *gradient descent*, que diferem na quantidade de dados que usamos para calcular o gradiente da função objetivo. Dependendo da quantidade de dados, existe um *trade-off* entre a precisão da atualização dos parâmetros e o tempo necessário para realizar a atualização (25).

O algoritmo de *batch gradient descent* atualiza os parâmetros baseado nos gradientes (∇E) avaliados em todas as amostras de treinamento. Enquanto isso, o gradiente descendente estocástico (*stochastic gradient descent*) atualiza sequencialmente os parâmetros de peso computando o gradiente com base em uma amostra por vez. Quando se trata de aprendizado em grande escala, como em *deep learning*, é recomendado aplicar o gradiente descendente estocástico (26). Como um *trade-off*

entre o gradiente em lote e o gradiente estocástico, o método *mini-batch gradient descent*, que calcula e atualiza os parâmetros com base em um pequeno conjunto de amostras, é comumente usado na literatura (27).

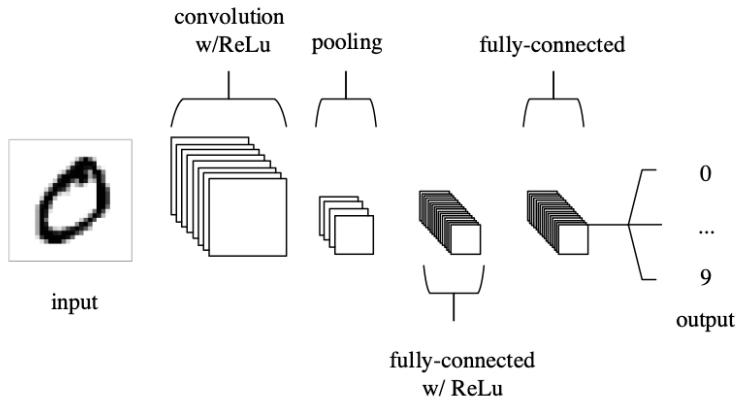
2.3 Convolutional Neural Network

Uma das maiores limitações das formas tradicionais de Redes Neurais Artificiais (ANN) é que elas tendem a enfrentar dificuldades com a complexidade computacional necessária para processar imagens em alta resolução ou grande quantidade de imagens. Assim, CNNs (*Convolutional Neural Networks*) focam principalmente na premissa de que a entrada será composta por imagens. Isso direciona a arquitetura a ser configurada de forma a melhor atender à necessidade de lidar com esse tipo específico de dados. (28), (29). Diferentemente das redes neurais multicamadas convencionais, uma CNN utiliza o compartilhamento de pesos de forma extensiva para reduzir os graus de liberdade dos modelos. A camada de *pooling* ajuda a reduzir o tempo de computação e a desenvolver gradualmente a invariância espacial.

A CNN é um tipo de rede neural *feedforward* que utiliza estruturas de convolução para extrair características dos dados. Diferente dos métodos tradicionais de extração de características (30), (31), (32), a rede CNN não precisa extraír as características manualmente. A arquitetura da CNN é inspirada na percepção visual (33).

As CNNs consistem em três tipos de camadas: convolucionais, de *pooling* e totalmente conectadas (*fully-connected*). Ao empilhar essas camadas, cria-se a arquitetura de uma CNN. A Figura 8 apresenta uma representação simplificada da arquitetura de uma CNN para classificação de dígitos na famosa base de dados MNIST.

Figura 8: Arquitetura CNN simplificada.

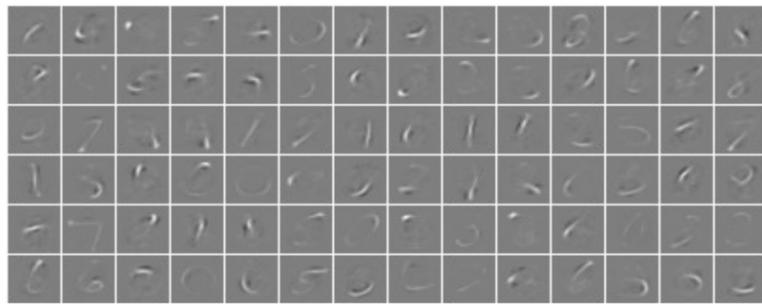


Fonte: extraído de (28).

2.3.1 Camadas de convolução e pooling

A camada de convolução desempenha um papel vital na forma como as CNNs operam (28). Essa camada opera de maneira muito diferente em comparação com as outras camadas da rede neural e não utiliza pesos de conexão e uma soma ponderada. Em vez disso, os parâmetros das camadas estão relacionados ao uso de *kernels* aprendíveis. Esses *kernels* geralmente têm baixa dimensionalidade espacial, mas se estendem por toda a profundidade da entrada. Quando os dados atingem uma camada de convolução, a camada convolui cada filtro ao longo da dimensionalidade espacial da entrada para produzir um mapa de ativação. Esses mapas de ativação podem ser visualizados, como visto na Figura 9.

Figura 9: Ativações obtidas da primeira camada convolucional de uma rede neural convolucional simplificada, após o treinamento no banco de dados MNIST de dígitos escritos à mão.



Fonte: extraído de (28).

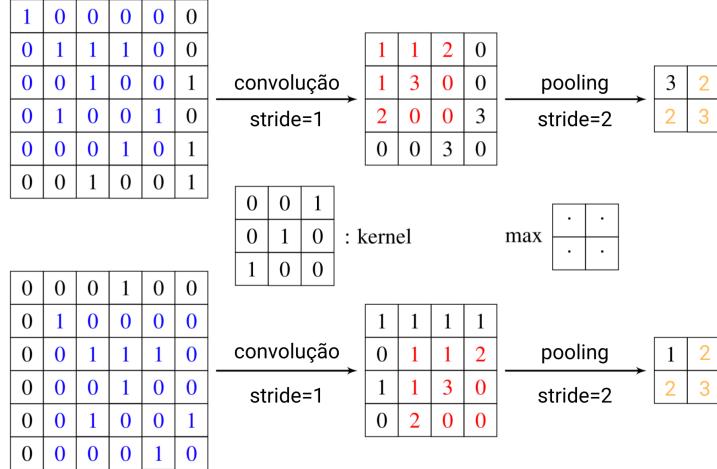
De acordo com (34), o mapa de características destaca as características únicas da imagem original. Essas características locais, em diferentes posições nos mapas de características, são obtidas usando *kernels* (pesos de conexão) aprendíveis $k_{ij}^{(l)}$, ou seja, pesos de conexão entre o *feature map* i na camada $l - 1$ e o *feature map* j na camada l . Especificamente, as unidades da camada de convolução l calculam suas ativações $A_j^{(l)}$ com base apenas em um subconjunto espacialmente contíguo de unidades nos *feature maps* $A_i^{(l-1)}$ da camada anterior $l - 1$ convoluindo os *kernels* $k_{ij}^{(l)}$ da seguinte forma

$$A_j^{(l)} = f \left(\sum_i^{M^{(l-1)}} A_i^{(l-1)} * k_{ij}^{(l)} + b_j^{(l)} \right) \quad (12)$$

na qual $M^{(l-1)}$ denota o número de mapas de características na camada $l - 1$, $*$ denota um operador de convolução, $b_j^{(l)}$ corresponde ao *bias* e $f()$ é uma função de ativação não linear. Devido à conectividade local e compartilhamento de pesos, podemos reduzir significativamente o número de parâmetros em comparação com uma rede neural totalmente conectada, e assim é possível evitar o *overfitting*. Além disso, quando a imagem de entrada é deslocada, a ativação das unidades nos mapas de características também é deslocada pela mesma quantidade, o que permite que uma CNN seja equivariante a pequenos deslocamentos, conforme ilustrado na Fig 10. Na figura, quando os valores dos *pixels* na

imagem de entrada são deslocados um pixel para a direita e um pixel para baixo, as saídas após a convolução também são deslocadas um pixel para a direita e um pixel para baixo.

Figura 10: Exemplo da invariância à translação em uma rede neural convolucional. A entrada no canto inferior esquerdo é uma versão transladada da imagem de entrada no canto superior esquerdo em um pixel para a direita e um pixel para baixo.



O *pooling* é uma técnica típica que muitos outros métodos de processamento de imagens já estão utilizando. A camada de *pooling* reduz o tamanho da imagem, pois combina *pixels* vizinhos de uma determinada área da imagem em um único valor representativo (34). Especificamente, cada mapa de características em uma camada de *pooling* está conectado a um mapa de características na camada de convolução, e cada unidade em um mapa de características da camada de *pooling* é calculada com base em um subconjunto de unidades. No caso do *max-pooling*, um dos tipos mais comuns de métodos de *pooling*, ele divide a imagem em retângulos de sub-regiões e retorna apenas o valor máximo dentro dessa sub-região (35). Outra função importante da camada de *pooling* é proporcionar invariância à translação para pequenos deslocamentos espaciais na entrada. Na Figura 10, enquanto a imagem mais à esquerda na parte inferior é uma versão transladada da imagem mais à esquerda na parte superior, deslocada um *pixel* para a direita e um *pixel* para baixo, suas saídas após as operações de convolução e *pooling* são as mesmas, especialmente para as unidades em laranja.

2.3.2 Otimizador

Segundo (36), “o treinamento de redes neurais é trabalhoso e consome muito tempo, às vezes levando dias ou até semanas para treinar uma rede neural. Isso limita a aplicação de sistemas de aprendizado profundo em diversos campos de pesquisa”. Devido a essas razões, há uma necessidade de velocidade de treinamento eficiente e aprimorada para realizar tais aplicações, especialmente ao considerar redes neurais convolucionais (CNNs). Em CNNs, frequentemente, precisamos otimizar funções não convexas. Métodos matemáticos requerem grande poder computacional, portanto, otimi-

zadores são usados no processo de treinamento para minimizar a função de perda e obter os parâmetros ótimos da rede dentro de um tempo aceitável.

Essa necessidade de treinamento rápido e eficiente é especialmente importante quando consideramos o uso de CNNs em diversas áreas de pesquisa. Os treinamentos podem levar dias ou até semanas, e a capacidade de acelerar esse processo é crucial para permitir a aplicação prática desses modelos em cenários do mundo real.

Entre algumas técnicas para melhorar o processo de otimização, destacam-se o *momentum*, o *Root-mean-square prop* (RMSprop), o *adaptive moment estimation* (Adam) e outros (37). Cada um desses algoritmos possui suas características e propriedades específicas, mas todos compartilham o objetivo de encontrar os parâmetros que minimizam a função de perda de forma eficiente.

Em resumo, os otimizadores são essenciais para superar os desafios computacionais e de tempo associados ao treinamento de CNNs e outras redes neurais. Um exemplo disso pode ser observado no trabalho (38), onde os autores aplicam a um modelo de classificação de imagens de última geração, a técnica de otimização de Normalização em Lote (Batch Normalization) e alcançam a mesma precisão com 14 vezes menos passos de treinamento, superando significativamente o modelo original. De acordo com os autores, “utilizando um conjunto de redes normalizadas em lote, melhoramos o melhor resultado publicado na classificação do ImageNet, alcançando um erro de teste de 4.82% nos top 5, superando a precisão dos avaliadores humanos”.

Nesse contexto, fica evidente a importância dos otimizadores no treinamento de redes neurais, possibilitando uma convergência mais rápida e uma melhora de desempenho. A contínua evolução e desenvolvimento de otimizadores mais eficientes e robustos são fundamentais para impulsionar o avanço da inteligência artificial e do aprendizado profundo.

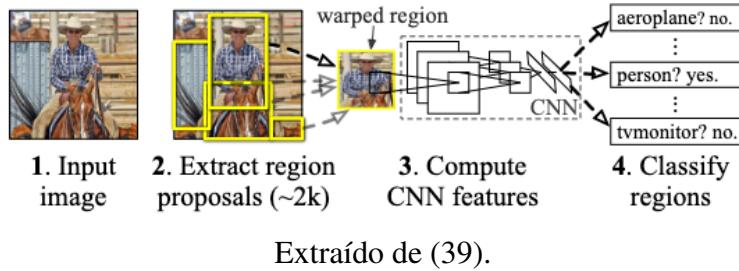
2.4 R-CNN

Ao contrário do problema de classificação de imagens, a detecção requer localizar (provavelmente muitos) objetos dentro de uma imagem (39). As CNNs têm sido usadas dessa forma há pelo menos duas décadas, geralmente em categorias de objetos restritos, como faces (40), (41) e pedestres (42). De acordo com (43), desde os primeiros trabalhos sobre detecção de rosto no final dos anos 90, a estratégia dominante para detecção de objetos em uma cena tem sido a varredura em várias escalas. No entanto, existem alguns pontos insatisfatórios nessa abordagem. Em primeiro lugar, a classificação de uma janela contendo, por exemplo, um cavalo, não é a mesma coisa que segmentar os pixels correspondentes a um cavalo. Portanto, algum pós-processamento seria necessário para alcançar esse objetivo. Em segundo lugar, a natureza força bruta da classificação de janelas não é particularmente atraente devido a sua complexidade computacional. Em terceiro lugar (e isso pode ser mais relevante para alguns do que para outros), ela difere significativamente da natureza da detecção visual humana, na qual a atenção é direcionada a determinadas localizações, em vez de ser uniformemente distribuída em todas as localizações.

Dito isso, o método R-CNN, resolve esse problema de localização de objetos em CNNs operando dentro do paradigma de “reconhecimento usando regiões”, conforme argumentado por (43). Nessa abordagem são geradas cerca de 2000 propostas de região independente de categoria para a imagem de entrada, feita a extração de um vetor de características de comprimento fixo de cada proposta usando uma CNN e, em seguida, a classificação de cada região com SVMs (*Support Vector Machines*) lineares específicos de cada categoria.

A rede de detecção de objetos R-CNN consiste em três módulos (vide Fig. 11). O primeiro gera propostas de região independente de categoria. Essas propostas definem o conjunto de detecções candidatas disponíveis para o detector e são geradas a partir do método de busca seletiva (*selective search*) baseada em trabalhos anteriores de detecção (44), (45). O segundo módulo é uma rede neural convolucional (CNN) que extrai um vetor de características de comprimento fixo de cada região. O terceiro módulo é um conjunto de SVMs lineares específicos de cada classe.

Figura 11: Arquitetura da rede R-CNN.



2.5 Feature Extraction

A extração de características, ou *feature extraction*, desempenha um papel essencial no campo da análise de imagens e aprendizado de máquina. De acordo com (46), a extração de características pode ser definida como “a construção e identificação de um conjunto de variáveis que representam uma observação ou um padrão”, ou seja, consiste em identificar e extrair as informações mais relevantes e distintivas presentes nos dados brutos, a fim de melhorar a representação dos dados e facilitar tarefas subsequentes, como classificação ou detecção de objetos.

Existem várias abordagens e técnicas disponíveis para a extração de características, cada uma com suas próprias vantagens e limitações. No contexto deste trabalho, foi utilizada a arquitetura VGG16 para realizar a extração de características. Portanto, nesta seção do referencial teórico, concentraremos nossa análise e discussão na VGG16, explorando suas características e aplicações relevantes para o nosso estudo.

A VGG16 é uma rede neural convolucional pré-treinada que se destaca por sua capacidade de aprendizado de alto nível em imagens. Ela consiste em várias camadas convolucionais e de *pooling*, seguidas por camadas totalmente conectadas, que fornecem uma representação compacta e discriminativa dos dados de entrada.

A vantagem de utilizar a VGG16 para extração de características é que ela foi treinada em grandes conjuntos de dados de imagens, como o ImageNet, o que lhe permite aprender padrões e características visuais de forma eficaz, mesmo quando os conjuntos de dados de imagem são pequenos (47). Ao usar a VGG16, podemos aproveitar o conhecimento prévio que a rede adquiriu durante o treinamento em tarefas de classificação de imagens em geral.

Durante o processo de extração de características com a VGG16, as imagens de entrada passaram por uma série de camadas convolucionais, que aplicam filtros para detectar diferentes padrões e características em várias escalas. As informações extraídas nessas camadas são então utilizadas para formar uma representação de alto nível da imagem.

Ao utilizar a VGG16 para extração de características, esperamos obter representações ricas e informativas dos dados de entrada, que possam capturar os aspectos mais relevantes das imagens e facilitar a distinção entre diferentes classes ou objetos de interesse.

2.6 Aprendizado não supervisionado e clusterização

Nesta seção, abordaremos o conceito de aprendizado não supervisionado e sua aplicação na técnica de clusterização. De acordo com (48), “o aprendizado não supervisionado tenta extrair estruturas ocultas a partir dos dados brutos. Esse esforço pode ser significativo, porque na maioria das aplicações os dados de entrada estão longe de serem aleatórios; eles são produzidos por processos físicos”. O aprendizado não supervisionado é uma abordagem de aprendizado de máquina em que não são fornecidas informações de rótulo ou classe para o algoritmo durante o treinamento. Em vez disso, o objetivo é descobrir padrões, estruturas ou agrupamentos intrínsecos nos dados.

A clusterização é uma técnica comum de aprendizado não supervisionado que visa agrupar objetos ou amostras de dados similares em *clusters* ou grupos. Esses *clusters* são formados com base em características compartilhadas ou proximidade entre os pontos de dados, permitindo identificar padrões e estruturas subjacentes nos dados.

Neste trabalho, iremos nos limitar a duas técnicas populares de clusterização: *k-means* e *fuzzy k-means*. O *k-means* é um algoritmo clássico de clusterização que visa particionar os dados em *k clusters*, onde *k* é um valor pré-definido pelo usuário. Segundo (49), “o *k-means* padrão é o algoritmo de clusterização particional mais amplamente utilizado”.

O algoritmo começa selecionando *k* centroides iniciais aleatórios e, em seguida, itera entre duas etapas principais: atribuir cada ponto de dados ao *cluster* mais próximo (com base na distância euclidiana, por exemplo) e atualizar os centroides dos *clusters* com base na média dos pontos atribuídos a cada *clusters*. Esse processo é repetido até que ocorra uma convergência e os *clusters* estejam estáveis. O *K-means* utiliza um esquema de realocação iterativa para produzir uma clusterização rígida de *k* maneiras que minimiza localmente a distorção entre os objetos de dados e um conjunto de representantes de *k* *clusters* (49). O passo a passo descrito acima pode ser observado através do pseudo-código 1

Algoritmo 1: Algoritmo de clusterização *K-means*

Input : Data set X , número de *clusters* k

Output: Atribuições dos clusters

- 1 **Passo 1:** Crie uma clusterização inicial arbitrária com $\{\mu_1, \dots, \mu_k\}$ centroides;
 - 2 **repeat**
 - 3 **Passo 2:** Para cada objeto $x_i \in X$:
 - 4 **for** $c = 1$ **to** k **do**
 - 5 | Calcule $\|x_i - \mu_c\|$;
 - 6 **end**
 - 7 Atribua x_i ao *cluster* correspondente com o centroide mais próximo;
 - 8 **Step 3:** Atualize o controide do *cluster*;
 - 9 **until** Critério de parada ser atingido;
-

O *fuzzy k-means* é uma extensão do *k-means* que permite que cada ponto de dados seja atribuído a vários *clusters* com um grau de pertinência ou "fuzziness". Dunn (50), ao apresentar o algoritmo *Fuzzy c-means* (FCM) ou *Fuzzy k-means*, permitiu que objetos pertençam a diferentes *clusters* em diferentes graus, indicados por pesos probabilísticos. Esses pesos podem ser representados como uma matriz V de dimensões $n \times k$, onde $V_{ij} \in [0, 1]$ representa o grau de pertencimento do objeto x_i ao cluster C_j , e $\sum_j V_{ij} = 1$. Como mencionado anteriormente, o objetivo de clusterizar é minimizar a distorção entre os objetos e os centróides, sendo que agora essa distorção é medida pela função de critério fuzzy.

Ao contrário do *k-means*, que atribui cada ponto a um único *cluster*, o *fuzzy k-means* atribui graus de pertinência para cada ponto em relação a cada *cluster*, indicando o quanto fortemente o ponto pertence a cada um dos *clusters*. Essa abordagem permite uma maior flexibilidade na modelagem de dados que podem pertencer a múltiplos grupos simultaneamente.

As técnicas de clusterização pode ser aplicada em diversas áreas, como detecção de anomalias (51), informática genética (52), processamento de conteúdo multimídia (49), análise de poluição do ar (53), segmentação de clientes (54) e processamento de imagens (55), entre outros. Ao explorarmos algoritmos de clusterização, podemos revelar informações valiosas e *insights* ocultos nos dados, auxiliando na tomada de decisões e na compreensão dos padrões subjacentes aos conjuntos de dados.

3 Trabalhos Relacionados

Diversos sistemas têm sido utilizados em operações de dutos e cabos subaquáticos. Para detecção de dutos em longas distâncias e em condições visuais desafiadoras, são empregados sistemas acústicos, como sonares de varredura frontal, sonares de varredura lateral ou ecobatímetros (56), (57), (58). Em condições de visibilidade favoráveis e proximidade maior ao duto, sistemas baseados em câmeras são utilizados. Dentre esses trabalhos, alguns utilizam técnicas como redes neurais e *deep learning* para detecção e classificação de anomalias, (59), (60).

Uma vez que a maioria dos sistemas desenvolvidos para inspeção visual de dutos subaquáticos é realizada por empresas privadas, poucos trabalhos foram publicados na literatura. A seguir, serão revisadas as abordagens mais relevantes baseadas em visão computacional.

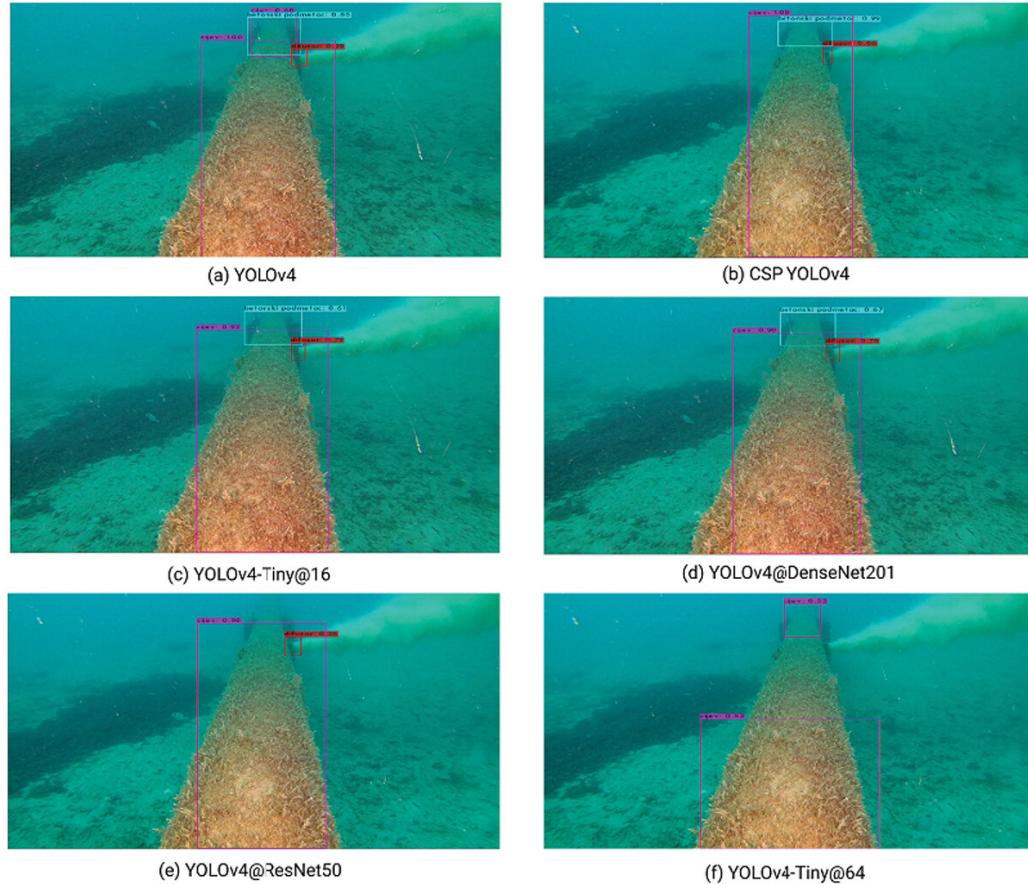
O trabalho (59) utiliza o conceito de *transfer learning* para treinar uma Rede Neural Convolucional Profunda em imagens originais de inspeções de dutos submarinos, com o objetivo de realizar a classificação automática de cinco eventos distintos (duto exposto a possível corrosão, enterramento, vão livre, junção de campo, ânodo). A avaliação de desempenho do *framework* proposto é realizada utilizando conjuntos de dados provenientes de vídeos de inspeção obtidos por um ROV. A rede neural utilizada pelos autores possibilita a classificação de imagens *multi-label*, o que permite identificar simultâneas anomalias em um único quadro (por exemplo, exposição e junção de campo). A técnica de *data augmentation* também é utilizada para aprimorar ainda mais os conjuntos de dados de treinamento, facilitando o tratamento da variabilidade presente nas imagens subaquáticas devido aos desafios criados pelo movimento dinâmico do ROV, brilho e contraste. De acordo com os autores, o desempenho da arquitetura proposta, com o uso de *transfer learning* com ResNet-50, resulta em uma alta taxa de *exact match ratio* e F1 score de 91,9% e 96,6%, respectivamente.

Outro uso de visão computacional subaquática, que também é o foco deste trabalho, é a detecção de dutos submarinos, realizada por (61). Os pesquisadores utilizaram o algoritmo YOLOv3 para localizar o ponto de vazamento de óleo do duto submarino. No trabalho, os autores mapearam 2 tipos de alvos para detecção: duto e ponto de vazamento. O modelo treinado conseguiu alcançar uma precisão de detecção de ponto de vazamento de 77,5% com um tempo de processamento de 36 quadros por segundo. A precisão de detecção para o duto foi de 93,67%.

O trabalho (62) também apresenta o uso de redes neurais e aprendizado profundo para detecção de oleodutos em ambientes subaquáticos. De acordo com o artigo, foram treinados e testados seis detectores de objetos baseados em redes neurais convolucionais de aprendizado profundo: cinco deles são baseados nas arquiteturas *You Only Look Once* (YOLO) (YOLOv4, YOLOv4-Tiny, CSP-YOLOv4, YOLOv4@Resnet, YOLOv4@DenseNet), e um deles na arquitetura *Faster Region-based CNN* (RCNN). O desempenho dos modelos foi avaliado em termos de precisão de detecção, *mean average precision* (mAP) e velocidade de processamento medida em quadros por segundo (FPS), utilizando um conjunto de dados personalizado contendo imagens de dutos submarinos. Os resultados obtidos demonstraram que o YOLOv4 obteve o melhor resultado de mAP de 94,21% no conjunto de

dados testado. Um exemplo de detecção pode ser observado na figura 12.

Figura 12: Performance de detecção dos modelos YOLO

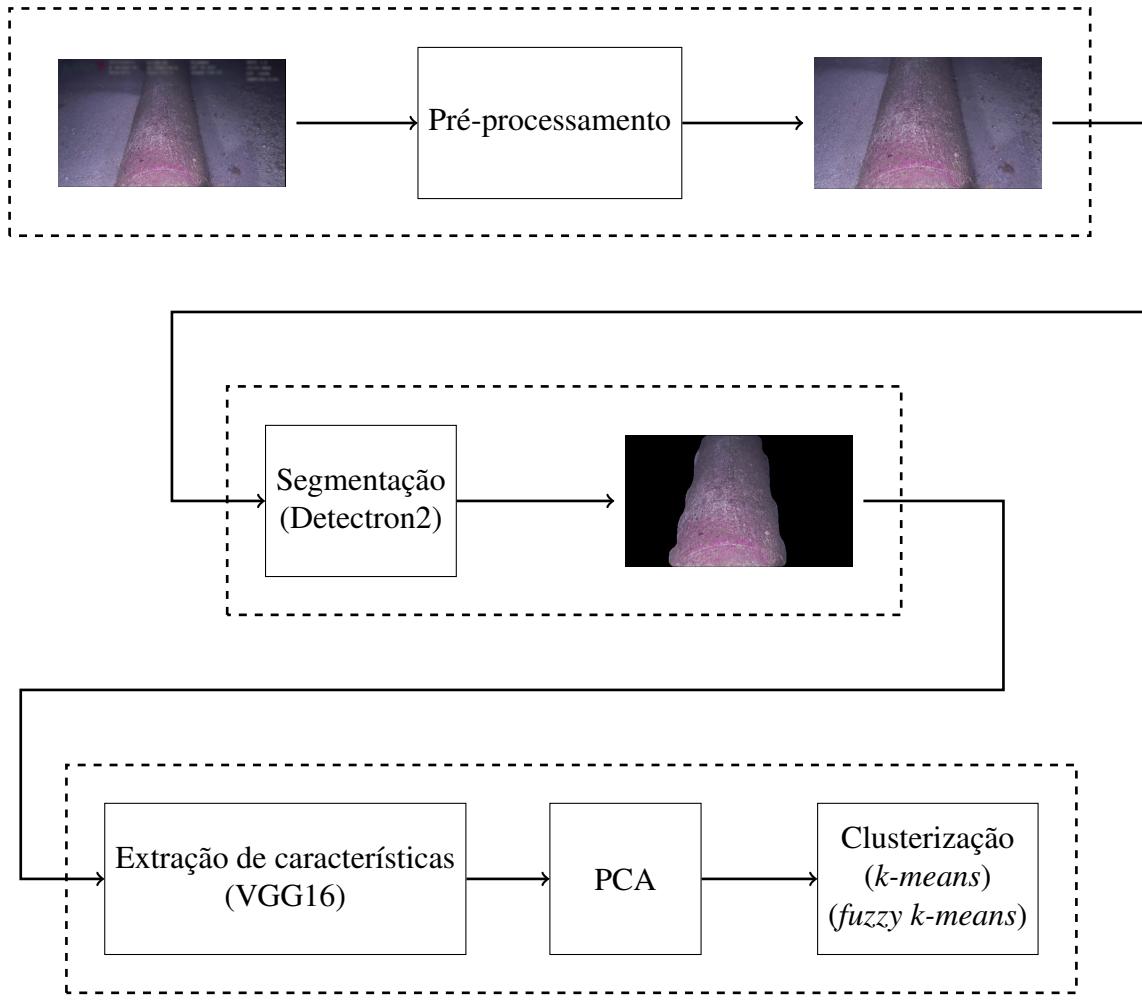


Extraído de (62).

4 Solução Proposta

Neste capítulo serão abordadas as etapas de desenvolvimento do trabalho. A primeira parte apresenta a fase de construção da base de dados para a detecção de oleodutos subaquáticos, o treinamento da base com o uso da ferramenta *Detectron 2* e a geração de máscaras binárias representando a posição do oleoduto. A segunda parte mostra a etapa de extração de características e a clusterização dos oleodutos segmentados (representados por vetores de características e obtidos a partir das máscaras binárias na saída do *Detectron 2*). A arquitetura do sistema proposto pode ser observada na figura 13.

Figura 13: Arquitetura da solução proposta.



4.1 Segmentação do oleoduto

A etapa de segmentação do oleoduto é crucial para a identificação precisa de sua posição e extensão nas imagens obtidas pelo ROV. Nessa etapa, o objetivo é separar o oleoduto do restante da cena, destacando-o de maneira clara e precisa. Para isso, utilizamos o *framework Detectron 2*, que é

uma poderosa ferramenta de segmentação baseada em redes neurais convolucionais.

4.1.1 Construção da base de dados

Primeiramente, foi realizada a construção da base de dados para treinamento do modelo de segmentação. Essa base consiste em um conjunto de 34.323 imagens de oleodutos subaquáticos juntamente com suas anotações, que indicam a posição do oleoduto na imagem. Para realizar essa construção, utilizamos o programa desenvolvido em *Python* pelo doutorando da COPE, Roberto Esteban Campos Ruiz.

Esse programa automatiza o processo de anotação manual das imagens e utiliza a biblioteca *OpenCV* para extrair as imagens *frame a frame* dos vídeos obtidos por meio dos ROVs. Através de uma interface gráfica intuitiva, os *frames* são exibidos, permitindo a anotação manual da posição do oleoduto.

Durante o processo de anotação, o usuário registra apenas as coordenadas superiores e inferiores que compõem o oleoduto, visto que em nossa base de dados trabalhamos somente com oleodutos inflexíveis. Essas informações são então armazenadas em um arquivo CSV, que serve como uma espécie de “mapa” das posições dos oleodutos nas imagens. O programa automatiza esse processo de anotação, proporcionando uma maior agilidade e padronização na criação das futuras máscaras binárias.

É importante destacar que para cada vídeo utilizado, é gerado um arquivo CSV específico contendo as anotações do oleoduto. Isso permite uma organização eficiente dos dados e a possibilidade de realizar análises individualizadas para cada vídeo.

Essa etapa é essencial para o treinamento do modelo de segmentação no *Detectron 2*, possibilitando a geração de máscaras binárias de alta qualidade que representam de forma precisa a posição do oleoduto nas imagens.

Com a base de dados construída, prosseguimos com o pré-processamento da base de dados.

4.1.2 Pré-processamento das imagens

Nesta etapa, realizamos o pré-processamento dos dados antes de fornecê-los à rede de segmentação *Detectron 2*. O objetivo principal é garantir a privacidade e a confidencialidade das informações contidas nas imagens, removendo rótulos e coordenadas sensíveis que não são relevantes para o algoritmo de segmentação.

Para alcançar esse objetivo, aplicamos as seguintes técnicas de pré-processamento:

- Corte de Informações Sensíveis: Identificamos áreas nas imagens que contêm informações sensíveis e as removemos por completo. Essas informações continham dados confidenciais como localização do oleoduto, empresa responsável pela manutenção etc. É importante ressaltar que todos os dados sensíveis foram anonimizados e descartados de forma segura, a fim de preservar a privacidade das partes envolvidas.

- Redimensionamento: Com o objetivo de garantir consistência no tamanho das imagens e otimizar o desempenho da rede de segmentação, realizamos o redimensionamento de todas as imagens para uma resolução padrão. As imagens originais possuíam uma resolução de 640×360 pixels. Após o corte das informações sensíveis e redimensionamento, as imagens passaram a ter uma resolução de 510×295 pixels. Essa abordagem preserva a proporção e a qualidade das imagens, evitando distorções durante o processo de segmentação.

Após a conclusão do pré-processamento dos dados, as imagens resultantes são preparadas para serem alimentadas à rede de segmentação *Detectron 2*.

4.1.3 Criação do catálogo dos dados no formato do *Detectron 2*

Nesta etapa, realizamos a criação do catálogo dos dados no formato do *Detectron 2*, a fim de preparar as imagens e suas respectivas anotações para serem utilizadas pelo algoritmo de segmentação. Esse catálogo é uma estrutura de dados que contém informações sobre as imagens, máscaras binárias e *bounding boxes* correspondentes às posições do oleoduto em cada imagem.

O processo de criação do catálogo dos dados envolve as seguintes etapas:

- Iteração sobre a base de dados: Percorremos a base de dados, que contém as imagens e as anotações das posições do oleoduto. Para cada imagem, utilizamos as coordenadas superiores e inferiores previamente anotadas para calcular a máscara binária que representa a presença do oleoduto na imagem. Essa máscara binária é essencial para treinar o algoritmo de segmentação.
- Cálculo do *bounding box*: Além da máscara binária, calculamos o *bounding box* correspondente à região do oleoduto em cada imagem. O *bounding box* é uma caixa delimitadora retangular que envolve a área da máscara binária. Ele fornece informações sobre a posição e o tamanho aproximado do objeto de interesse na imagem. Essa caixa delimitadora é calculada da seguinte maneira:

$$x_1 = \min(\vec{x}), y_1 = \min(\vec{y}), x_2 = \max(\vec{x}) - \min(\vec{x}) \text{ e } y_2 = \max(\vec{y}) - \min(\vec{y}) \quad (13)$$

no qual \vec{x}, \vec{y} representam os pontos x, y da nossa máscara binária.

- Paralelização com a biblioteca *dask.dataframe*: Para otimizar o processamento e lidar com grandes volumes de dados, utilizamos a biblioteca *dask.dataframe* do Python. Essa biblioteca nos permite realizar as etapas de cálculo da máscara binária e do *bounding box* em paralelo, aproveitando ao máximo os recursos do sistema e acelerando o tempo de processamento. Utilizamos 8 threads para paralelização.

Ao final dessa etapa, obtemos um catálogo dos dados no formato do *Detectron 2*, que contém as informações necessárias para treinar e avaliar o algoritmo de segmentação. Esse catálogo será

utilizado como entrada para o treinamento do modelo e permitirá que o algoritmo aprenda a segmentar o oleoduto de forma precisa e eficiente.

É importante ressaltar que todo o processo de criação do catálogo dos dados foi desenvolvido levando em consideração as boas práticas de processamento de dados e o uso eficiente dos recursos computacionais. A utilização da biblioteca *dask.dataframe* nos possibilitou realizar as etapas de cálculo de forma paralela, melhorando significativamente o desempenho do processo.

4.1.4 Treinamento do modelo *Detectron 2*

Após finalizada a etapa de carregamento dos dados, realizamos o treinamento do modelo de segmentação usando o *Detectron 2*. Segundo (63), “o Detectron2 é atualmente um dos melhores softwares para detecção de objetos e segmentação de instâncias, proposto pelo *Facebook Artificial Intelligence Research (FAIR)*”. Esse framework é baseado em arquiteturas de redes neurais convolucionais avançadas, como a *Mask R-CNN*, que é especialmente adequada para tarefas de segmentação de objetos. Durante o treinamento, o modelo é exposto a um grande número de exemplos de imagens de oleodutos juntamente com suas máscaras binárias correspondentes. O objetivo é fazer com que o modelo aprenda a identificar e segmentar corretamente o oleoduto nas imagens.

A rede *Mask R-CNN* (64) é um método que tem três objetivos principais: gerar caixas delimitadoras (*bounding boxes*) para cada objeto na imagem, classificar cada uma dessas caixas delimitadoras e gerar uma máscara de segmentação para os objetos de interesse. Esse método é uma evolução do Faster-RCNN (65), com a adição de um ramo de segmentação para cada região de interesse (ROI).

O processo envolve a proposição de regiões estratégicas por meio de uma RPN. As caixas delimitadoras geradas passam por um processo de alinhamento para manter a coerência espacial na imagem. Após o processo de alinhamento, o método se divide em duas etapas distintas: a classificação e regressão das caixas delimitadoras, e a geração da máscara de segmentação usando camadas convolucionais. No entanto, neste trabalho, a etapa de classificação das caixas delimitadoras não é utilizada, uma vez que o objetivo é focado exclusivamente na segmentação do oleoduto. Como temos apenas um objeto para segmentar, representando uma única classe, a classificação torna-se dispensável.

Além disso, o algoritmo utiliza uma função de perda que consiste na soma das três perdas para cada objetivo:

$$\text{TotalLoss} = \text{classification}_{loss} + \text{bounding box}_{loss} + \text{mask}_{loss} \quad (14)$$

A perda de classificação e a perda de máscara são medidas utilizando a perda de entropia cruzada, enquanto a perda de caixa delimitadora é medida pela perda L_1 .

Para o treinamento da rede, realizamos uma etapa inicial de *fine tuning* (ajuste dos hiperparâmetros) da ferramenta *Detectron 2*. Os hiperparâmetros configurados são:

- Número de iterações: O número total de iterações que a rede *Detectron 2* irá executar durante o treinamento.

- Iterações *warmup*: O número de iterações iniciais em que o *learning rate* é gradualmente aumentado para evitar instabilidades no treinamento.
- *Learning rate*: A taxa de aprendizado, que determina o tamanho dos ajustes feitos nos pesos da rede durante o treinamento.
- *Gamma*: O fator de redução aplicado ao *learning rate* em épocas específicas do treinamento.
- *Solver steps*: As épocas em que o *learning rate* é ajustado pelo fator *Gamma* durante o treinamento.
- *Eval Period*: O número de iterações entre as avaliações da rede em um conjunto de validação para monitorar seu desempenho.
- *Batch size per image*: O número de amostras de treinamento usadas em cada iteração de atualização dos pesos da rede para uma única imagem.
- *Learning Rate Scheduler Name*: O nome da estratégia usada para ajustar automaticamente o *learning rate* durante o treinamento.
- *Momentum*: Um valor que controla a influência das atualizações de pesos anteriores nas atualizações atuais durante o treinamento.
- *Gradients Clip*: Um valor máximo para limitar a magnitude dos gradientes durante o treinamento, evitando explosão dos gradientes.
- *Norm Type*: O tipo de normalização aplicado aos gradientes durante o treinamento.
- *Clip Value*: Um valor máximo para limitar a magnitude dos gradientes após a normalização.

Vale destacar que os elementos não listados acima permaneceram como padrão.

Após o treinamento, o modelo é capaz de gerar máscaras binárias precisas (vide seção de resultado) que representam a posição do oleoduto em cada imagem. Essas máscaras são utilizadas posteriormente para realizar a clusterização dos oleodutos segmentados.

4.2 Clusterização de Anomalias

A clusterização foi a técnica de aprendizado não supervisionado escolhida para a detecção de anomalias nos oleodutos. A etapa de clusterização permite identificar padrões e diferenças entre os oleodutos, contribuindo para um agrupamento mais preciso e uma análise mais detalhada das anomalias encontradas. Antes de clusterizarmos os oleodutos, extraímos suas *features*, via VGG16, das máscaras binárias preditas pelo *Detectron 2*. Realizamos a extração de características com o objetivo

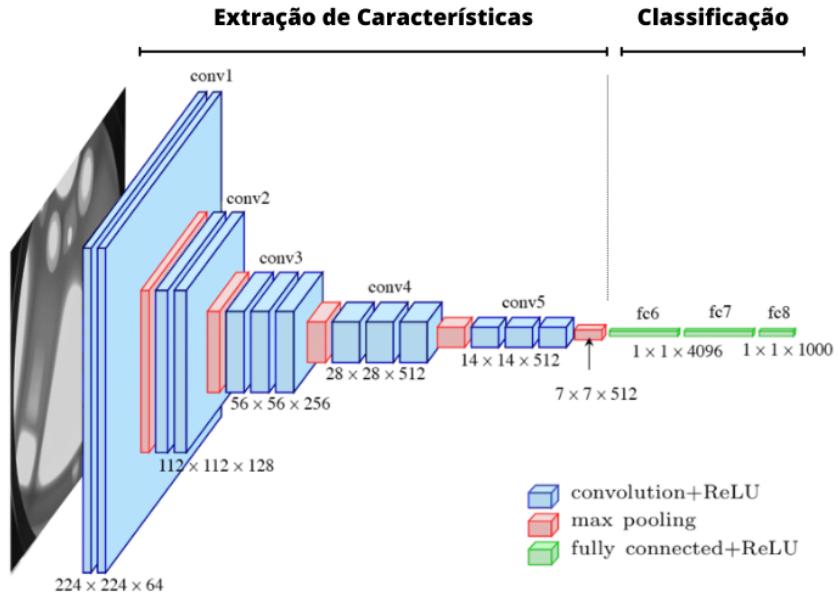
de reduzir computacionalmente a representação de nossas imagens, descartando *pixels* não tão relevantes para a aplicação. Após a extração de características, aplicamos as técnicas de clusterização *k-means* e *fuzzy k-means* com o objetivo de categorizar os oleodutos em dois grupos distintos: aqueles que apresentam anomalias e aqueles que não apresentam.

4.2.1 Extração de Características

Inicialmente, realizamos o *feature extraction* utilizando a arquitetura VGG16 (Figura 14), uma rede neural convolucional amplamente reconhecida no campo de reconhecimento de imagem (66). No entanto, adaptamos a VGG16 para atender às necessidades específicas deste projeto.

Removemos as camadas finais de classificação da VGG16, uma vez que nosso foco está na extração de características e não na classificação. Essa abordagem nos permite obter representações ricas e discriminativas das imagens de oleodutos, fundamentais para a etapa de clusterização.

Figura 14: Arquitetura da rede VGG16.



Fonte: Researchgate.net

O desenvolvimento da extração de características também foi realizado em *Python*, utilizando a computação paralela com 8 *threads*. Utilizamos a implementação da rede VGG16 presente na biblioteca *Keras*.

Juntamente com a VGG16, utilizamos o conceito de *transfer learning*, reaproveitando treinamentos anteriores da rede e reduzindo o custo computacional, utilizando pesos pré-treinados a partir da base de dados *Imagenet*, projeto fundamental no avanço da pesquisa em visão computacional e aprendizado profundo (67).

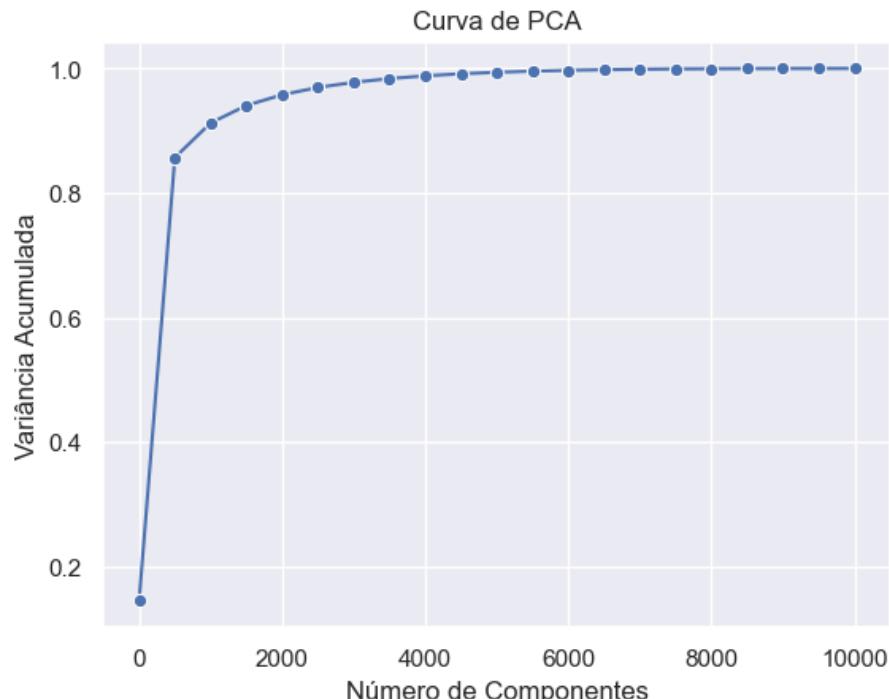
4.2.2 Principal Component Analysis (PCA)

Após extraímos as características das imagens, realizamos uma etapa exploratória dos dados através da técnica de Análise de Componentes Principais (PCA). Nessa etapa, executamos o algoritmo PCA em todo o conjunto de dados predito pelo *Detectron 2*, a fim de examinar a variância dos dados em relação ao número de componentes.

A análise de PCA mostrou-se altamente vantajosa ao reduzir o número de componentes dos dados de entrada, preservando informações relevantes em nossa base de dados. Ao avaliar a variância explicada por cada componente principal, identificamos quantos componentes eram necessários para capturarmos a maior parte da variabilidade dos dados. Isso nos permitiu reduzir a dimensionalidade dos dados, simplificando o modelo e acelerando o processo de treinamento sem comprometer significativamente o desempenho.

Por meio da análise do algoritmo PCA, utilizando uma configuração de 70% dos dados para treinamento e 30% dos dados para teste, foi possível identificar que cerca de 95% da variância dos dados era representada por apenas 2000 componentes (conforme evidenciado no gráfico 15). Essa descoberta nos permitiu reduzir significativamente o número de componentes necessários para representar os dados, passando de 10115 para 2000, o que representa uma redução de aproximadamente 80,2%. Essa redução considerável no número de componentes preserva a essência das informações contidas nos dados, ao mesmo tempo em que simplifica a representação e o processamento desses dados.

Figura 15: Variância acumulada x Número de componentes



4.2.3 Clusterização de anomalias: *k-means* e *fuzzy k-means*

Técnicas de aprendizado não-supervisionado foram escolhidas, visto que não possuíamos anotações (*ground truth*) das anomalias nos oleodutos e também devido a inviabilidade da construção de tal base. Nesse contexto, os algoritmos *k-means* e *fuzzy k-means* foram escolhidos como abordagens adequadas. Esses algoritmos permitem a clusterização das imagens de forma automatizada, identificando padrões e agrupando os oleodutos com base em suas características, mesmo na ausência de rótulos prévios.

Ambos os algoritmos, *k-means* e *fuzzy k-means*, são aplicados às características extraídas das máscaras binárias dos oleodutos. Essa abordagem nos permite uma análise mais detalhada e a identificação de padrões entre os diferentes grupos de oleodutos, contribuindo para a detecção de anomalias com maior precisão.

O desenvolvimento da clusterização foi realizado através da biblioteca *scikit-learn* para o algoritmo *k-means* e através da biblioteca *fcmeans* para o algoritmo *fuzzy k-means*. Como resultado final da arquitetura proposta, obtemos a categorização dos oleodutos em dois grupos distintos: aqueles que apresentam anomalias e aqueles que não apresentam. É importante ressaltar que uma análise quantitativa foi realizada para identificarmos o *cluster* que apresenta anomalia e o *cluster* que não apresenta (vide seção 5).

5 Resultados

Neste capítulo, são apresentados diversos experimentos que abordam a metodologia proposta. Optamos por fixar o número de épocas em todos os experimentos da etapa de segmentação, a fim de garantir uma comparação justa entre eles.

Um dos objetivos é obter resultados similares, para permitir uma comparação com o trabalho previamente mencionado (62). A comparação com este trabalho é adequada, uma vez que, embora os algoritmos utilizados sejam diferentes, ambos têm como base fundamental as CNNs. Além disso, é importante destacar que empregamos a mesma métrica de avaliação, ou seja, a "Precisão Média"(*Average Precision*).

As métricas de AP do COCO *framework* mensuram o desempenho do nosso modelo de segmentação de oleodutos subaquáticos. O COCO *framework* define três variantes do AP: AP, AP@.50 e AP@.75.

O AP representa o *Average Precision* médio calculado em diferentes valores de *threshold* de IoU que definem a sobreposição necessária para considerar uma detecção como correta. O AP@.50 é calculado considerando apenas as detecções com um *threshold* de IoU igual a 0.50, enquanto o AP@.75 considera um *threshold* de IoU igual a 0.75. Essas variantes permitem avaliar o desempenho do modelo em diferentes níveis de exigência de sobreposição entre as regiões preditas e as regiões verdadeiras.

Ao analisar as métricas de AP, AP@.50 e AP@.75, podemos compreender como o modelo se comporta em termos de precisão em diferentes cenários de sobreposição. O AP nos fornece uma visão geral do desempenho médio, enquanto o AP@.50 e AP@.75 nos fornecem informações mais específicas sobre a capacidade de detecção do modelo em níveis mais baixos e mais altos de sobreposição, respectivamente. Essas métricas são importantes para avaliar o quanto bem o modelo consegue detectar com precisão os oleodutos subaquáticos em diferentes condições e são amplamente utilizadas na comunidade de visão computacional para avaliar a qualidade dos resultados obtidos pelos modelos de detecção (4).

Com relação à clusterização, como não possuímos uma referência (*ground truth*) para essa etapa, avaliamos o desempenho da clusterização por meio de amostragem e inspeção visual. Vale ressaltar que o conceito de anomalia em oleodutos é subjetivo. Neste trabalho, consideramos que um oleoduto apresenta anomalia quando está submerso em areia e possui baixa visibilidade ou está envolto por plantas que possam danificar a estrutura.

Os experimentos foram feitos em uma máquina contendo placa de vídeo *Nvidia GeForce RTX 2060* com 6GB de memória, 32GB de memória RAM, processador *Ryzen 7 3600x* e *Windows 11*. O código fonte escrito (sem a base de dados utilizada e considerando a versão 3.10 da linguagem de programação *Python*), está disponível publicamente no *github* (68).

5.1 Experimento 1

Nessa abordagem, treinamos a rede *Detectron 2* com três conjuntos de hiperparâmetros diferentes. Nosso objetivo é realizar um *tuning* dos hiperparâmetros da rede e encontrar a melhor combinação. Vale ressaltar que, dado o custo computacional de treinamento da rede, seria inviável aplicarmos técnicas de busca exaustiva de hiperparâmetros, como *grid search*.

Dito isso, realizamos o treinamento para os conjuntos de hiperparâmetros abaixo:

Tabela 1: Configuração 1 dos hiperparâmetros

Parâmetros/Configurações	Valores
Número de iterações	2000
Iterações warmup	1000
Learning rate	0.001
Gamma	0.2
Solver steps	[800, 1200, 1600]
Eval Period	200
Batch size per image	64
Scheduler do learning rate	WarmupMultiStepLR
Momentum	0.95
Gradients Clip	norm
Norm Type	2.0
Clip value	5.0

Tabela 2: Configuração 2 dos hiperparâmetros

Parâmetros/Configurações	Valores
Número de iterações	2000
Iterações warmup	250
Learning rate	0.001
Gamma	0.5
Solver steps	[500, 1000]
Eval Period	200
Batch size per image	256
Scheduler do learning rate	WarmupCosineLR
Momentum	0.85
Gradients Clip	norm
Norm Type	2.0
Clip value	5.0

Tabela 3: Configuração 3 dos hiperparâmetros

Parâmetros/Configurações	Valores
Número de iterações	2000
Iterações warmup	500
Learning rate	0.001
Gamma	0.1
Solver steps	[1000, 1500]
Eval Period	200
Batch size per image	128
Scheduler do learning rate	WarmupCosineLR
Momentum	0.9
Gradients Clip	norm
Norm Type	2.0
Clip value	5.0

Ao analisar o gráfico 16, é evidente que a configuração 3 se destaca com uma maior precisão média na detecção dos oleodutos. Essa tendência é consistente tanto nos resultados apresentados no gráfico 17 quanto no gráfico 18, indicando que, mesmo considerando diferentes *thresholds* de avaliação, a configuração 3 continua demonstrando um maior AP. Essa configuração também apresenta o menor valor de *total loss* da rede, como visto na figura 19.

Figura 16: Precisão média - COCO x Épocas

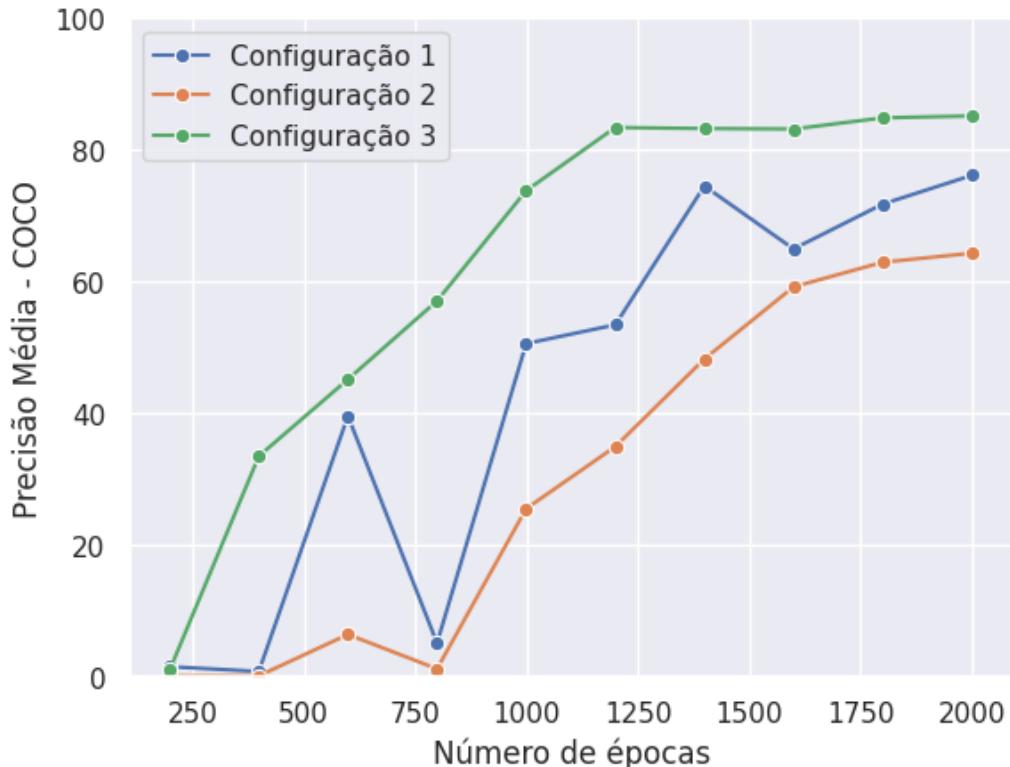


Figura 17: Precisão média - COCO (IoU=0.50) x Épocas

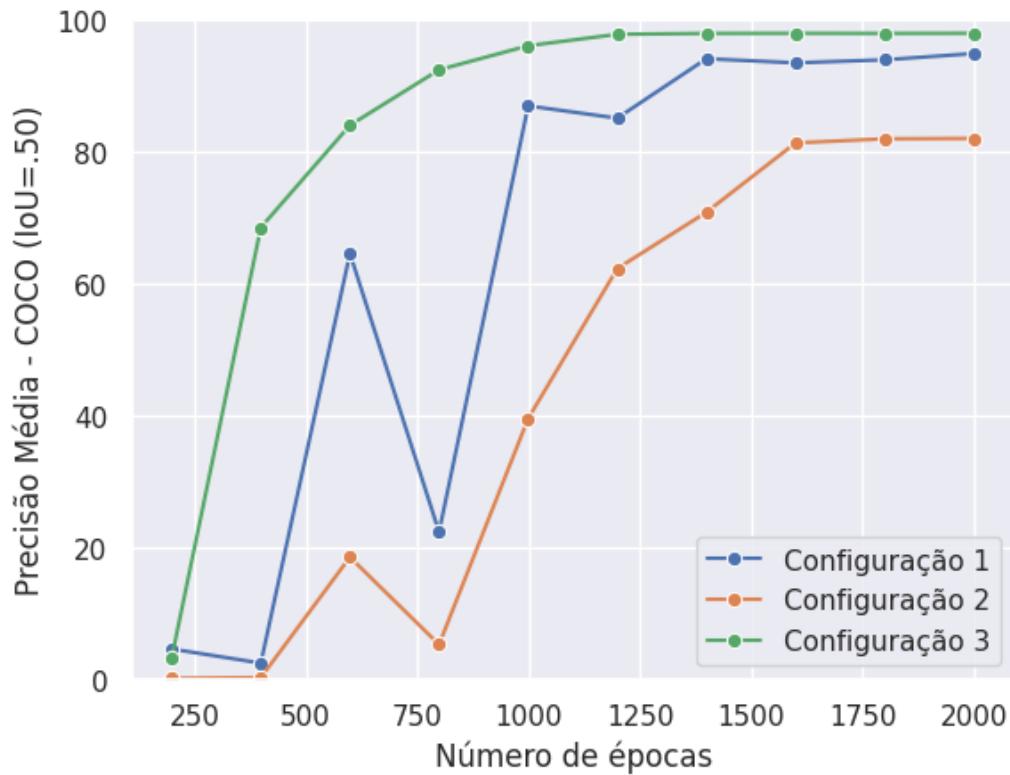


Figura 18: Precisão média - COCO (IoU=0.75) x Épocas

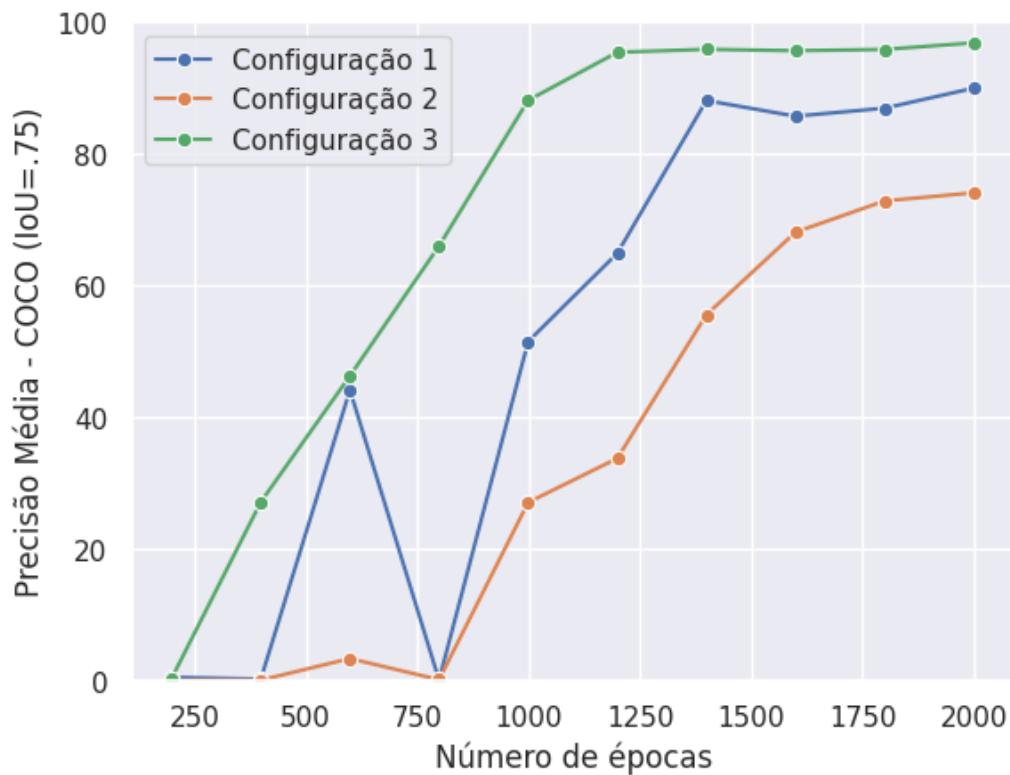
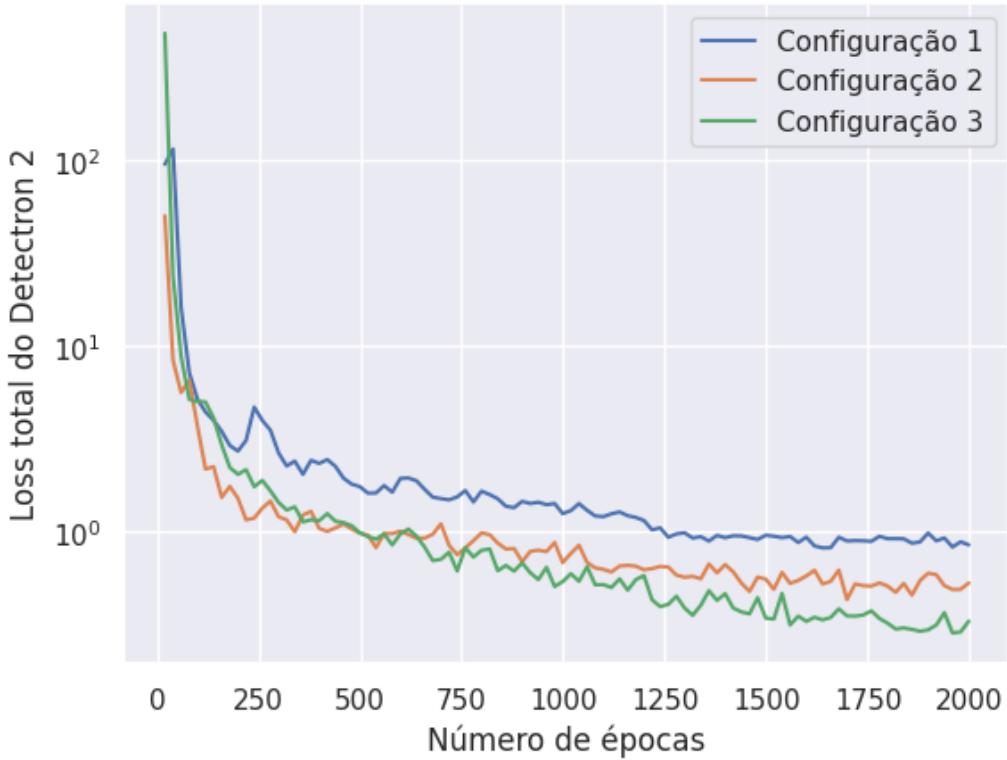


Figura 19: *Total loss* da rede (em escala logarítmica) x Épocas



5.2 Experimento 2

Nessa análise, realizamos o treinamento do Detectron 2 variando apenas o tamanho dos conjuntos de treino e teste. Testamos duas configurações: 70% dos dados de entrada para treino e 30% para teste; 80% dos dados de entrada para treino e 20% para teste. Ademais, utilizamos a configuração 3 de hiperparâmetros da subseção 5.1, visto ter demonstrado maior precisão.

Ao examinar o gráfico 20, torna-se notório que a abordagem de dividir o conjunto de dados em uma proporção de 70% para treinamento e 30% para teste se destaca pela sua precisão média superior na detecção de oleodutos. Essa tendência demonstra-se constante tanto nos resultados apresentados no gráfico 21 quanto no gráfico 22, o que indica que, mesmo ao considerar diferentes limiares de avaliação, essa divisão continua a apresentar um AP mais elevado. Vale ressaltar que essa configuração também registra o valor mais baixo de *total loss* da rede, conforme ilustrado na figura 23.

Figura 20: Precisão média - COCO x Épocas

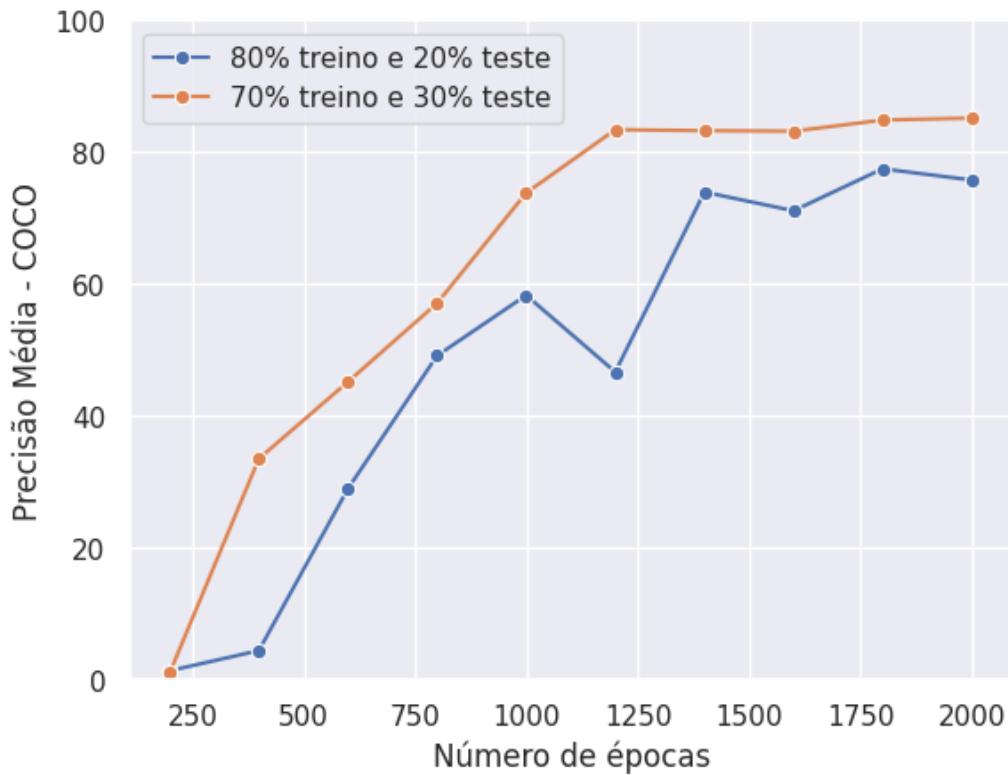


Figura 21: Precisão média - COCO (IoU=0.50) x Épocas

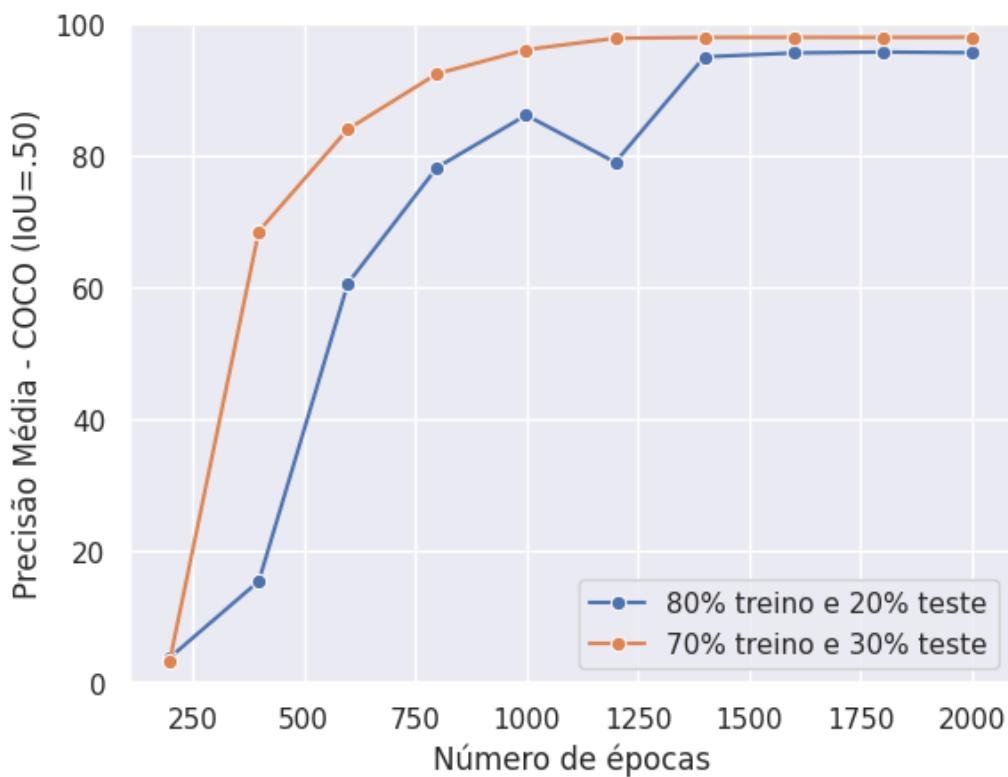


Figura 22: Precisão média - COCO (IoU=0.75) x Épocas

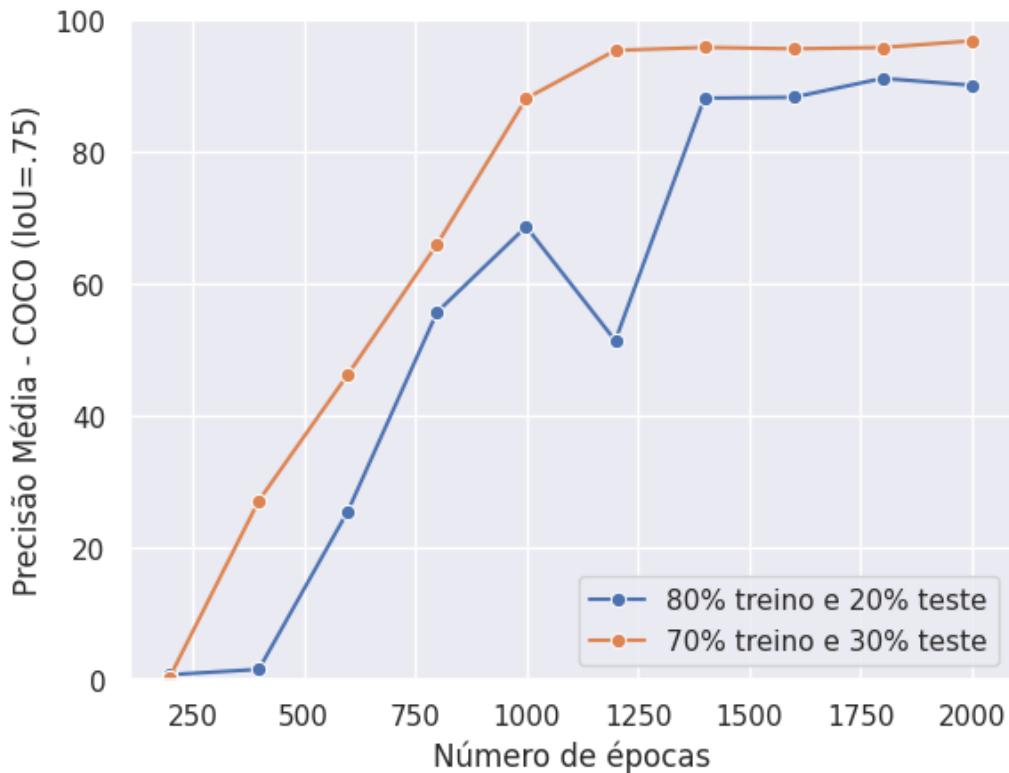
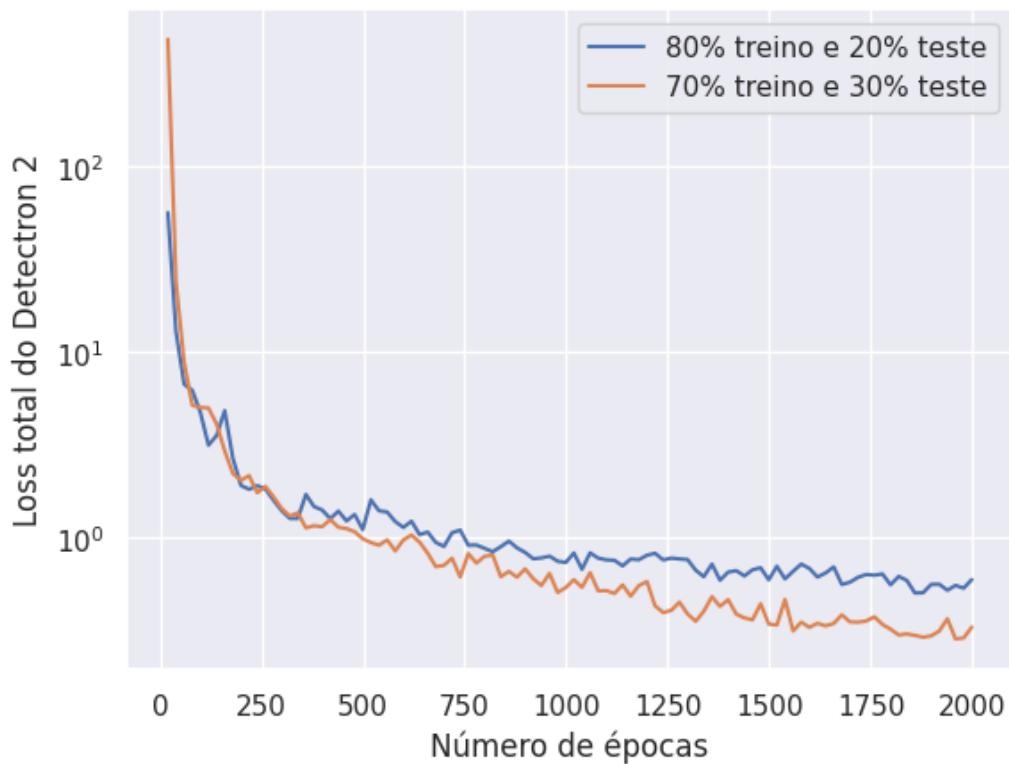


Figura 23: Total loss da rede (em escala logarítmica) x Épocas



5.3 Experimento 3

Nesse experimento, realizamos o treinamento do *Detectron 2*, variando apenas os pesos pré-treinados da rede. Também utilizamos a configuração 3 de hiperparâmetros da subseção 5.1, visto ter demonstrado maior precisão.

Testamos três configurações: sem o uso de *transfer learning*; utilizando *transfer learning* com pesos pré-treinados a partir da rede *Resnet50* e da *Resnet101*. Analisando os gráficos de precisão média e *total loss* do treinamento da rede, é evidente que o uso da técnica de *transfer learning* apresenta uma maior precisão média na detecção dos oleodutos.

Os resultados obtidos demonstram que a *ResNet50* alcançou um AP de 94.1, enquanto a *ResNet101* obteve um AP de 94.4 (vide figura 25). Além disso, ambas as redes apresentaram resultados semelhantes para as métricas AP@.50 e AP@.75, como observado nas figuras 26, 24, assim como o *total loss* da rede (figura 27). Embora os resultados sejam muito próximos, a *ResNet50* possui uma menor demanda computacional em comparação com a *ResNet101*, o que a torna uma opção mais viável em termos de eficiência.

Figura 24: Precisão média - COCO x Épocas

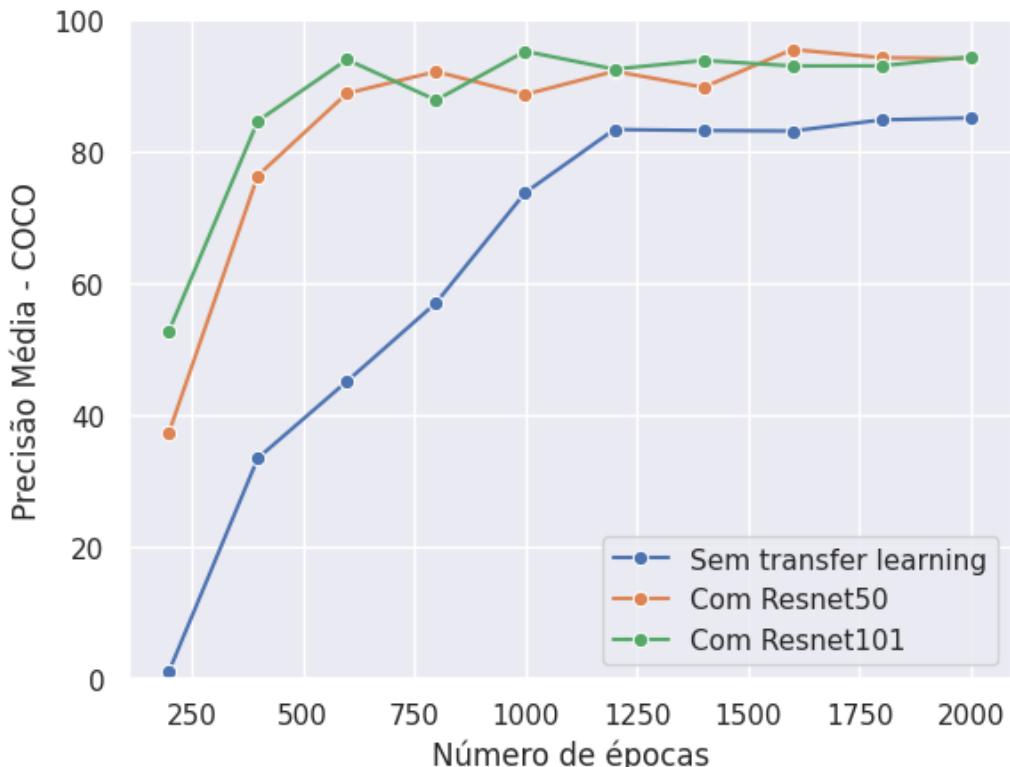


Figura 25: Precisão média - COCO (IoU=0.50) x Épocas

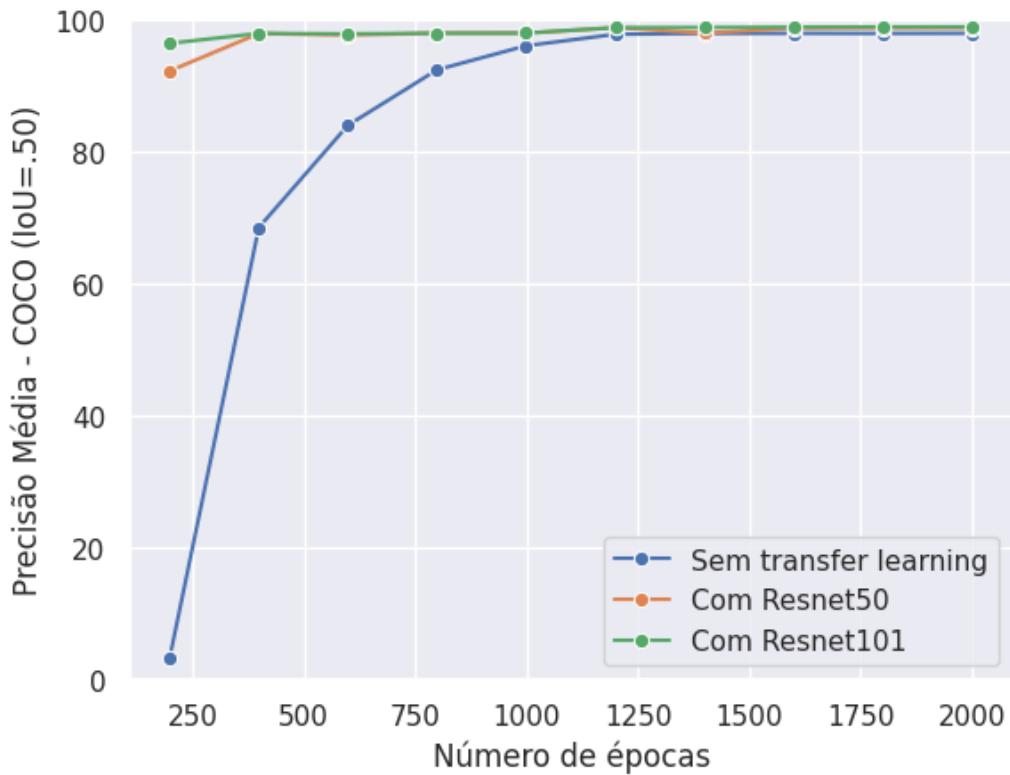


Figura 26: Precisão média - COCO (IoU=0.75) x Épocas

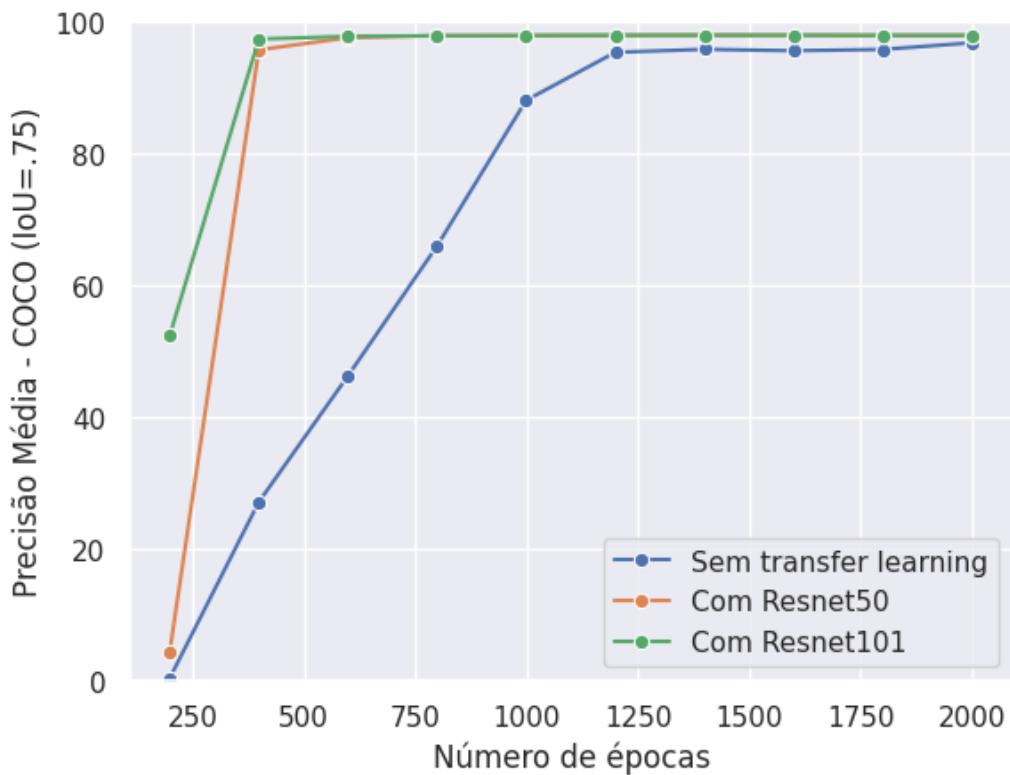
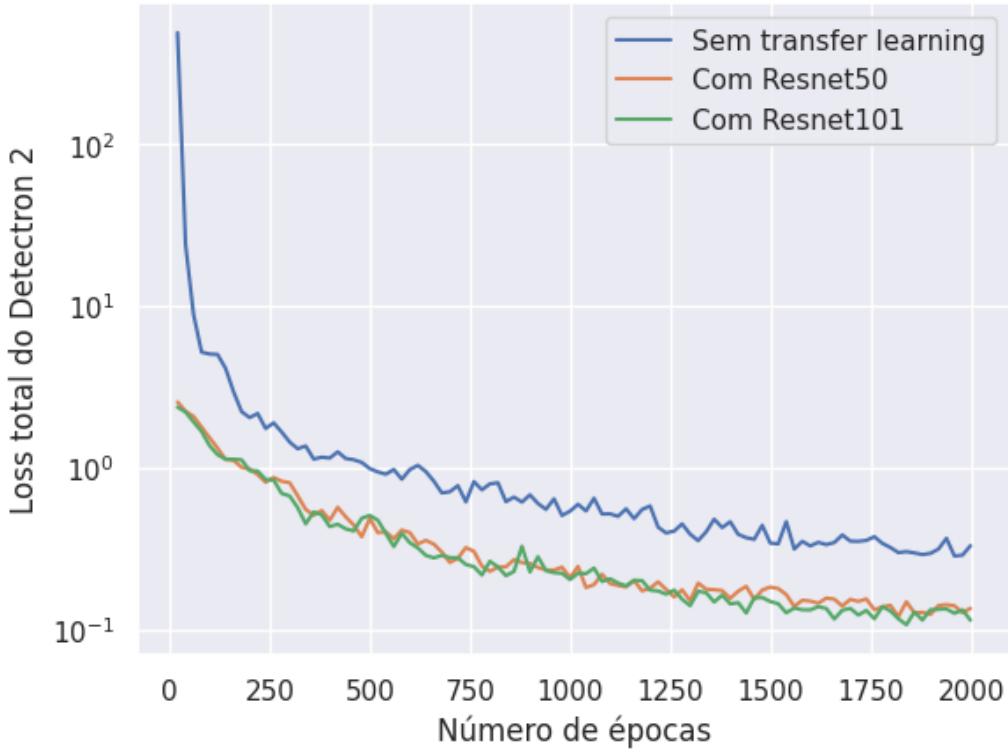


Figura 27: *Total loss* da rede (em escala logarítmica) x Épocas



5.4 Experimento 4

Ao longo desta análise, treinamos os algoritmos de clusterização, *k-means* e *fuzzy k-means*, com o objetivo de comparar os resultados dessas diferentes técnicas. Treinamos os algoritmos com três valores distintos para o parâmetro *random state*. Os parâmetros utilizados no treinamento podem ser observados nas tabelas 4 e 5. É importante destacar que o vetor de características, entrada da nossa etapa de clusterização, passou por uma redução de componentes, indo de 10155 para 2000, como evidenciado em 4.2.2.

Tabela 4: Parâmetros usados no algoritmo *k-means*

Parâmetro	Configuração
Número de <i>clusters</i>	2
Inicialização	<i>k-means++</i>
Máximo de iterações	300
Tolerância	1e-4

Tabela 5: Parâmetros usados no algoritmo *fuzzy k-means*

Parâmetro	Configuração
Número de <i>clusters</i>	2
Grau de incerteza	1.1
Máximo de iterações	100
Tolerância	1e-09

Os resultados da clusterização podem ser observados nas figuras 28 e 29. Desenvolvemos nossa análise quantitativa a partir de 500 imagens de cada *cluster*, selecionando-as aleatoriamente, e realizando inspeção visual nas imagens para identificar anomalia ou normalidade. A partir da clusterização, obtivemos as seguintes distribuições de amostra por *cluster*:

Figura 28: *k-means* - Distribuição de amostras por *clusters*

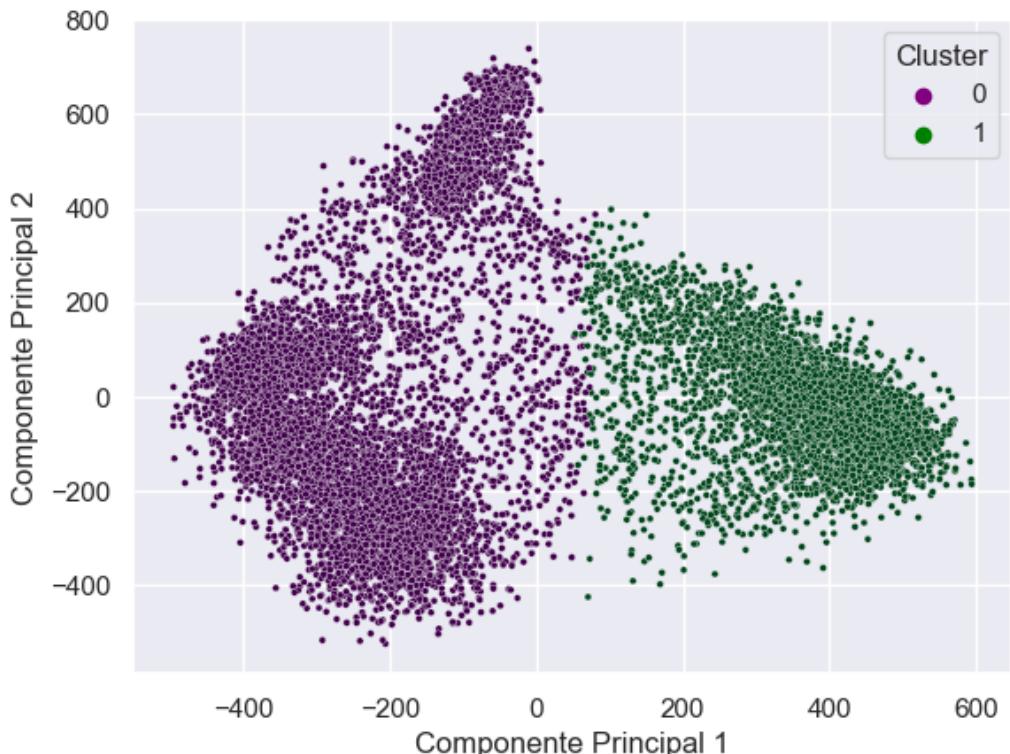
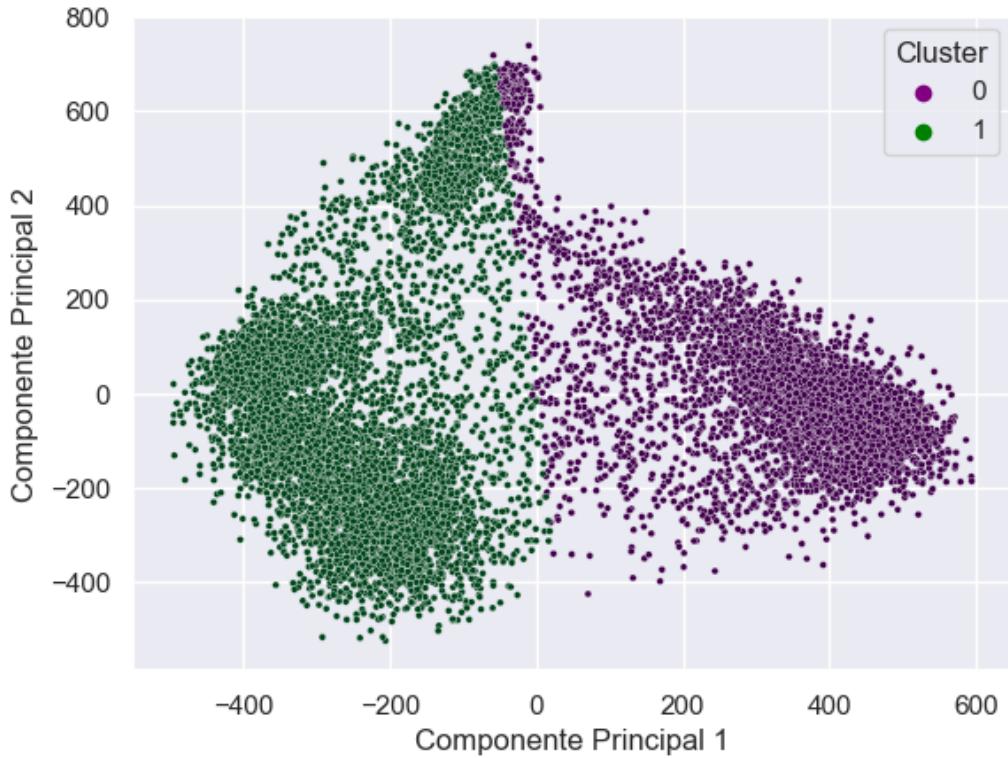


Figura 29: *fuzzy k-means* - Distribuição de amostras por *clusters*



Por meio de uma análise visual, podemos observar que os agrupamentos obtidos utilizando o algoritmo *k-means* e o algoritmo *fuzzy k-means* apresentam uma semelhança significativa, com diferença na inversão das *labels*. No entanto, nossa análise quantitativa será discutida na subseção 5.5, fornecendo uma avaliação mais detalhada dos resultados.

5.5 Considerações finais

Durante o treinamento da rede de segmentação, realizamos testes com diferentes configurações de hiperparâmetros e divisões do conjunto de dados. Essas combinações variadas tiveram um impacto significativo na segmentação do oleoduto. Nas configurações 2 e 3 do Experimento 3 (com 70% dos dados para treinamento e utilizando as redes *ResNet50* e *ResNet101*), alcançamos valores ótimos de AP, no conjunto de teste, que se equiparam ao trabalho de referência (62) (veja a tabela 6). No entanto, na melhor configuração sem uso de *transfer learning* (Experimento 1, configuração 3), obtivemos precisão inferior em comparação ao trabalho de referência.

Analisando as Figuras 24, 25 e 26, observamos que os gráficos de AP não tendem a melhorar aumentando o número de épocas, principalmente porque eles estão próximos ou já atingiram certa estabilidade. Portanto, otimizar o desempenho da configuração que não utiliza *transfer learning* por meio do aumento do número de iterações não seria uma solução viável. No entanto, devido ao custo computacional, testamos apenas três configurações de hiperparâmetros para a rede *Detectron 2*. Em

trabalhos futuros, seria interessante realizar testes com outras configurações de hiperparâmetros com o objetivo de aprimorar a precisão da configuração que não utiliza *transfer learning*.

A comparação dos resultados entre os diferentes experimentos de segmentação pode ser observada na tabela 6. É importante ressaltar que o termo "E" na tabela se refere ao experimento e o termo "C" se refere à configuração (na ordem apresentada nas seções anteriores).

Tabela 6: Comparação entre as diferentes configurações apresentadas.

Experimento	AP no treinamento	Total loss no treinamento	AP no teste
E1_C1	76.1%	0.84	76.35%
E1_C2	64.26%	0.52	62.62%
E1_C3 / E2_C2 / E3_C1	85.11%	0.32	84.87%
E2_C1	75.70%	0.59	73.92%
E3_C2	94.12%	0.13	93.15%
E3_C3	94.4%	0.11	93.80%

Na etapa de clusterização, utilizamos como métrica avaliativa uma medida quantitativa (veja as tabelas 7 e 8), obtida por meio de uma inspeção visual em uma amostragem dos dados clusterizados, para analisar os resultados obtidos pelos algoritmos *k-means* e *fuzzy k-means*. Em ambas as técnicas, um dos resultados se destacou nos *clusters* de saída, indicando que conseguimos fazer uma distinção razoavelmente boa entre anomalias e normalidade. Obtivemos uma maior facilidade na clusterização de anomalias com o *fuzzy k-means*, alcançando uma taxa de acerto de 95%, e uma maior facilidade na clusterização de oleodutos em condições normais com o *k-means*, alcançando uma taxa de acerto de 71,1%.

Tabela 7: Análise quantitativa do algoritmo *k-means*

	Cluster 0	Cluster 1
Anomalia	71,1%	36,7%
Normalidade	28,9%	63,3%

Tabela 8: Análise quantitativa do algoritmo *fuzzy k-means*

	Cluster 0	Cluster 1
Anomalia	34,4%	95,1%
Normalidade	65,5%	4,9%

6 Conclusão

Este trabalho abordou uma tarefa de segmentação e clusterização de anomalias em oleodutos subaquáticos, com o objetivo de analisar e identificar possíveis falhas, em imagens de oleodutos subaquáticos capturadas por ROVs, utilizando técnicas de visão computacional. A escolha pela análise de imagens de oleodutos subaquáticos deu-se pela relevância em se manter a integridade e segurança dessas estruturas críticas. Para detecção de falhas usamos uma abordagem baseada em técnicas de visão computacional, visando a segmentação dos oleodutos e clusterização de eventuais falhas presentes. Por meio dessas técnicas, é possível identificar e agrupar falhas com eficiência, contribuindo para a prevenção de possíveis danos ambientais e prejuízos financeiros.

Neste trabalho, diferentes configurações foram treinadas e testadas em um mesmo conjunto de dados personalizado de imagens subaquáticas, a fim de investigar a robustez do modelo *Detectron 2* na detecção de oleodutos em cenários subaquáticos desafiadores. Os resultados de detecção da rede *Detectron 2* passaram por uma etapa de extração de características e redução de componentes (PCA) e foram apresentados aos algoritmos de clusterização *k-means* e *fuzzy k-means* com o objetivo de agruparmos os oleodutos em situações de normalidade ou anomalia.

A partir dos resultados expostos na etapa de segmentação dos oleodutos, é possível destacar que neste trabalho obtivemos valor de AP ligeiramente inferior ao realizado por (62). Nossa estudo demonstrou a efetividade da rede *Detectron 2* na segmentação de oleodutos subaquáticos, alcançando AP de 93,8% para o conjunto de testes. Em comparação com outros métodos semelhantes, nosso método apresenta resultados promissores no tratamento do problema de detecção de dutos subaquáticos. Para obter métricas de desempenho ainda melhores, é possível utilizar métodos de aprimoramento de imagens para melhorar a qualidade do conjunto de dados de imagens subaquáticas.

Na etapa de clusterização de anomalias, realizamos a comparação dos algoritmos *k-means* e *fuzzy k-means* e observamos um agrupamento significativamente semelhante, indicando que ambas as técnicas identificaram comportamentos semelhantes. Como não foi possível criar um "ground truth" das anomalias, nossa análise foi realizada de maneira quantitativa, levando em consideração a quantidade de amostras normais e anômalas em cada cluster. Obtivemos uma maior facilidade na clusterização de anomalias com o *fuzzy k-means*, alcançando uma taxa de acerto de 95%, e uma maior facilidade na clusterização de oleodutos em condições normais com o *k-means*, alcançando uma taxa de acerto de 71,1%.

Por fim, os resultados deste estudo podem ser aplicados pelos operadores de ROVs, técnicos e pesquisadores como base para melhoria da inspeção de oleodutos subaquáticos e para prevenção de possíveis problemas.

REFERÊNCIAS

- 1 GAUTO, M. A. et al. **Petróleo e gás: princípios de exploração, produção e refino.** [S.l.]: Bookman Editora, 2016.
- 2 ORTIZ, A.; SIMÓ, M.; OLIVER, G. A vision system for an underwater cable tracker. **Machine vision and applications**, Springer, v. 13, p. 129–140, 2002.
- 3 WU, Y. et al. **Detectron2**. 2019. <https://github.com/facebookresearch/detectron2>.
- 4 CONTEXT), C. D. C. O. in. **Detection Evaluation**. 2020. <https://cocodataset.org/#detection-eval>.
- 5 SOCIETY, R. O. V. C. of the M. T. **Rov market - commercial applications**. 2023. <https://rov.org/market/>.
- 6 MAPFRE Global Risks. **Oleodutos e Gasodutos, as veias da economia**. Accessed on March 11, 2023. Disponível em: <https://www.mapfreglobalisks.com/pt-br/gerencia-riscos-seguros/estudos/oleodutos-e-gasodutos-as-veias-da-economia/>.
- 7 BRAITHWAITE, J. Operational pipeline inspection. **The Assessment and Control of Major Hazards**, Pergamon, n. 322, p. 45, 1985.
- 8 TEAM, T. O. P. **Gulf oil spill**. 2010. Disponível em: <http://ocean.si.edu/gulf-oil-spill>.
- 9 REID, A. **Rov market prospects**. 2013. Online. Disponível em: <http://www.subseauk.com/documents/presentations/ssuk%20-\%20rov\%20event%20-\%20sep\%202013\%20\%5Bweb\%5D.pdf>.
- 10 RODRIGUES, C. E. Relatório Final PIBIC. p. 3–5, 2019.
- 11 POGGIO, T.; TORRE, V.; KOCH, C. Computational vision and regularization theory. In: FISCHLER, M. A.; FIRSCHEIN, O. (Ed.). **Readings in Computer Vision**. San Francisco (CA): Morgan Kaufmann, 1987. p. 638–643. ISBN 978-0-08-051581-6. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780080515816500611>.
- 12 TAVARES, J. M. R.; JORGE, R. M. N. Database system for clinical and computer assisted diagnosis of dermoscopy images. In: _____. **Topics in Medical Image Processing and Computational Vision**. 1. ed. [S.l.]: Springer Dordrecht. (2212-9413), p. 261–274.
- 13 _____. Tracking red blood cells in microchannels: A comparative study between an automatic and a manual method. In: _____. **Topics in Medical Image Processing and Computational Vision**. 1. ed. [S.l.]: Springer Dordrecht. (2212-9413), p. 165–174.
- 14 NEGRI, J. D. et al. Scaled autonomous car. **Journal of production and automation**, v. 2, p. 32–40, 2019.
- 15 B., C.; P., S. An overview on image processing techniques. **International Journal of Innovative Research in Computer and Communication Engineering**, v. 2, n. 11, Nov 2014. Disponível em: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c3853d6a22f539dfd23ff7a1d97a5241a07eb22d>.

- 16 GUPTA, N. et al. Artificial neural network. **Network and Complex Systems**, v. 3, n. 1, p. 24–28, 2013.
- 17 FIORIN, D. V. et al. Aplicações de redes neurais e previsões de disponibilidade de recursos energéticos solares. **Revista Brasileira de Ensino de Física**, Sociedade Brasileira de Física, v. 33, n. 1, p. 01–20, Jan 2011. ISSN 1806-1117. Disponível em: <https://doi.org/10.1590/S1806-11172011000100009>.
- 18 AGGARWAL, C. C. Single computational layer: The perceptron. In: _____. **Neural Networks and Deep Learning**. 1. ed. Springer Cham. p. 5–17. Disponível em: <https://link.springer.com/book/10.1007/978-3-319-94463-0>.
- 19 FERNÁNDEZ, A. S.; DELGADO-MATA, C.; VELAZQUEZ, R. Training a single-layer perceptron for an approximate edge detection on a digital image. In: **2011 International Conference on Technologies and Applications of Artificial Intelligence**. [S.l.: s.n.], 2011. p. 189–193.
- 20 AGGARWAL, C. C. Chapter 1. an introduction to neural networks. In: _____. **Neural Networks and Deep Learning**. 1. ed. Springer Cham. p. 10. Disponível em: <https://link.springer.com/book/10.1007/978-3-319-94463-0>.
- 21 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.
- 22 SHARMA, S.; SHARMA, S.; ATHAIYA, A. Activation functions in neural networks. **International Journal of Engineering Applied Sciences and Technology**, v. 4, n. 12, p. 310–316, Apr 2020. Disponível em: <https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf>.
- 23 ZHANG, Z.; SABUNCU, M. Generalized cross entropy loss for training deep neural networks with noisy labels. In: BENGIO, S. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2018. v. 31. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2018/file/f2925f97bc13ad2852a7a551802feea0-Paper.pdf.
- 24 MOHAMMAD, R.; THABTAH, F.; MCCLUSKEY, T. Predicting phishing websites based on self-structuring neural network. **Neural Computing and Applications**, v. 25, n. 2, p. 443–458, 2014. ISSN 0941-0643.
- 25 RUDER, S. An overview of gradient descent optimization algorithms. **arXiv preprint arXiv:1609.04747**, 2016.
- 26 LECUN, Y. et al. Neural networks: Tricks of the trade. **Springer Lecture Notes in Computer Sciences**, v. 1524, n. 5-50, p. 6, 1998.
- 27 LI, M. et al. Efficient mini-batch training for stochastic optimization. In: **Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2014. p. 661–670.
- 28 O'SHEA, K.; NASH, R. **An Introduction to Convolutional Neural Networks**. 2015.
- 29 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.
- 30 LINDEBERG, T. Scale invariant feature transform. 2012.

- 31 DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: IEEE. **2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)**. [S.I.], 2005. v. 1, p. 886–893.
- 32 AHONEN, T.; HADID, A.; PIETIKAINEN, M. Face description with local binary patterns: Application to face recognition. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 28, n. 12, p. 2037–2041, 2006.
- 33 HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. **J. Physiol.**, v. 160, n. 1, p. 106–154, Jan. 1962.
- 34 KIM, P. Convolutional neural network. In: _____. **MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence**. Berkeley, CA: Apress, 2017. p. 121–147. ISBN 978-1-4842-2845-6. Disponível em: https://doi.org/10.1007/978-1-4842-2845-6_6.
- 35 ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: IEEE. **2017 international conference on engineering and technology (ICET)**. [S.I.], 2017. p. 1–6.
- 36 HABIB, G.; QURESHI, S. Optimization and acceleration of convolutional neural networks: A survey. **Journal of King Saud University - Computer and Information Sciences**, v. 34, n. 7, p. 4244–4268, 2022. ISSN 1319-1578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157820304845>.
- 37 POOJARY, R.; PAI, A. Comparative study of model optimization techniques in fine-tuned cnn models. In: IEEE. **2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)**. [S.I.], 2019. p. 1–4.
- 38 IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: BACH, F.; BLEI, D. (Ed.). **Proceedings of the 32nd International Conference on Machine Learning**. Lille, France: PMLR, 2015. (Proceedings of Machine Learning Research, v. 37), p. 448–456. Disponível em: <https://proceedings.mlr.press/v37/ioffe15.html>.
- 39 GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.I.: s.n.], 2014.
- 40 ROWLEY, H. A.; BALUJA, S.; KANADE, T. Neural network-based face detection. **IEEE Transactions on pattern analysis and machine intelligence**, IEEE, v. 20, n. 1, p. 23–38, 1998.
- 41 VAILLANT, R.; MONROCQ, C.; CUN, Y. L. Original approach for the localisation of objects in images. **IEE Proceedings-Vision, Image and Signal Processing**, IET, v. 141, n. 4, p. 245–250, 1994.
- 42 SERMANET, P. et al. Pedestrian detection with unsupervised multi-stage feature learning. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.I.: s.n.], 2013. p. 3626–3633.
- 43 GU, C. et al. Recognition using regions. In: IEEE. **2009 IEEE Conference on computer vision and pattern recognition**. [S.I.], 2009. p. 1030–1037.

- 44 UIJLINGS, J. R. et al. Selective search for object recognition. **International journal of computer vision**, Springer, v. 104, p. 154–171, 2013.
- 45 WANG, X. et al. Regionlets for generic object detection. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2013. p. 17–24.
- 46 BENEDIKTSSON, J.; PESARESI, M.; AMASON, K. Classification and feature extraction for remote sensing images from urban areas based on morphological transformations. **IEEE Transactions on Geoscience and Remote Sensing**, v. 41, n. 9, p. 1940–1949, 2003.
- 47 THECKEDATH, D.; SEDAMKAR, R. R. Detecting affect states using vgg16, resnet50 and se-resnet50 networks. **SN COMPUT. SCI.**, v. 1, p. 79, 2020. Disponível em: <https://doi.org/10.1007/s42979-020-0114-9>.
- 48 WANG, D. Unsupervised learning: Foundations of neural computation. **AI Magazine**, v. 22, n. 2, p. 101, Jun. 2001. Disponível em: <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1565>.
- 49 GREENE, D.; CUNNINGHAM, P.; MAYER, R. Unsupervised learning and clustering. In: CORD, M.; CUNNINGHAM, P. (Ed.). **Machine Learning Techniques for Multimedia**. Berlin, Heidelberg: Springer, 2008.
- 50 DUNN, J. C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Taylor & Francis, 1973.
- 51 DEEPAK, P. Anomaly detection for data with spatial attributes. In: CELEBI, M. E.; AYDIN, K. (Ed.). **Unsupervised Learning Algorithms**. 1. ed. [S.l.]: Springer Cham, 2016. p. 1–32.
- 52 WONG, Y. L. K.-C.; ZHANG, Z. Unsupervised learning in genome informatics. In: CELEBI, M. E.; AYDIN, K. (Ed.). **Unsupervised Learning Algorithms**. 1. ed. [S.l.]: Springer Cham, 2016. p. 405–448.
- 53 GOVENDER, P.; SIVAKUMAR, V. Application of k-means and hierarchical clustering techniques for analysis of air pollution: A review (1980–2019). **Atmospheric Pollution Research**, v. 11, n. 1, p. 40–56, 2020. ISSN 1309-1042. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1309104219304556>.
- 54 NANDAPALA, E.; JAYASENA, K. The practical approach in customers segmentation by using the k-means algorithm. In: **2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)**. [S.l.: s.n.], 2020. p. 344–349.
- 55 ANAND, R.; VENI, S.; ARAVINTH, J. An application of image processing techniques for detection of diseases on brinjal leaves using k-means clustering method. In: **2016 International Conference on Recent Trends in Information Technology (ICRTIT)**. [S.l.: s.n.], 2016. p. 1–6.
- 56 CHEN, J. et al. A detection method based on sonar image for underwater pipeline tracker. In: **Second International Conference on Mechanic Automation and Control Engineering**. [S.l.: s.n.], 2011. p. 3766–3769.
- 57 VILLAR, S. et al. Evaluation of an efficient approach for target tracking from acoustic imagery for the perception system of an autonomous underwater vehicle. **International Journal of Advanced Robotic Systems**, v. 11, p. 1–13, August 2013.

- 58 PAVIN, A. M. The pipeline identification method basing on auv's echo-sounder data. In: **OCEANS 2006**. [S.l.: s.n.], 2006. p. 1–6.
- 59 STAMOULAKATOS, A. et al. Automatic annotation of subsea pipelines using deep learning. **Sensors**, MDPI, v. 20, n. 3, p. 674, 2020.
- 60 GUO, W. et al. Detection and classification of pipe defects based on pipe-extended feature pyramid network. **Automation in Construction**, v. 141, p. 104399, 2022. ISSN 0926-5805. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0926580522002722>.
- 61 ZHAO, X.; WANG, X.; DU, Z. Research on detection method for the leakage of underwater pipeline by yolov3. In: **IEEE. 2020 IEEE international conference on mechatronics and automation (ICMA)**. [S.l.], 2020. p. 637–642.
- 62 GAŠPAROVIĆ, B. et al. Deep learning approach for objects detection in underwater pipeline images. **Applied Artificial Intelligence**, Taylor Francis, v. 36, n. 1, p. 2146853, 2022. Disponível em: <https://doi.org/10.1080/08839514.2022.2146853>.
- 63 de Albuquerque, A. O. et al. Instance segmentation of center pivot irrigation systems using multi-temporal sentinel-1 sar images. **Remote Sensing Applications: Society and Environment**, v. 23, p. 100537, 2021. ISSN 2352-9385. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2352938521000732>.
- 64 HE, K. et al. Mask r-cnn. In: **2017 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2017. p. 2980–2988.
- 65 GIRSHICK, R. Fast r-cnn. In: **2015 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. p. 1440–1448.
- 66 SIMONYAN, K.; ZISSERMAN, A. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. 2015.
- 67 ImageNet. **ImageNet: A Large-Scale Hierarchical Image Database**. [2023]. <https://www.image-net.org/about.php>. Accessed: [19/06].
- 68 EMILIANO, C.; OLIVEIRA, M. **Segmentação e clusterização de anomalias em oleodutos subaquáticos**. 2023. https://github.com/caio-emiliano/underwater_pipelines_segmentation_classification. Acesso em: 19/05/2023.