Pontifícia Universidade Católica
do Rio de Janeiro

Pedro Gabriel Serodio Sales

Abordagem One-Shot Learning para Classificação de Imagens de Inspeções Submarinas

Relatório de Projeto Final

Relatório de Projeto Final, apresentado como requisito parcial para obtenção do grau de Bacharel pelo Programa de Graduação em Engenharia da Computação, do Departamento de Informática da PUC-Rio .

Orientador : Prof. Marco Aurélio Pacheco
Co-orientador: Dra. Manoela Rabello Kohler

Rio de Janeiro
Outubro de 2025

Pontifícia Universidade Católica
do Rio de Janeiro

Pedro Gabriel Serodio Sales

Abordagem One-Shot Learning para Classificação de Imagens de Inspeções Submarinas

Relatório de Projeto Final, apresentado como requisito parcial para obtenção do grau de Bacharel pelo Programa de Graduação em Engenharia da Computação da PUC-Rio . Aprovada pela Comissão Examinadora abaixo:

Prof. Marco Aurélio Pacheco
Departamento de Engenharia Elétrica da PUC-Rio

Dra. Manoela Rabello Kohler
Departamento de Engenharia Elétrica da PUC-Rio

Dr. Vitor Bento de Souza
PUC-Rio

Rio de Janeiro, 20 de Outubro de 2025

Pedro Gabriel Serodio Sales

Graduado em engenharia da computação pela Pontifícia Universidade Católica do Rio de Janeiro.

Ficha Catalográfica

Serodio Sales, Pedro Gabriel

Abordagem One-Shot Learning para Classificação de Imagens de Inspeções Submarinas / Pedro Gabriel Serodio Sales; orientador: Marco Aurélio Pacheco; co-orientador: Manoela Rabello Kohler. – 2025.

54 f: il. color. ; 30 cm

Relatório de Projeto Final (graduação) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2025.

Inclui bibliografia

1. Informática – Teses. 2. One-Shot Learning. 3. Redes Siamesas. 4. Inspeções Submarinas. 5. Redes Neurais Artificiais de Aprendizado Profundo. 6. Redes Convolucionais. I. Aurélio Pacheco, Marco. II. Rabello Kohler, Manoela. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Aos meus pais, amigos e família
pelo apoio e encorajamento.

Agradecimentos

Ao meu orientador Professor Marco Aurélio Pacheco pelo estímulo e parceria para a realização deste trabalho.

Aos professores Vitor Bento e Manoela Kohler pela paciência, confiança e por não desistir de mim, mesmo eu deixando a desejar em diversos momentos e que foram muito profissionais e acolhedores durante a realização desse trabalho.

Ao meu amigo Eduardo Pêgas, pelas incontáveis horas de atenção e reflexão para manter o foco necessário pro projeto.

A minha amiga Maria Júlia, que me trouxe esperança e leveza durante momentos difíceis. Me possibilitando uma clareza maior sobre o que fazer e como prosseguir.

A minha amiga Shinnie que mesmo em outro país, se fez muito presente em momentos de apoio.

Aos meus amigos Erik Tronkos e Vinicius Nascimento pelo estudo em conjunto, discussões, paciência e diversos ensinamentos para confecção do trabalho.

Aos professores que participaram da Comissão examinadora.

A todos os professores e funcionários do Departamento pelos ensinamentos e pela ajuda.

Aos meus pais e familiares que de uma forma ou de outra me estimularam ou me ajudaram.

Resumo

Serodio Sales, Pedro Gabriel; Aurélio Pacheco, Marco; Rabello Kohler, Manoela. **Abordagem One-Shot Learning para Classificação de Imagens de Inspeções Submarinas**. Rio de Janeiro, 2025. 54p. Relatório de Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

As inspeções submarinas são essenciais para a manutenção de infraestruturas offshore, mas enfrentam desafios como alto custo de coleta, baixa qualidade das imagens e escassez de dados rotulados. Devido à diversidade de objetos e estruturas, modelos tradicionais são limitados, tornando técnicas que aprendem com poucos exemplos relevantes e representativos. O presente trabalho propõe um classificador de imagens voltado para inspeções submarinas utilizando a abordagem de One-Shot Learning. Essa técnica permite que modelos de aprendizado de máquina reconheçam novas classes a partir de um número extremamente limitado de exemplos, superando a limitação de conjuntos de dados escassos, comuns em aplicações submarinas. Para isso, foi utilizada uma rede neural do tipo Siamesa, capaz de aprender uma função de similaridade entre pares de imagens. O modelo foi treinado em um conjunto de imagens de inspeções submarinas contendo diferentes categorias, como ROV, dutos, flanges, manipuladores, objetos e equipamentos, e avaliado em tarefas *N-way*, utilizando métricas como acurácia, precisão, *recall* e *F1-score*. Os resultados demonstraram que a abordagem é eficaz na identificação de classes inéditas com desempenho competitivo, evidenciando a aplicabilidade do *One-Shot Learning* em cenários de inspeção submarina com disponibilidade limitada de dados.

Palavras-chave

One-Shot Learning; Redes Siamesas; Inspeções Submarinas; Redes Neurais Artificiais de Aprendizado Profundo; Redes Convolucionais.

Abstract

Serodio Sales, Pedro Gabriel; Aurélio Pacheco, Marco (Advisor); Rabello Kohler, Manoela (Co-Advisor). **One-Shot Learning Approach for image Classification of Submarine Inspections**. Rio de Janeiro, 2025. 54p. Undergraduate Thesis – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Underwater inspections are essential for maintaining offshore infrastructure but face challenges such as high collection costs, low image quality, and limited labeled data. Due to the diversity of objects and structures, traditional models are limited, making techniques that learn from few examples particularly relevant and representative. This work presents an image classifier for underwater inspections based on the One-Shot Learning approach. This technique enables machine learning models to recognize new classes from a very limited number of examples, overcoming the common limitation of scarce datasets in underwater applications. A Siamese neural network was employed to learn a similarity function between image pairs. The model was trained on a dataset of underwater inspection images containing different categories, such as ROVs, pipelines, objects, junctions, and equipment, and evaluated in N-way tasks using metrics including accuracy, precision, recall, and F1-score. The results demonstrated that the approach effectively identifies unseen classes with competitive performance, highlighting the applicability of One-Shot Learning in underwater inspection scenarios with limited data availability. This study contributes to the development of intelligent tools that can assist in the maintenance and monitoring of submerged structures.

Keywords

One-Shot Learning; Siamese Networks; Underwater Inspection; Deep Learning; Convolutional Neural Networks.

Sumário

1 Introdução **14**

2 Situação Atual **18**
2.1 Apresentação 18
2.2 Problemática Proposta 18
2.3 Trabalhos Relacionados 18
2.4 Deep Learning 19
2.5 Arquitetura Convolucional 20
2.6 Tipos de Aprendizagem 22
2.7 Rede Siamesa 26
2.8 Caso de Estudo 28
2.9 Métricas de Avaliação 29

3 Objetivo **32**
3.1 Objetivo Geral 32
3.2 Objetivos Específicos 32
3.3 Contribuições 32

4 Atividades realizadas **34**
4.1 Conhecimento Prévio 34
4.2 Estudos Necessários 34
4.3 Testes 35
4.4 Protótipos 35
4.5 Método 36
4.6 Modelo 36
4.7 Cronograma 39

5 Projeto e Especificação do Sistema **40**
5.1 Funcionalidades 40

6 Implementação e Avaliação **46**
6.1 Planejamento e execução de testes funcionais 46

7 Conclusão e trabalhos futuros **50**

8 Referências bibliográficas **51**

Lista de figuras

Figura 1.1 ROV em atividades de inspeção submarina. **Fonte:**(ALVES, 2020) 16

Figura 2.1 Subáreas de Inteligência Artificial. **Fonte**:(HAND, 2020) 19
Figura 2.2 Exemplo de uma Rede Convolucional (CNN). **Fonte:**(SWAPNA, 2020) 21
Figura 2.3 Tipos de Aprendizagem. **Fonte:**(CHERNYSHOV, n.d.) 23
Figura 2.4 Equipamento Flange de um Duto. **Fonte:**(INTERNATIONAL, 2022) 26
Figura 2.5 Rede Siamesa. **Fonte:**(SILVA et al., 2022) 27
Figura 2.6 Dataset Omniglot (Alfabetos Manuscritos). **Fonte:**(KUZNETSOV, 2025) 28
Figura 2.7 Transformação Random Affine. **Fonte:**(KIM, 2025) 29

Figura 4.1 N-Way One-Shot. **Fonte:** Elaborado pelo autor. 35

Figura 5.1 Distribuição de amostras com classes únicas 41
Figura 5.2 Duto 42
Figura 5.3 ROV 42
Figura 5.4 Manipulador 42
Figura 5.5 Flange 42
Figura 5.6 Objeto 42
Figura 5.7 Equipamento 42
Figura 5.8 Classes Presentes do dataset de Inspeções Submarinas. 42

Lista de tabelas

Tabela 1.1	Versões das Bibliotecas Utilizadas	16
Tabela 4.1	Arquitetura da Rede Siamese para dataset Omniglot	38
Tabela 4.2	Arquitetura da Rede Siamese para Inspeções Submarinas	38
Tabela 4.3	Cronograma Projeto Final II	39
Tabela 6.1	Resultados referentes ao dataset Omniglot	46
Tabela 6.2	Melhores Resultados de Cada Caso Avaliado	48
Tabela 6.3	Resultados referentes ao dataset de Inspeções Submarinas	48

Lista de algoritmos

Algoritmo 1	Módulo de Pré-Processamento de Dados	42
Algoritmo 2	Módulo de Treinamento Treinamento	44

Lista de Abreviaturas

ADI – Análise Digital de Imagens

AUV – Autonomous Underwater Vehicle

CV – Computer Vision

FN – False Negative

FP – False Positive

ROV – Remote Operated Vehicle

SOTA – State of the art

Grad-CAM – Mapeamento de Ativação de Classe Ponderado por Gradiente

TN – True Negative

TP – True Positive

BIF – *Banded Iron Formation*

A good programmer is someone who always looks both ways before crossing a one-way street.

Doug Linder, .

1
Introdução

Para garantir a integridade e a segurança das operações *offshore* na indústria de óleo e gás, são realizadas inspeções submarinas regulares com o objetivo principal de monitorar e manter equipamentos submersos, como dutos, flanges, estruturas e válvulas. Essas inspeções utilizam recursos como vídeos captados por ROVs (LIAN; WEI, 2019) (Veículos Submarinos Operados Remotamente, do inglês *Remotely Operated Vehicle*) e AUVs (ZHOU; SI; CHEN, 2023) (Veículos Submarinos Autônomos, do inglês *Autonomous Underwater Vehicle*). No contexto da indústria de petróleo e gás, seu uso é voltado majoritariamente à detecção de anomalias, desgaste, obstruções e falhas que possam comprometer a operação segura dos sistemas submarinos. (PAULA, 2011).

Equipamentos com falhas aumentam o risco de acidentes com impactos ambientais, materiais e até humanos (ZHANG, 2023). Devido às condições adversas em que esses sistemas estão instalados, a manutenção preventiva é frequentemente realizada por meio de monitoramento remoto por vídeo, conhecido como inspeção submarina.

Nessas inspeções, as imagens são registradas em vídeo por meio de ROVs, e esses registros são fundamentais para a análise da integridade dos componentes e a identificação de anomalias ou intervenções necessárias. Após a coleta, os dados são analisados manualmente por especialistas, que avaliam visualmente a integridade das estruturas e registram qualquer sinal de dano externo que possa comprometer o funcionamento futuro. Trata-se de uma atividade trabalhosa, que demanda atenção e conhecimento técnico, dado o enorme volume de vídeos a ser inspecionado.

Durante toda sua vida útil, estruturas, equipamentos submarinos e suportes permanecem submersos em grandes profundidades, operando sob condições marinhas complexas e exigentes. Além do risco de eventos como corrosão, impactos mecânicos, sobrecarga estrutural, falhas por fadiga ou deslocamentos causados por correntes marítimas, que podem causar danos diretos aos equipamentos. Fatores como a salinidade e o pH da água também representam uma ameaça significativa aos componentes, favorecendo processos de corrosão, degradação de materiais e comprometimento da integridade estrutural ao longo do tempo (MELCHERS, 2003).

Na Figura 1.1 é apresentado um exemplo típico dessas inspeções, observa-se que nem todas as imagens que são obtidas serão nítidas e de rápido entendimento, mesmo para um especialista. Na Figura 1.1 apesar de apresentar

uma imagem em full HD (1920x1080), para imagens de inspeções submarinas, não é tão simples identificar quais tipos de equipamentos do ROV (medidor, manipulador, escova rotativa) estão presentes.

Este projeto se propõe a desenvolver uma solução original para um problema real de classificação de imagens submarinas com dados escassos, utilizando técnicas modernas de Inteligência Artificial, especificamente o Aprendizado One-Shot com Redes Siamesas. Treinando e avaliando um modelo de Deep Learning completo. Os dados são referentes a uma problemática real de uma empresa de óleo e gás.

A abordagem *One-Shot Learning* permite treinar um classificador eficaz mesmo com um número extremamente reduzido de amostras por classe. Essa característica é especialmente vantajosa no contexto de inspeções submarinas, onde é comum a necessidade de identificar novos tipos de objetos ou equipamentos ainda não observados pelo modelo. Diferentemente do aprendizado supervisionado tradicional, que exige a coleta de muitas amostras e o retreinamento completo do modelo sempre que uma nova classe é adicionada, o *One-Shot Learning* permite incluir e reconhecer novas classes com uma única imagem de suporte, mantendo o desempenho e reduzindo significativamente o custo computacional e o tempo de adaptação. Foram utilizadas classes de dutos, ROVs, manipuladores, flanges, equipamentos e objetos, ou seja, seis classes no total.

Trabalhos passados exploraram o uso de modelos supervisionados nesta mesma base de dados, avaliando diversos modelos de *Deep Learning* para classificação de imagens submarinas utilizando imagens de ROV. O modelo *DenseNet121* obteve entre 98% a 98.44% de acurácia, superando alternativas como VGG, *ResNet* e *MobileNet* (ATHIRA NAMBIAR, 2024). Apesar dessa contribuição, uma limitação deste trabalho é a chegada de novas classes, o que é comum nesses cenários de inspeções submarinas. Por isso, há a necessidade de explorar modelos *one-shot learning*.

Figura 1.1: ROV em atividades de inspeção submarina. **Fonte:**(ALVES, 2020)

O presente projeto pretende contribuir de forma significativa para a maior qualidade das decisões nessa área através do desenvolvimento de um conjunto de funcionalidades. Através da ferramenta proposta, o classificador será capaz de tratar o problema de classificação com dados reduzidos, acelerando o processo de tomada de decisão.

Para assegurar a reprodutibilidade dos experimentos, as bibliotecas e suas respectivas versões estão listadas na tabela 1.1.

Biblioteca	**Versão**
torch	2.1.0
torchvision	0.16.0
numpy	1.24.3
Pillow	9.5.0
scikit-learn	1.3.0

Tabela 1.1: Versões das Bibliotecas Utilizadas

Durante a construção de todo o projeto, foi aplicado o conhecimento adquirido durante o curso como programação orientada a objetos, engenharia de requisitos, definindo escopos, modularizando e realizando testes de código e inteligência artificial na aplicação de modelos siameses, realizando validação cruzada e análise de métricas como *F1-score*, precisão, *recall* e acurácia.

Inicialmente, foi construído um protótipo inicial com o dataset chamado *Omniglot*, de caracteres manuscritos de diversos alfabetos. Esse protótipo foi crucial para o projeto final, pois ele foi utilizado como referência para a construção do classificador referente ao dataset de inspeções submarinas. Como foram obtidos resultados muito satisfatórios no caso do protótipo com o dataset

Omniglot, como, por exemplo, 83% de acurácia no teste e 0.828 de F1-score no teste, foi usado parte do protótipo como base para o novo classificador.

Apesar de serem feitas mudanças, como por exemplo o acréscimo de regularizadores na arquitetura convolucional e de hiperparâmetros como learning rate e batch size, foi um excelente início para testes iniciais do dataset de inspeções submarinas.

Com os devidos ajustes, testes e validações, foi obtido um resultado de métricas como 0.764 de *F1 Score* e 76.5% de acurácia nos testes e 0.96 de *F1 Score* e 96.5% de acurácia no treinamento. Com a curva de perda se comportando adequadamente, caindo gradativamente, como o esperado. Assim como também foram realizadas diversas repetições desse treinamento para garantir a confiabilidade de seus resultados.

Os resultados deste trabalho serão enviados para um congresso da área de óleo e gás. O código referente ao projeto se encontra acessando esta página do **github**.

2
Situação Atual

2.1
Apresentação

Será realizada uma breve revisão teórica das tecnologias de *Machine Learning* e *Deep Learning* contextualizadas para o desafio atual. Assim como quais foram utilizadas e seus motivos por trás de cada escolha. A problemática relacionada às inspeções submarinas e seus desafios, em conjunto com os trabalhos relacionados, será abordada em sua amplitude.

2.2
Problemática Proposta

No monitoramento de leito marinho, há uma necessidade de manutenção constante dos equipamentos utilizados para fazer as inspeções submarinas. ROVs, dutos, objetos, equipamentos, acessórios, entre outros tipos de elementos, são constantemente requisitados e são danificados durante o tempo devido à salinidade da água, o pH e diversos eventos que podem causar até anomalias.

É realizada então uma manutenção preventiva a fim de prevenir tais danos de operação e eventualidades. Inspeções estas feitas pelos ROVs, as quais criam uma demanda exaustiva para especialistas analisarem cada amostragem retirada. Surgindo, por conseguinte, uma demanda para análises de diversos tipos de amostras para várias finalidades, não somente para conseguir detectar uma anomalia, mas também, como por exemplo, diferenciar uma vida marinha de uma parte de um equipamento.

Além da relevância econômica da produção em alto-mar, existe uma certa complexidade tanto relacionada às tecnologias utilizadas, quanto à logística dos insumos extraídos.

2.3
Trabalhos Relacionados

Neste âmbito, já houveram outras frentes a solucionar este desafio, como por exemplo utilizando uma abordagem com aprendizado semi e auto-supervisionado aplicado a classificação *multi-label* (PEREIRA, 2023) a qual propôs um novo método *mPAWS* para a classificação das imagens de inspeções submarinas, o qual adiciona ao *PAWS* uma etapa de pré-processamento inspirada na técnica de *label powerset*.

Outra frente relacionada à inspeção visual automatizada de dutos submarinos, a qual propõe um sistema de anotação automática que utiliza *Deep Learning* para identificar eventos específicos em amostras de inspeções submarinas (NILSTAD; SALBERG; AURDAL, 2020); entretanto, difere da abordada no presente trabalho. Neste caso, utilizando a *ResNet-50* com pesos pré-treinados no *ImageNet*, adaptada para classificação *multi-label* e com a *binary cross-entropy* como função de perda (a mesma função que será utilizada no projeto).

2.4 Deep Learning

O Aprendizado Profundo (*Deep Learning*) (GOODFELLOW; BENGIO; COURVILLE, 2016) é um subconjunto do aprendizado de máquina (*Machine Learning*) (Figura 2.1) que utiliza redes neurais com múltiplas camadas, chamadas redes neurais profundas. Ao invés de depender da extração manual de características (como em ML tradicional), redes profundas aprendem sozinhas os padrões dos dados.

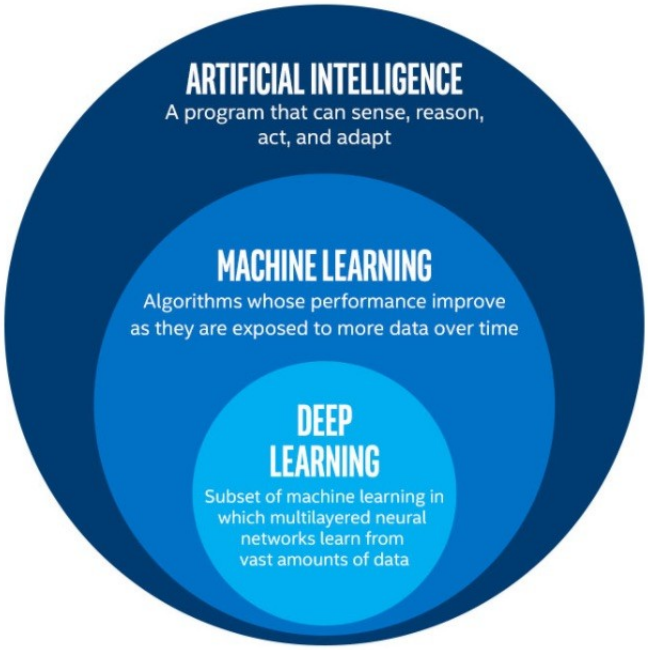

Figura 2.1: Subáreas de Inteligência Artificial. **Fonte**:(HAND, 2020)

Suas aplicações podem ser das mais diversas, como:

- Visão Computacional(REDMON et al., 2016): Reconhecimento facial, detecção de objetos, carros autônomos.
- Processamento de Linguagem Natural(VASWANI et al., 2017): Tradução automática, assistentes virtuais, análise de sentimentos.

– Multimodal(VINYALS et al., 2015): Assistentes Virtuais, Sistemas de descrição automática de imagens.

Visão computacional é o campo da inteligência artificial que busca dar aos computadores a capacidade de "ver" e interpretar o mundo visual, assim como os humanos fazem. Ela envolve técnicas para processar e analisar imagens ou vídeos, permitindo identificar objetos, reconhecer padrões, detectar movimentos, reconstruir cenas 3D, entre outras tarefas.

Já o Processamento de linguagem natural (PLN) é a área da inteligência artificial dedicada a permitir que sistemas entendam, interpretem e gerem a linguagem humana, seja falada ou escrita. Envolve desde tarefas simples, como correção ortográfica, até desafios complexos, como tradução automática, análise de sentimentos e resposta a perguntas. O PLN combina conhecimentos de linguística, ciência da computação e estatística para lidar com a ambiguidade e a complexidade do idioma humano.

E por fim, Multimodal refere-se a sistemas capazes de processar e integrar informações provenientes de diferentes modalidades de dados, como texto, imagens, áudio e vídeo, de forma combinada. Isso possibilita, por exemplo, que um modelo analise uma foto e sua legenda ao mesmo tempo, ou que entenda um vídeo considerando tanto a imagem quanto o áudio. Essa abordagem amplia o contexto e melhora a compreensão, permitindo aplicações mais ricas, como assistentes virtuais que interpretam voz e expressões faciais simultaneamente.

2.5 Arquitetura Convolucional

Redes Neurais Convolucionais (*Convolutional Neural Networks* – CNNs) são um tipo especializado de rede neural projetado para processar dados com estrutura de grade (LECUN et al., 1998), como imagens (2D) ou séries temporais (1D).

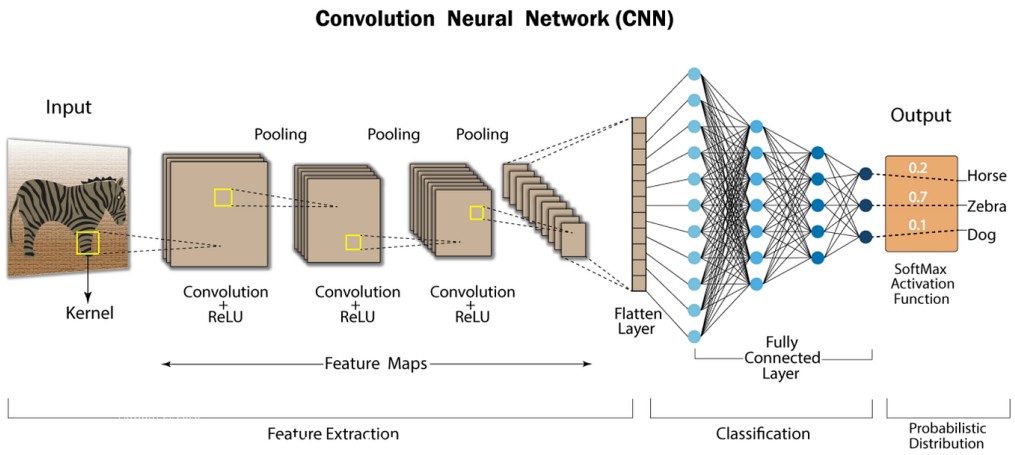

Figura 2.2: Exemplo de uma Rede Convolucional (CNN). **Fonte:**(SWAPNA, 2020)

A característica central dessas redes é o uso de camadas convolucionais(KRIZHEVSKY; SUTSKEVER; HINTON, 2012), que aplicam filtros (kernels) sobre a entrada para extrair padrões locais. Ao contrário das redes totalmente conectadas, que tratam todos os pixels como independentes, as CNNs exploram a correlação espacial entre vizinhos, capturando desde bordas e texturas nas primeiras camadas até formas complexas e padrões de alto nível nas camadas mais profundas.

Uma arquitetura (Figura 2.2) típica de CNN é composta por:

- Camadas Convolucionais (Conv2D): As quais aplicam filtros deslizantes sobre a entrada, gerando mapas de ativação que destacam padrões específicos. O tamanho do kernel e a quantidade de filtros controlam a granularidade e a diversidade das características extraídas.
- Funções de Ativação: ReLU (*Rectified Linear Unit*), sendo fácil de calcular e implementar, bem eficiente para redes neurais profundas.
- Normalização: Técnicas como *Batch Normalization* estabilizam o treinamento, acelerando a convergência e reduzindo a sensibilidade à inicialização dos pesos.
- *Pooling* (*MaxPooling* ou *AveragePooling*): Reduz a resolução espacial, condensando informações e tornando a rede mais robusta a pequenas variações na posição do objeto.
- Camadas Totalmente Conectadas (*Fully Connected*): Convertem os mapas de características extraídos em vetores de representação para classificação ou outras tarefas.
- *Dropout*: Técnica de regularização que desativa aleatoriamente neurônios durante o treinamento, evitando sobreajuste (*overfitting*).

No presente projeto, a CNN foi implementada dentro de uma rede neural siamesa, uma abordagem amplamente utilizada em aprendizado *One-Shot*. Nesse contexto, o objetivo não é classificar diretamente uma imagem, mas aprender uma função de similaridade entre pares de imagens. A rede recebe duas entradas e retorna um valor que indica quão semelhantes elas são.

2.6 Tipos de Aprendizagem

Os tipos de aprendizagem em inteligência artificial descrevem diferentes formas pelas quais um modelo pode ser treinado a partir dos dados disponíveis. Cada abordagem varia de acordo com a quantidade e o tipo de informação fornecida durante o treinamento, bem como o grau de intervenção humana necessária para rotular ou estruturar os dados. Entre as principais categorias estão a aprendizagem supervisionada, não supervisionada, semi-supervisionada e auto-supervisionada, cada uma com características próprias e aplicabilidade em diferentes cenários e desafios.

2.6.1 Aprendizado Supervisionado (*Supervised Learning*)

Este é um tipo de aprendizagem na qual o modelo aprende a partir de exemplos rotulados. Ou seja, para cada entrada, há uma saída esperada conhecida e o modelo tenta prever a saída correta com base nos dados de entrada.

Mais comumente utilizada em casos de Classificação (JORDAN; MITCHELL, 2015), e de Regressão. Podendo-se utilizar diversos tipos de algoritmos como: Regressão linear (HASTIE; TIBSHIRANI; FRIEDMAN, 2009), Árvores de decisão (QUINLAN, 1986), K-NN (COVER; HART, 1967), entre outros.

2.6.2 Aprendizado Não Supervisionado (*Unsupervised Learning*)

Nesse tipo de abordagem, o modelo opera sem acesso a rótulos, procurando estruturar os dados a partir de suas similaridades e diferenças. A ausência de uma resposta de referência torna a avaliação da performance um desafio, além de os resultados, muitas vezes, serem de difícil interpretação.

Seus tipos principais envolvem algumas tarefas como: *Clustering*, agrupando dados semelhantes, redução de dimensionalidade, representando os dados em espaços menores, mantendo as relações e modelagem de distribuição

de dados (JAIN; MURTY; FLYNN, 1999), entendendo como os dados estão distribuídos.

2.6.3 Aprendizado Semi-Supervisionado (*Semi-Supervised Learning*)

Usa uma pequena quantidade de dados rotulados combinada com uma grande quantidade de dados não rotulados (ZHU, 2005). O objetivo é propagar a informação rotulada pelos dados não rotulados, aproveitando padrões semelhantes.

Reduzindo a necessidade de rotular muitos dados(KINGMA et al., 2014), é ideal para domínios com dados caros de rotular. Entretanto, o modelo pode propagar rótulos incorretos se os dados forem ruidosos, e exige técnicas de regularização robustas. Atua-se em áreas de classificação de imagens com poucos rótulos, processamento de linguagem natural com apenas algumas frases anotadas e análise de sentimentos em domínios novos.

2.6.4 Aprendizado Auto-Supervisionado (*Self-Supervised Learning*)

Os rótulos serão gerados a partir dos próprios dados, servindo geralmente como pré-treinamento para redes neurais profundas, que depois são refinadas com poucos dados rotulados. Usados em casos de PLN, prevendo a próxima palavra ou palavras faltantes, ou de Visão Computacional, prevendo qual parte da imagem foi cortada, ou se duas imagens vêm do mesmo vídeo.

Permitindo usar grandes volumes de dados não anotados e requerendo zero rótulo manual, esse tipo de aprendizagem aprende representações genéricas e reutilizáveis como, por exemplo, *embeddings*, os quais serão discutidos mais à frente. Em contrapartida, o treinamento pode ser muito custoso, como em modelos de linguagem gigantes, e também em escolher a tarefa pretextual certa. BERT(DEVLIN et al., 2019), GPT(RADFORD et al., 2018), SimCLR(CHEN et al., 2020) e BYOL(GRILL et al., 2020) são alguns exemplos mais conhecidos dessa aplicação.

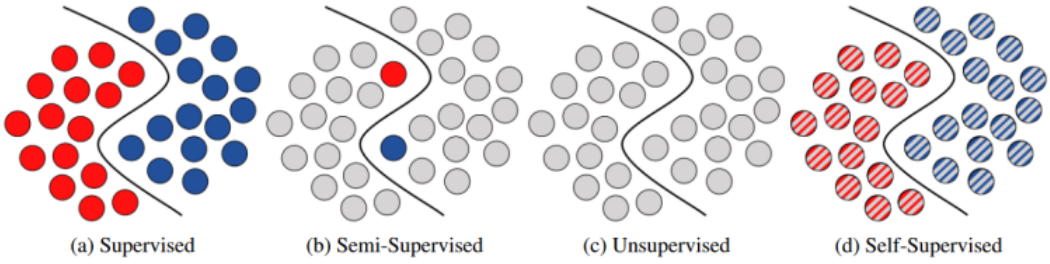

Figura 2.3: Tipos de Aprendizagem. **Fonte:**(CHERNYSHOV, n.d.)

Na Figura 2.3, mostram-se ilustrações das quatro estratégias de *Deep Learning* apresentadas: os círculos vermelho e azul-escuro representam pontos de dados rotulados de diferentes classes. Os círculos cinza-claro representam pontos de dados não rotulados. As linhas pretas definem o limite de decisão subjacente entre as classes. As classes listadas representam pontos de dados que ignoram e usam as informações do rótulo em diferentes estágios do processo de treinamento.

2.6.5 Low-data Learning

Low-data learning é um termo usado para descrever técnicas de aprendizado de máquina projetadas para funcionar bem quando se tem pouquíssimos dados rotulados disponíveis (HU et al., 2021) para treino — muito menos do que seriam necessários para métodos tradicionais de *deep learning*.

Contemplando vários tipos de aprendizados, os quais serão apresentados a seguir:

2.6.6 N-Way K-Shot

Em tarefas de aprendizado com poucas amostras, é comum utilizar a notação N-Way K-Shot (VINYALS et al., 2016) para descrever a configuração experimental do problema de classificação. Essa notação indica o número de classes envolvidas e a quantidade de exemplos disponíveis por classe durante a fase de aprendizado ou de avaliação.

O termo N-Way refere-se ao número de categorias (ou classes) distintas que compõem a tarefa. Por exemplo, em uma configuração 5-Way, o modelo deve distinguir entre cinco classes diferentes. Já o termo K-Shot representa a quantidade de exemplos (shots) fornecidos por classe. Assim, em um cenário One-Shot, o modelo recebe apenas uma amostra por classe para realizar a classificação; em um cenário 5-Shot, são disponibilizadas cinco amostras por classe, e assim por diante.

A estrutura N-Way K-Shot permite avaliar a capacidade do modelo de generalizar para novas classes com poucas amostras disponíveis, permitindo aprender de forma eficiente e robusta em situações de baixo volume de dados.

2.6.6.1 One-Shot Learning

One-Shot Learning (LAKE; SALAKHUTDINOV; TENENBAUM, 2013) é uma abordagem de aprendizado de máquina cujo objetivo é ensinar um

modelo a reconhecer novas classes a partir de apenas um exemplo por categoria. Diferente do aprendizado supervisionado tradicional, que exige milhares de imagens rotuladas, o *One-Shot Learning* busca generalizar rapidamente para classes nunca vistas antes.

Para isso, em vez de aprender classificadores fixos, o modelo aprende uma função de similaridade entre amostras. Assim, dado um exemplo de suporte (a classe de referência) e uma imagem de consulta, o sistema mede a distância entre suas representações, geralmente usando arquiteturas como redes siamesas. Se a similaridade for alta, conclui-se que pertencem à mesma classe.

Essa técnica é especialmente útil em cenários com poucos dados rotulados, como reconhecimento facial, diagnóstico médico ou inspeções industriais, pois permite reaproveitar o conhecimento aprendido em tarefas anteriores para lidar com novas categorias com mínima supervisão.

2.6.6.2 Few-Shot Learning

Caso seja possível obter um número mais alto de amostras por classes, pode-se utilizar o *Few-Shot Learning* (WANG et al., 2019). O *One-Shot* é um caso específico de *Few-Shot*, no qual o modelo aprende com apenas uma amostra por classe. Ambos são úteis quando os dados são escassos, mas o *Few-Shot* oferece um pouco mais de robustez ao modelo, já que ele vê mais exemplos.

2.6.6.3 Zero-Shot Learning

Apesar de atender à exigência relacionada ao número escasso de amostras, o qual neste caso não precisará de nenhuma amostra, a abordagem *Zero-Shot Learning* (SOCHER et al., 2013) enfrenta sérias limitações pela ausência de representações semânticas ricas e discriminativas.

Classes como ROV, Duto, Flange, são altamente visuais e específicas do domínio, e é difícil descrever semanticamente essas categorias de forma precisa e discriminativa para o modelo entender.

Neste caso, ele parte da premissa de que o modelo nunca viu exemplos visuais da classe durante o treinamento e precisa reconhecê-la com base em alguma representação semântica alternativa, como, por exemplo:

“Um duto é parecido com um tronco de árvore“

2.6.7 Similarity Learning

Similarity Learning (KOCH; ZEMEL; SALAKHUTDINOV, 2015) (ou Aprendizado por Similaridade) é uma abordagem dentro de *Machine Learning* cujo objetivo principal é aprender uma função que mede o quão semelhantes duas amostras são entre si. Ao invés de simplesmente classificar diretamente uma entrada como sendo de uma classe específica, o modelo aprende a comparar pares de exemplos e decidir se são do mesmo tipo (ou classe) ou não.

Será apresentado, portanto, um caso em que é apresentado um exemplo de uma classe como, por exemplo, uma flange (Figura 2.4), e o modelo dirá se a amostra é ou não uma flange, ou seja, uma classificação binária.

Figura 2.4: Equipamento Flange de um Duto. **Fonte:**(INTERNATIONAL, 2022)

Esta técnica é muito utilizada principalmente em casos como: Reconhecimento facial, verificação de assinaturas, busca por imagens semelhantes e classificação *one-shot* ou *few-shot*.

O *Similarity Learning* funciona muito bem para poucos exemplos por classe, além de ser escalável para muitas classes (sem treinar um classificador para cada) e adaptável para novas classes (sem precisar re-treinar tudo). Apesar disto, requer uma construção eficiente de pares ou trios de treinamento, é sensível à escolha da função de similaridade e da arquitetura de embedding.

2.7 Rede Siamesa

Uma Rede Siamesa é um tipo especial de rede neural simétrica, composta por duas (ou mais) sub-redes idênticas, entretanto diferenciando-se por uma inicialização aleatória de seus pesos(KOCH; ZEMEL; SALAKHUTDINOV, 2015). Ela foi criada para aprender a comparar dois objetos e estimar a

similaridade entre eles(BROMLEY et al., 1993). E tem como objetivo aprender uma função de similaridade entre pares de dados (por exemplo, duas imagens, textos ou sinais).

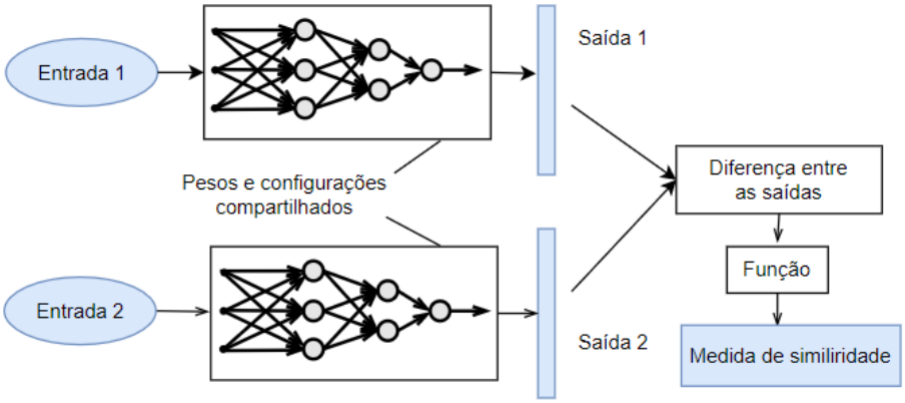

Figura 2.5: Rede Siamesa. **Fonte:**(SILVA et al., 2022)

Ela tem como entrada duas amostras, uma positiva e outra negativa (Figura 2.5). Assim, é extraída uma representação vetorial (embedding) de cada entrada, transformando cada entrada em um vetor numérico fixo que codifica suas principais características.

Em seguida, é calculada a distância entre os dois *embeddings*, na qual os embeddings serão concatenados passando por várias camadas densas, produzindo um valor escalar. Esse valor representa a semelhança (ou distância invertida) entre os dois inputs. Por fim, será passado para uma função de perda como, por exemplo, a *Binary Cross Entropy* (BCEWithLogitsLoss) (GOODFELLOW; BENGIO; COURVILLE, 2016) com o rótulo 1 se forem da mesma classe ou com o rótulo 0 se forem de classes diferentes.

A BCE é uma função de perda para classificação binária que mede a diferença entre as probabilidades previstas pelo modelo e os rótulos reais. Ela combina a sigmoide + *cross-entropy* em uma única etapa, tornando o cálculo mais estável numericamente.

Essa, por sua vez, tem a fórmula abaixo:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)\Big] \tag{2-1}$$

Onde:

- N → Número de amostras
- i → Índice da amostra
- y_i → Rótulo real (0 ou 1)
- p_i → Probabilidade prevista pelo modelo (entre 0 e 1)

– log → Logarítmo natural

Neste caso, a rede aprende uma função de similaridade própria, em vez de usar diretamente uma métrica explícita como a distância euclidiana.

2.8 Caso de Estudo

2.8.1 Projeto Omniglot

O projeto *Omniglot* (KOCH; ZEMEL; SALAKHUTDINOV, 2015) foi criado como um *benchmark* para estudar como algoritmos podem aprender novos conceitos a partir de poucos exemplos, inspirado em como humanos aprendem símbolos e alfabetos. Essa base de dados é comumente utilizada para comparar métodos de *one-shot/few-shot*.

Figura 2.6: Dataset Omniglot (Alfabetos Manuscritos). **Fonte:**(KUZNETSOV, 2025)

As amostras presentes no dataset *omniglot* (Figura 2.6) são de alfabetos escritos à mão, contemplando 50 alfabetos diferentes, sendo 30 deles utilizados para treinamento e 20 deles utilizados para teste. Dimensões de cada caractere de 105x105 pixels e com 20 amostras por caractere, sendo cada caractere de um alfabeto uma classe.

No artigo (KOCH; ZEMEL; SALAKHUTDINOV, 2015), os autores usaram uma rede neural siamesa com uma arquitetura convolucional profunda, projetada para comparar imagens ao invés de classificá-las diretamente.

Feito com os seguintes parâmetros:

– Otimizador: Adam
– *Learning Rate*: 0.0001
– *Batch Size*: 128

– *Way*: 20

– Épocas: 30.000 a 50.000

Obtendo resultados de 92% de acurácia no teste sem a transformação *RandomAffine* 2.7 e 96,7% de acurácia no teste com a *RandomAffine*.

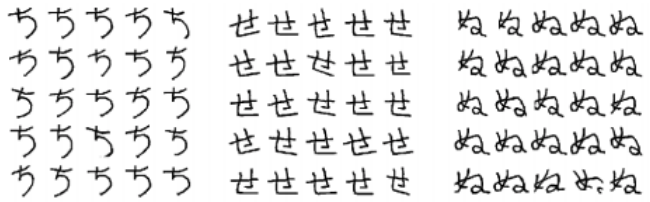

Figura 2.7: Transformação Random Affine. **Fonte:**(KIM, 2025)

Neste caso, foi utilizada uma abordagem baseada em episódios, chamados de *N-Way, K-shot*. Para cada episódio, escolhem-se N classes aleatórias para cada classe e fornece-se K imagens de suporte (como se trata da abordagem *One-Shot*, utiliza-se K = 1).

2.9 Métricas de Avaliação

É o que irá determinar o quão bom foi o modelo apresentado. Existem muitas métricas disponíveis para apresentar a performance de um modelo. Uma métrica errada pode levar a conclusões erradas, pois pode não capturar o problema real, não tornando-o capaz de avaliar o modelo de forma correta.

2.9.1 Acurácia

Irá dizer o quão boas são as previsões do modelo, analisando as previsões com o valor real, descobrindo os erros e calculando a acurácia. Entretanto, apesar dessa métrica ser muito utilizada em aprendizagem supervisionada, existe um problema ligado a ela chamado o Paradoxo da Acurácia (AFONJA, 2017). Para exemplificar a problemática, imagine uma base de dados de registros fraudulentos e 1% da base são registros fraudulentos e 99% desta base são registros não fraudulentos. Mesmo podendo ter uma acurácia de 99%, ao analisar a matriz de confusão, irá ser explicado que o algoritmo acertou todos da classe não fraudulenta, porém errou todos da classe fraudulenta, ou seja, qualquer input neste caso pode ser classificado como não fraudulento. Então este é um caso que pode-se obter uma acurácia muito boa, mas o modelo em si não aprendeu as diferenças entre as classes, registros. (DIGEST, 2023)

Sendo assim, é de extrema importância que qualquer projeto não seja avaliado somente se baseando no valor obtido na acurácia e também em outras métricas às quais serão abordadas a seguir, para que se tenha uma avaliação mais correta sobre o dado experimento.

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN}$$

- TP (*True Positives*): casos positivos corretamente classificados.
- TN (*True Negatives*): casos negativos corretamente classificados.
- FP (*False Positives*): casos negativos classificados incorretamente como positivos.
- FN (*False Negatives*): casos positivos classificados incorretamente como negativos.

2.9.2 Precisão

A precisão mede a proporção de predições positivas corretas entre todas as predições positivas feitas pelo modelo. Indica quão confiáveis são as predições positivas.

No qual uma alta precisão indica poucos falsos positivos.

$$\text{Precisão} = \frac{TP}{TP + FP}$$

2.9.3 Recall

O *recall* mede a proporção de casos positivos que foram corretamente identificados pelo modelo. Indica a capacidade do modelo de encontrar todos os positivos reais.

No qual um alto *recall* indica que tem poucos falsos negativos.

$$\text{Recall} = \frac{TP}{TP + FN}$$

2.9.4 F1-Score

O *F1-score* é a média harmônica entre precisão e recall, equilibrando as duas métricas. É útil quando há desequilíbrio entre classes.

No qual um alto *F1-Score* indica que existe um bom equilíbrio entre precisão e recall.

$$F1 = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}}$$

3
Objetivo

3.1
Objetivo Geral

O objetivo principal deste trabalho é solucionar o problema de classificação utilizando poucas amostras referentes a inspeções submarinas.

3.2
Objetivos Específicos

Os objetivos específicos do presente trabalho estão listados a seguir:

1. Realizar uma revisão bibliografica de arquiteturas convolucionais como a rede siamesa e de tipos de aprendizado *Low-Data* como o *One-Shot, Few-Shot* e *Zero-Shot*;

2. Propor um modelo baseado em redes siamesas para classificação de imagens *multi-label* explorando a abordagem *One-Shot Learning*;

3. Avaliar de forma quantitativa e qualitativa, para resolução do problema de classificação de imagens de inspeções submarinas.

3.3
Contribuições

As principais contribuições deste trabalho estão listadas a seguir:

1. Desenvolvimento de um classificador baseado em One-Shot Learning, utilizando redes siamesas convolucionais, capaz de reconhecer classes inéditas com base em apenas uma imagem de suporte por categoria — uma abordagem particularmente relevante para cenários com escassez de dados rotulados, como inspeções submarinas;

2. Avaliação quantitativa abrangente do modelo proposto, por meio de métricas como acurácia, precisão, recall, F1-score e análise da curva de perda, aplicada tanto no conjunto de treinamento quanto no de teste, permitindo verificar sua capacidade de generalização;

3. Análise qualitativa dos erros de classificação, com destaque para confusões recorrentes entre classes visualmente similares (ex: "Flange"vs. "Duto", "ROV"vs. "Equipamento"), além da investigação do impacto do

número de classes e exemplos por tarefa (cenários *N-Way K-Shot*), observando como a dificuldade da tarefa aumenta com a complexidade do cenário;

4. Exploração da distribuição das embeddings no espaço latente, com o objetivo de verificar se imagens semanticamente semelhantes são projetadas em regiões próximas — o que reforça a coerência interna do modelo de similaridade aprendido pela rede siamesa.

4
Atividades realizadas

4.1
Conhecimento Prévio

Antes de entrar no tema específico de redes neurais e *one-shot learning*, o autor já havia o domínio de fundamentos de programação, organização de dados e arquitetura de software, que foram a base necessária para implementar e treinar um classificador real de imagens submarinas.

4.2
Estudos Necessários

Além dos conhecimentos prévios, foi necessário aprofundar os estudos em várias camadas como:

- Aprendizado de máquina e *Deep Learning*: Conceitos fundamentais de redes neurais artificiais, funções de ativação, camadas convolucionais e técnicas de regularização.
- Redes Siamesas e *One-Shot Learning*: Estudo da arquitetura, do funcionamento e da aplicação em tarefas de reconhecimentos de padrões com poucos exemplos.
- Treinamento e Avaliação de Modelos: Métricas como acurácia, precisão, *recall*, *F1-score*, além de curvas de perda.
- Visão Computacional: Técnicas de pré-processamento de imagens, normalização, redimensionamento e organização de dataset.
- Conceitos de Otimização: Uso de otimizadores como *Adam*, funções de perda (BCEWithLogitsLoss, *Contrastive Loss*) e técnicas de ajuste de hiperparâmetros.

4.3 Testes

Foram realizados diferentes testes para validar o desempenho do classificador.

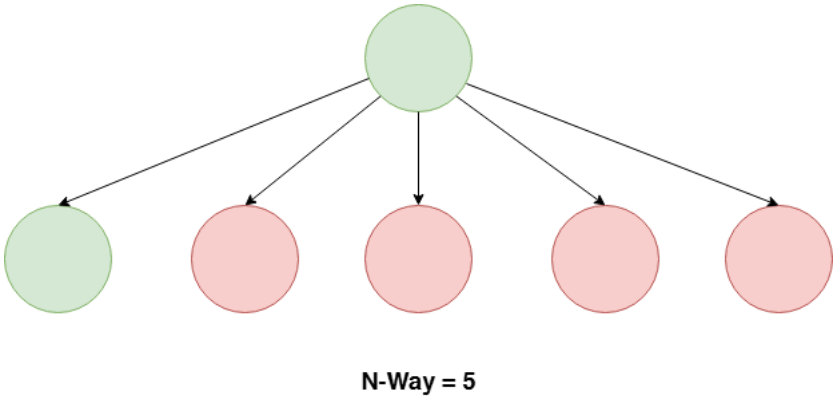

Figura 4.1: N-Way One-Shot. **Fonte:** Elaborado pelo autor.

O modelo foi avaliado em tarefas *N-way* (KOCH; ZEMEL; SALAKHUTDINOV, 2015), nas quais precisava identificar a classe correta entre várias opções, permitindo verificar sua capacidade de generalização. Na Figura 4.1, é simulado um exemplo com $N\text{-}Way = 5$, no qual é apresentada uma classe X, e será realizado um teste com outras 5 classes, sendo uma dessas classes igual à classe apresentada (X) e as outras 4 classes diferentes, e ele deverá responder adequadamente qual é a classe igual e quais são as diferentes.

Também foram conduzidos experimentos com diferentes combinações de classes de treino e teste, além de variações em hiperparâmetros como taxa de aprendizado, batch size e profundidade da rede. Durante o treinamento, métricas quantitativas como acurácia, precisão, *recall* e *F1-score* foram registradas periodicamente, possibilitando o acompanhamento da evolução do modelo e a identificação de padrões de erro entre as classes submarinas.

4.4 Protótipos

O projeto desenvolvido utilizando o dataset *Omniglot* pode ser considerado um protótipo experimental, pois serviu como prova de conceito para a aplicação de redes siamesas em tarefas de *one-shot learning*. Ele permitiu validar a arquitetura, testar o processo de treinamento e avaliação *N-way*, além de ajustar hiperparâmetros e funções de perda, fornecendo uma base segura para o desenvolvimento subsequente do classificador de imagens reais de inspeções submarinas.

Destacam-se a implementação de uma rede siamesa funcional em *PyTorch*, capaz de aprender representações discriminativas entre imagens, e a construção de um pipeline modular de treinamento e avaliação, com registros de métricas salvos em arquivos CSV para análise posterior. Além disso, foi desenvolvido um protótipo de avaliação one-shot, no qual o modelo consegue classificar novas imagens a partir de exemplos de suporte, simulando um cenário real de inspeções submarinas com poucos dados disponíveis.

4.5 Método

O processo de projeto e desenvolvimento seguiu um método experimental e iterativo, caracterizado pela construção, teste e refinamento progressivo do modelo. Inicialmente, foi realizado um estudo exploratório utilizando o dataset *Omniglot*, servindo como protótipo para validar a abordagem de *one-shot learning* com redes siamesas e ajustar hiperparâmetros, funções de perda e procedimentos de avaliação *N-way*. Com base nos resultados desse protótipo, foi definido o pipeline para o conjunto de imagens submarinas, contemplando atividades de pré-processamento, organização dos dados em classes de treino e teste, implementação da arquitetura siamese em *PyTorch* e criação de scripts modulares para treinamento e avaliação.

Durante o desenvolvimento, foram elaborados diferentes modelos experimentais: redes siamesas com variações na profundidade convolucional, tamanhos de *batch* e taxas de aprendizado, permitindo identificar a configuração mais eficiente para o problema. Paralelamente, foram conduzidos testes quantitativos (acurácia, precisão, *recall*, *F1-score*) e qualitativos (análise de pares de imagens) para monitorar a performance e refinar a arquitetura. Todo o processo seguiu um ciclo contínuo de experimentação, avaliação e ajuste, garantindo a construção de um classificador funcional e confiável para imagens de inspeções submarinas.

4.6 Modelo

O modelo desenvolvido para classificação de imagens de inspeções submarinas utiliza a abordagem de *Similarity Learning*, que consiste em aprender uma função de similaridade entre pares de dados, ao invés de aprender diretamente a classificar cada imagem em uma classe específica. Dessa forma, o modelo consegue avaliar se duas imagens pertencem à mesma categoria com base em suas características extraídas, permitindo a generalização para classes que não foram vistas durante o treinamento.

Para este projeto, foi adotada uma rede siamesa, arquitetura típica de *Similarity Learning*. Nela, cada par de imagens de entrada é processado por redes convolucionais compartilhadas, que geram embeddings vetoriais representativos das imagens. Em seguida, o modelo calcula a diferença absoluta entre os *embeddings* e, por meio de uma camada totalmente conectada com função de ativação sigmoide, prediz a probabilidade de similaridade entre as imagens. Essa estratégia permite aplicar a abordagem *one-shot learning*, já que novas classes podem ser reconhecidas a partir de apenas uma imagem de suporte.

A principal vantagem do *Similarity Learning* é a capacidade de generalização para novas classes, tornando o método particularmente adequado para inspeções submarinas, onde a coleta de imagens de cada equipamento é limitada. Além disso, o modelo pode ser avaliado em tarefas *N-way*, comparando uma imagem de teste com múltiplas imagens de suporte, o que possibilita medir seu desempenho de forma robusta mesmo com conjuntos de dados reduzidos. Comparado a outras abordagens de aprendizado supervisionado, o *Similarity Learning* oferece uma solução eficiente e escalável para cenários de *low-data learning*.

4.6.1 Arquitetura do Classificador Siamese

O classificador desenvolvido utiliza uma rede neural siamesa convolucional, com base na utilizada no artigo (KOCH; ZEMEL; SALAKHUTDINOV, 2015) referente ao dataset *Omniglot*. Foram adicionados regularizadores e pequenos ajustes nas camadas convolucionais, como mostrado na Tabela 4.1.

Layer	**Operation**	**Kernel / Stride**	**Channels**	**Output Size**
Encoder 1	Conv + ReLU	3×3, s=1	$1 \rightarrow 16$	105×105
MaxPool 1	MaxPooling	2×2, s=2	$16 \rightarrow 16$	52×52
Encoder 2	Conv + ReLU	3×3, s=1	$16 \rightarrow 32$	52×52
MaxPool 2	MaxPooling	2×2, s=2	$32 \rightarrow 32$	26×26
Encoder 3	Conv + ReLU	3×3, s=1	$32 \rightarrow 64$	26×26
MaxPool 3	MaxPooling	2×2, s=2	$64 \rightarrow 64$	13×13
Bottleneck	Conv + ReLU	3×3, s=1	$64 \rightarrow 128$	13×13
Upconv 1	Transposed Conv	2×2, s=2	$128 \rightarrow 64$	26×26
Decoder 1	Conv + ReLU	3×3, s=1	$64 \rightarrow 64$	26×26
Upconv 2	Transposed Conv	2×2, s=2	$64 \rightarrow 32$	52×52
Decoder 2	Conv + ReLU	3×3, s=1	$32 \rightarrow 32$	52×52
Upconv 3	Transposed Conv	2×2, s=2	$32 \rightarrow 16$	105×105
Decoder 3	Conv + ReLU	3×3, s=1	$16 \rightarrow 16$	105×105
Final	Conv + Sigmoid	1×1, s=1	$16 \rightarrow 1$	105×105

Tabela 4.1: Arquitetura da Rede Siamese para dataset Omniglot

E a partir dessa arquitetura, com a realização de testes e ajustes e obtenção de bons resultados para o caso referente ao dataset *Omniglot*, ela serviu como base para a arquitetura utilizada no dataset de inspeções submarinas, mostrada na Tabela 4.2. Com modificações como a adição de regularizadores e pequenos ajustes nas camadas *Conv2D* e *MaxPooling*, resultando na arquitetura convolucional final para o dataset de inspeções submarinas.

Layer	**Operation**	**Kernel / Stride**	**Channels**	**Output Size**
Encoder 1	Conv2D*	7×7, s=1	$1 \rightarrow 32$	$99 \times 99 \times 32$
MaxPool 1	MaxPooling	2×2, s=2	32	$49 \times 49 \times 32$
Encoder 2	Conv2D*	5×5, s=1	$32 \rightarrow 64$	$45 \times 45 \times 64$
MaxPool 2	MaxPooling	2×2, s=2	64	$22 \times 22 \times 64$
Encoder 3	Conv2D*	3×3, s=1	$64 \rightarrow 128$	$20 \times 20 \times 128$
MaxPool 3	MaxPooling	2×2, s=2	128	$10 \times 10 \times 128$
Bottleneck	Flatten*	-	$128 \times 10 \times 10 \rightarrow 512$	512
Decoder 1	FC*	-	$512 \rightarrow 128$	128
Final	FC	-	$128 \rightarrow 1$	1

Tabela 4.2: Arquitetura da Rede Siamese para Inspeções Submarinas

Onde:

- Conv2D* = Conv2D + Batch Normalization + ReLU
- Flatten* = Flatten + Fully Connected + Batch Normalization + ReLU + Dropout
- FC* = Fully Connected + Sigmoid

– FC = Fully Connected

Foi utilizada uma rede neural siamesa com CNN, treinada para aprender uma função de similaridade entre pares de imagens. Mas, além da siamesa, teríamos outros modelos que poderiam ser utilizados.

4.7 Cronograma

Sabendo que as etapas estão da seguinte forma:

– **Etapa 1**: Revisão Bibliográfica de Redes Neurais
– **Etapa 2**: Revisão Bibliográfica de One-Shot Learning
– **Etapa 3**: Revisão Bibliográfica de Engenharia Submarina
– **Etapa 4**: Tratamento dos Dados
– **Etapa 5**: Modelagem
– **Etapa 6**: Treinamento do Modelo
– **Etapa 7**: Teste do Modelo
– **Etapa 8**: Validação do Modelo
– **Etapa 9**: Escrita do Relatório

Foi remanejado e reorganizado o cronograma do relatório do projeto final I devido a um atraso que ocorreu no início do projeto final II. Desta forma, foi iniciado o projeto final II em janeiro de 2025 e com seu roteiro adaptado (Tabela 4.3) para este período até o fim de agosto de 2025.

	JAN	**FEV**	**MAR**	**ABR**	**MAI**	**JUN**	**JUL**	**AGO**
Etapa 1								
Etapa 2								
Etapa 3								
Etapa 4								
Etapa 5								
Etapa 6								
Etapa 7								
Etapa 8								
Etapa 9								

Tabela 4.3: Cronograma Projeto Final II

5
Projeto e Especificação do Sistema

Este capítulo apresenta uma descrição detalhada dos módulos, estruturas, assim como características marcantes da solução. As contribuições mais marcantes e os desafios enfrentados durante a elaboração, execução e teste do projeto.

5.1
Funcionalidades

5.1.1
Módulo de Preparação do Ambiente

Para garantir reprodutibilidade, portabilidade e consistência no treinamento e avaliação do modelo, foi desenvolvido um módulo de ambiente computacional baseado em contêineres. Esse módulo utiliza *Docker* para a criação e configuração do ambiente, permitindo que todas as dependências, bibliotecas (referenciadas na Tabela 1.1 e versões de *software* fossem explicitamente definidas em um único arquivo de configuração (*Dockerfile*).

A partir desse arquivo, foi gerada uma imagem *Docker*, que serve como base para execução local ou em *clusters* de computação. Para a execução em ambientes de *cluster* que requerem maior controle e isolamento, a imagem *Docker* foi convertida em um contêiner *Singularity* no formato .sif. O uso do *Singularity* garante compatibilidade com infraestruturas de HPC (*High Performance Computing*), mantendo o mesmo ambiente definido pelo Docker, sem necessidade de privilégios administrativos para execução.

Esse módulo contribui significativamente para a reprodutibilidade e confiabilidade do projeto, permitindo que os experimentos sejam replicados de forma consistente em diferentes máquinas e infraestruturas, além de facilitar a escalabilidade para execução em clusters de alta performance.

5.1.2
Módulo de Pré-Processamento de Dados

O primeiro passo consistiu no pré-processamento das imagens utilizadas para treinamento, validação e teste. Como as imagens presentes no conjunto de inspeções submarinas possuíam diferentes dimensões, formatos e qualidades, foi necessária a padronização para permitir que fossem processadas adequadamente pela rede neural.

Foi realizado um resize de 105x105, já seguindo como referência o *Omniglot*. As amostras poderiam ter tamanhos variáveis, então o resize era necessário. Transformações de reflexão horizontal e de rotação entre -20º e 20º, variações de luz e também a transformação afim aleatória na imagem (essa última também utilizada no dataset *omniglot*).

Uma normalização dos píxeis foi aplicada com valores para o intervalo [0,1] permitindo o aprendizado do modelo devido ao gradiente *vanishing*. As imagens foram organizadas por classes, em pastas correspondentes às suas respectivas categorias, ou seja, 6 pastas referentes às seguintes classes: Duto, ROV, Flange, Equipamento, Manipulador, Objeto. O melhor balanceamento encontrado foi com 1000 amostras para a maioria das classes, sendo que as classes Duto e ROV atingiram valores maiores que 1000 amostras, mas as demais conseguiram ter quantidade semelhante de amostras, como mostrado na figura 5.1

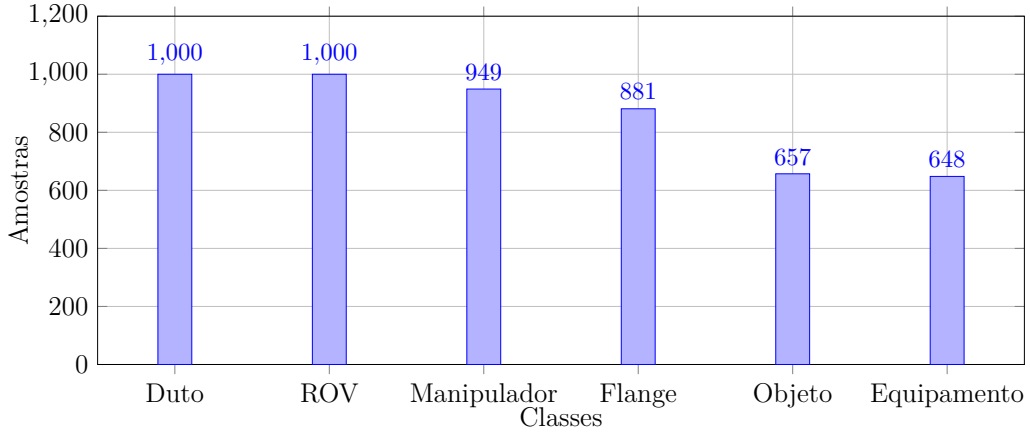

Figura 5.1: Distribuição de amostras com classes únicas

Estas amostras de classes são únicas, ou seja, cada amostra da classe Duto só contém a classe Duto e nenhuma outra classe. Vale lembrar que outras classes também foram testadas como Cruzamento, Fundo Mar, Vida Marinha, Sucata, Corda, etc. mas não obtiveram resultados satisfatórios devido à compatibilidade com as demais classes e principalmente devido ao número de amostras que acabava por ser muito reduzido. Encontrar o balanceamento ideal de amostras e a compatibilidade entre as classes foi um dos grandes desafios deste projeto.

Para o começo do pré-processamento, foram aplicados redimensionamentos (105x105) e data augmentations como flips horizontais, rotações de -20º a 20º, alteração de cor e transformações afins aleatórias. E, por fim, a normalização dos pixels em (0,1) e dividindo as amostras para dois datasets, sendo o dataset de treinamento com 70% delas e o de teste com 30% delas. A constru-

ção desse processamento de dados foi realizada tendo em base o algoritmo 1 descrito abaixo:

Algoritmo 1: Módulo de Pré-Processamento de Dados

Entrada: Caminho dos dados e tamanho da imagem

Saída: Dataset de treino e dataset de teste

1 *Carregue as imagens a partir do caminho fornecido*;
2 *Aplique as transformações de pré-processamento:*;
3 - Redimensionamento (105×105);
4 - Data Augmentation: flip, rotação, cor, afinamento;
5 *Normalize os pixels para o intervalo* $[0, 1]$;
6 *Divida o dataset em treino e teste*;
7 **retorna** dataset_treino, dataset_teste;

5.1.2.1 Detalhamento das Classes

As classes finais escolhidas foram as seguintes: Duto, ROV, Manipulador, Flange, Objeto e Equipamento. É possível visualizar um exemplo de cada uma delas na Figura 5.8 abaixo:

Figura 5.2: Duto

Figura 5.3: ROV

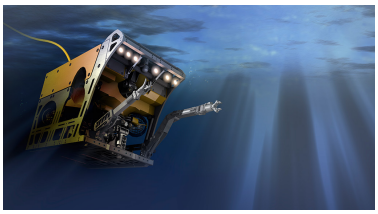

Figura 5.4: Manipulador

Figura 5.5: Flange

Figura 5.6: Objeto

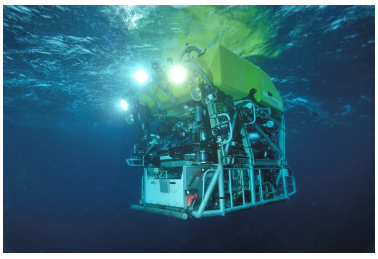

Figura 5.7: Equipamento

Figura 5.8: Classes Presentes do dataset de Inspeções Submarinas.

O Duto (Figura 5.2) é um tubo utilizado para transporte de fluidos, como petróleo ou gás no fundo do mar. Um ROV (Figura 5.3) é um veículo submersível controlado remotamente, utilizado para inspeções e manutenções em estruturas submarinas. O Manipulador (Figura 5.4) é um braço robótico

presente no ROV, permitindo interagir com objetos e realizar tarefas precisas, como manusear ferramentas. Uma Flange (Figura 5.5) é um componente de conexão entre dutos ou equipamentos para garantir vedação e alinhamento. Um objeto (Figura 5.6) refere-se a qualquer item encontrado no ambiente submarino durante as inspeções. E já um equipamento (Figura 5.7) abrange instrumentos ou dispositivos instalados no fundo do mar, como sistemas de monitoramento. As imagens das classes apresentadas acima são imagens públicas coletadas na internet para fins meramente ilustrativos.

5.1.3 Módulo de Modelagem - Rede Siamesa

A arquitetura central é uma rede siamese convolucional, composta por quatro camadas convolucionais seguidas de normalização, funções de ativação *ReLU* e *pooling*. As saídas das redes gêmeas são transformadas em embeddings vetoriais, que são comparados utilizando uma função de similaridade para determinar se as imagens pertencem à mesma classe. Diferentes configurações de profundidade e parâmetros foram testadas para otimização do desempenho.

Mesmo utilizando como base a arquitetura convolucional do dataset *Omniglot* foram feitos alguns ajustes como a adição de uma camada linear 1281010,512 e do normalizador *BatchNorm2d* que utiliza valores de 32, 64 e 128 nas camadas convolucionais 1, 2 e 3. Já na camada linear foram adicionados também um *BatchNorm2d* de 512, uma ReLU e também um regularizador *Dropout* de 0.5. Vale lembrar que esses dados estão representados também na Figura 4.2. E por fim, antes da sigmoide, foi adicionada mais uma camada linear de 512, 128. As funções ReLU e sigmoide foram utilizadas em suas configurações padrão.

5.1.4 Módulo da Função de Perda e Otimizador

Foi utilizado como função de custo a *Binary Cross-Entropy* com *Logits* (BCEWithLogitsLoss). Para otimizar os parâmetros da rede siamesa, foi utilizado o algoritmo Adam com *weight-decay* de 1e-4 e com 0.0001 de *learning rate*.

5.1.5 Módulo de Treinamento do Modelo

O módulo de treinamento foi responsável por ajustar os pesos da rede siamesa utilizada na classificação de imagens submarinas com *One-Shot Learning*. O processo foi conduzido em épocas, nas quais, para o dataset *omniglot*

foram 225 e para o de inspeções submarinas foram 600. Para cada lote de dados, eram gerados pares de imagens compostos por amostras positivas (mesma classe) e negativas (classes diferentes).

Apresentado abaixo no algoritmo 2, o procedimento completo do treino e teste das redes siamesas, chamando as funções de perda e o otimizador descrito no módulo anterior e salvando os parâmetros treinados, totalizando 38 milhões de parâmetros, em um arquivo .pth.

Algoritmo 2: Módulo de Treinamento Treinamento

Entrada: Modelo, *train_loader*, *dataset_treino*, *dataset_teste*, *max_iter*, *show_every*, *save_every*, *test_every*, *n_way*, *times*

Saída: Modelo treinado, métricas de treino e teste

para $epocas = 1$ **até** *max_epocas* **faça**
- Obtenha o próximo batch $(img1, img2, label)$ do **train_loader**;
- Envie os dados para o *device*;
- Zere os gradientes acumulados;
- $output \leftarrow modelo(img1, img2)$;
- $loss \leftarrow perda(output, label)$;
- $loss\_val \leftarrow loss\_val + loss.item()$;
- $loss.backward()$;
- $otimizador.step()$;
- **se** *epocas* mod *show_every* $= 0$ **então**
  - Exiba a perda média do intervalo;
  - Salve o valor em *train_loss*;
  - $loss\_val \leftarrow 0$;
- **se** *epocas* mod *save_every* $= 0$ **então**
  - Salve o modelo parcial: `model-batch-{epocas}.pth`;
- **se** *epocas* mod *test_every* $= 0$ **então**
  - Métricas_Treino $\leftarrow$ AVALIAR(*modelo*, *dataset_treino*, *n_way*, *times*);
  - Métricas_Teste $\leftarrow$ AVALIAR(*modelo*, *dataset_teste*, *n_way*, *times*);
  - Salve as métricas no CSV de logs;

5.1.6 Módulo de Avaliação Final

O módulo de avaliação final foi responsável por mensurar o desempenho do modelo baseado em *One-Shot Learning* com redes siamesas. A avaliação foi realizada utilizando tarefas do tipo *N-way*, sendo no dataset *omniglot* com N=20 e no submarina com N=3, nas quais o modelo recebe uma imagem de suporte por classe e uma imagem de consulta, calculando os vetores de *embeddings* e classificando a imagem com base na menor distância entre embeddings.

Foram calculadas métricas globais e por classe, incluindo acurácia, precisão, *recall* e *F1-score*. Todos os resultados foram salvos automaticamente em arquivos CSV, facilitando o acompanhamento e a comparação entre diferentes experimentos.

Essa abordagem possibilitou avaliar tanto o desempenho do modelo sobre dados conhecidos quanto a sua capacidade de generalização para classes não vistas durante o treinamento, requisito essencial para aplicações reais em inspeções submarinas.

6
Implementação e Avaliação

Foi utilizada uma arquitetura baseada em contêineres, implementando o ambiente de desenvolvimento com Docker e, posteriormente, convertendo a imagem para o formato Singularity e utilizando uma GPU Tesla V100-SXM2 de 16 GB.

6.1
Planejamento e execução de testes funcionais

Nesta sessão, será apresentada como foram realizados os testes e quais foram os resultados referentes tanto ao dataset *omniglot*, quanto ao dataset de inspeções submarinas. Quais foram os seus hiperparâmetros, número de execuções e outras variáveis decorrentes do processo de testes.

6.1.1
Caso Omniglot

Os testes referentes ao dataset *Omniglot* já tiveram bons prospectos desde o seu início. Neste dataset, as classes utilizadas no treinamento são diferentes das classes utilizadas para teste, ou seja, as classes do teste não são vistas no treinamento.

Cada experimento foi executado pelo menos 5 vezes para a obtenção de um resultado mais confiável e estável. Os resultados de cada experimento podem ser vistos na Tabela 6.1

Way	**Times**	**Learning Rate**	**Batch Size**	**Regul.**	**Otimiz.**	**Nov. Arq.**	**F1-Score**
20	400	0.0006	128	NÃO	Adam	-	0.854
20	400	0.0006	64	NÃO	Adam	-	0.836
20	400	0.006	128	NÃO	Adam	-	0.835
20	400	0.0006	32	NÃO	Adam	-	0.787
20	400	0.0001	128	NÃO	Adam	-	0.780

Tabela 6.1: Resultados referentes ao dataset Omniglot

Lembrando que para este caso tiveram um total de 1.137 classes e 32.460 amostras, com o N = 20 e *Times* = 400 em todos os casos

Realizando testes para o caso *omniglot* foi possível perceber uma variação apresentada pelos hiperparâmetros *Learning Rate* e *Batch Size* principalmente. No *Batch Size* a diminuição dele para valores de 64 e 32 causava já uma piora no rendimento, provavelmente proveniente de deixar o gradiente mais ruidoso, escapando de mínimos locais. A partir disto, foi encontrado o learning rate que equilibra com o valor de *Batch Size* encontrado, que apesar de ser o menor

entre os testados, a maior aproximação de um treinamento mais estável e preciso resultou em um melhor *F1-Score*.

Dada a agilidade e eficiência dos testes realizados referentes ao dataset *omniglot*, não foi preciso uma quantidade demasiado alta de testes e variações de hiperparâmetros para alcançar um resultado satisfatório.

6.1.2 Caso Inspecoes Submarinas

Serão apresentados 3 casos: O primeiro é referente a classes iguais, ou seja, as classes vistas no treinamento também serão vistas no teste; contudo, as amostras que aparecerão no treinamento não aparecerão no teste. O segundo caso é referente a classes diferentes, ou seja, as classes que aparecerão no treinamento não aparecerão no teste, e vice e versa. E o terceiro e último caso é referente a classes distratoras, ou seja, existirão classes no teste que não aparecerão no treino; porém, será adicionada uma quantidade bem pequena de amostras de algumas classes do treinamento, chamadas de distratoras.

1. **Treino e teste nas mesmas classes**: validação interna, para verificar se o modelo era capaz de aprender relações consistentes dentro de um mesmo conjunto.

2. **Treino e teste em classes distintas**: simulando o cenário real de One-Shot Learning, em que o modelo precisa generalizar para categorias nunca vistas anteriormente.

3. **Casos de distratores**: inclusão de imagens sem relação direta com a classe alvo, avaliando a robustez do sistema.

Dentre esses 3 casos, o caso de Treino e Teste com as mesmas classes foi o que obteve os melhores resultados, chegando a obter no treinamento 96.5% de acurácia e 0.96 de *F1-Score* e no teste 76.5% de acurácia e 0.764 de *F1-Score*, e também com a curva da loss decaindo como previsto.

É válido notar que para o caso do dataset *Omniglot*, o caso de treino e teste com classes distintas foi o que obteve o melhor resultado; entretanto, para o dataset de Inspeções Submarinas, o caso de treino e teste com classes iguais foi o que apresentou melhores resultados.

Uma consideração importante sobre o caso de Treino e Teste em classes distintas no dataset de Inspeções Submarinas, apesar de não ter tido o melhor resultado, apresentou uma evolução significativa (Tabela 6.2 com um aumento do número de amostras, mostrando que, dependendo do número de amostras utilizadas, poderá atingir resultados ainda melhores que os atuais.

	Nº de Classes	**Nº Max de Amostras**	**F1 Teste**
I. Classes Iguais	6	1000	0.764
	5		0.707
II. Classes Diferentes	6	250	0.520
	7	1000	0.408
III. Classes Distratoras	6	1500	0.545

Tabela 6.2: Melhores Resultados de Cada Caso Avaliado

Para todos os casos, foram realizadas diversas variações de hiperparâmetros, como por exemplo mudanças em valores de *Learning Rate*, *Batch Size*, transformações como *Random Affine*, vertical, horizontal, *Weight Decay* e *Dropout*.

Lembra-se que para cada variação, o sistema foi executado pelo menos de 3 a 5 vezes para garantir a sua estabilidade e segurança nos dados registrados.

Abaixo na Figura 6.3 , estão todos os experimentos realizados com todos os resultados obtidos:

Caso	**Classes**	**Amostras**	**Way**	**Times**	**LR**	**BS**	**R.**	**Oti.**	**Nov. Arq.**	**F1-Score**
I	6	1000	3	200	0.0001	128	SIM	Adam	-	0.764
I	5	1000	3	200	0.0001	128	SIM	Adam	-	0.707
I	6	1000	3	200	0.0001	128	SIM	Adam	SIM	0.600
I	6	1500	3	200	0.0001	128	SIM	Adam	-	0.600
I	6	1000	3	200	0.0001	128	SIM	Adam+L2	-	0.560
I	6	1500	3	200	0.0001	128	NÃO	Adam	-	0.556
III	6	1500	3	200	0.0006	64	NÃO	Adam	-	0.545
III	6	1500	3	200	0.0001	64	NÃO	Adam	-	0.540
II	7	250	2	200	0.0006	128	NÃO	Adam	-	0.520
I	7	250	5	200	0.0001	128	NÃO	Adam	-	0.505
III	6	1000	3	200	0.0001	64	SIM	Adam+L2	SIM	0.489
I	7	250	5	200	0.0006	128	NÃO	Adam	-	0.470
III	6	1000	3	200	0.0001	64	SIM	Adam	SIM	0.469
I	6	1500	4	200	0.0001	128	NÃO	Adam	-	0.437
II	6	1000	3	200	0.0001	128	SIM	Adam	-	0.408
I	7	500	7	300	0.0006	128	NÃO	Adam	-	0.286
I	7	250	5	200	0.0006	64	NÃO	Adam	-	0.220
II	6	1500	2	200	0.0006	64	NÃO	Adam	-	0.190
I	6	1500	5	200	0.0006	128	NÃO	Adam	-	0.170
II	7	500	5	300	0.0006	128	NÃO	Adam	-	0.150
I	7	500	5	400	0.0006	128	NÃO	Adam	-	0.100

Tabela 6.3: Resultados referentes ao dataset de Inspeções Submarinas

Onde:

- LR = *Learning Rate*
- BS = *Batch Size*

– R. = Uso de Regularizadores

– Otim. = Otimizador

Foi visto já de início nos testes, que o caso II era um caso muito difícil de aperfeiçoar para este projeto, pois era necessário avaliar quais classes seriam escolhidas para ficar exclusivamente para o teste, o que, somado ao tempo de execução do classificador, tornava-se uma tarefa muito custosa.

Já o caso III, apesar de também haver classes exclusivas no teste, com a adição de classes distratoras, acabou se provando mais eficiente; entretanto, ainda com as mesmas dificuldades pertinentes ao caso II, como identificar quais classes seriam as mais adequadas para serem testadas e quais seriam as melhores candidatas a serem distratoras.

Mesmo com uma avaliação detalhada para esses dois casos, acabaram provando não satisfatórios os resultados apresentados por eles.

Já para o caso I, tanto foi possível uma maior agilidade de treinamentos e testes, como também uma maior flexibilidade para alteração das classes às quais seriam utilizadas. Classes como Sucata, Vida Marinha e Cruzamento também foram presentes em treinamentos e testes, apesar de não serem escolhidas para serem as melhores candidatas para a obtenção do resultado final.

7
Conclusão e trabalhos futuros

O presente trabalho demonstrou a aplicabilidade de redes siamesas e aprendizado *One-Shot* para classificação de imagens de inspeções submarinas com poucas amostras por classe, contribuindo tecnicamente para sistemas automatizados e academicamente para estudos de one-shot learning em cenários industriais. Foram desenvolvidas metodologias de avaliação detalhadas, permitindo análise precisa do desempenho do modelo. Durante o projeto, foi possível aprofundar conhecimentos sobre *One-Shot Learning*, pré-processamento de imagens complexas e avaliação de modelos em conjuntos de dados desbalanceados, além de experiências práticas em ambientes controlados com *Docker* e *Singularity*. Limitações como sensibilidade a classes visualmente semelhantes, necessidade de ajustes de hiperparâmetros e potencial de melhoria com arquiteturas alternativas foram identificadas. Foram obtidos resultados satisfatórios para ambos os datasets, sendo 0.854 e 0.764 os valores de *F1-Score* do teste para os respectivos datasets *Omniglot* e de inspeções submarinas. Para trabalhos futuros, recomendam-se estratégias de meta-aprendizado, aprendizado auto-supervisionado, aumento de dados sintéticos, integração em inspeção em tempo real e visualização interpretável, e diminuição da complexidade da arquitetura convolucional, oferecendo oportunidades claras para a evolução do sistema e continuidade acadêmica.

8
Referências bibliográficas

ALVES, A. *O uso do ROV em atividades de inspeção submarina*. 2020. Acesso em: 01 set. 2025. Disponível em: <https://www.linkedin.com/pulse/o-uso-do-rov-em-atividades-de-inspe%C3%A7%C3%A3o-submarina-adan-alves/>. Citado 2 vezes nas páginas 9 e 16.

HAND, D. J. *The Relationship Between Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL)*. 2020. Acesso em: 01 set. 2025. Disponível em: <https://danieljhand.com/the-relationship-between-artificial-intelligence-ai-machine-learning-ml-and-deep-learning-dl.html>. Citado 2 vezes nas páginas 9 e 19.

SWAPNA. *Convolutional Neural Network | Deep Learning*. 2020. Acesso em: 01 set. 2025. Disponível em: <https://developersbreach.com/convolution-neural-network-deep-learning/>. Citado 2 vezes nas páginas 9 e 21.

CHERNYSHOV, V. *Self-Supervised Machine Learning: the story so far and trends for 2021*. n.d. Acesso em: 01 set. 2025. Disponível em: <https://hackernoon.com/self-supervised-machine-learning-the-story-so-far-and-trends-for-2021-sk1u34ad>. Citado 2 vezes nas páginas 9 e 23.

INTERNATIONAL, I. O. *Oceaneering é reconhecida no Pipeline and Gas Journal Awards*. 2022. Acesso em: 01 set. 2025. Disponível em: <https://www.oceaneering.com/pt/oceaneering-recognized-at-pipeline-and-gas-journal-awards/>. Citado 2 vezes nas páginas 9 e 26.

SILVA, F. C. e et al. Aprendizado few shot learning com redes neurais siamesas aplicado na classificação para e-commerce. In: *Anais do Congresso Brasileiro de Automação (CBA) 2022*. [s.n.], 2022. Acesso em: 01 set. 2025. Disponível em: <https://www.sba.org.br/cba2022/wp-content/uploads/artigos_cba2022/paper_2217.pdf>. Citado 2 vezes nas páginas 9 e 27.

KUZNETSOV, I. u. s. *Implementation of Siamese-Networks for One-Shot Learning in TensorFlow 2.0*. 2025. Repositório arquivado em 30 ago. 2025; Acesso em: 01 set. 2025. Disponível em: <https://github.com/schatty/siamese-networks-tf>. Citado 2 vezes nas páginas 9 e 28.

KIM, D.-K. *Siamese Nets: A Breakthrough in One-shot Image Recognition*. 2025. Medium; Acesso em: 01 set. 2025. Disponível em: <https://medium.com/@kdk199604/siamese-nets-a-breakthrough-in-one-shot-image-recognition-53aa4a4fa5db>. Citado 2 vezes nas páginas 9 e 29.

LIAN, L.; WEI, Z. Remotely operated vehicle (rov). In: CUI, W.; FU, S.; HU, Z. (Ed.). *Encyclopedia of Ocean Engineering*. [S.l.]: Springer, Singapore, 2019. p. 1–11. Citado na página 14.

ZHOU, J.; SI, Y.; CHEN, Y. A review of subsea auv technology. *Journal of Marine Science and Engineering*, v. 11, n. 6, p. 1119, 2023. Citado na página 14.

PAULA, R. V. de. Pipeline and gas journal awards. Marinha do Brasil, p. 59, 2011. Disponível em: <https://www.oceaneering.com/pt/oceaneering-recognized-at-pipeline-and-gas-journal-awards/>. Citado na página 14.

ZHANG, e. a. Probability analysis and prevention of offshore oil and gas accidents: Fire as a cause and a consequence. *R Discovery or similar*, 2023. Discute como falhas em equipamentos em plataformas offshore podem causar incêndios, explosões e liberações de hidrocarbonetos com consequências severas para o meio ambiente e pessoas. Citado na página 14.

MELCHERS, R. E. Marine corrosion of steel structures: Progress and challenges. *Corrosion Science*, Elsevier, v. 45, n. 5, p. 833–840, 2003. Citado na página 14.

ATHIRA NAMBIAR, N. P. Underwater sonar image classification and analysis using lime-based explainable artificial intelligence. *arXiv preprint arXiv:2408.12837*, 2024. DenseNet121 alcançou 98.21% de acurácia em teste e validação, com XAI usando LIME. Citado na página 15.

PEREIRA, A. L. *Aprendizado Semi e Auto-supervisionado aplicado à classificação multi-label de imagens de inspeções submarinas*. Dissertação (Mestrado) — PUC-Rio, 2023. Citado na página 18.

NILSTAD, J.; SALBERG, A.-B.; AURDAL, L. Automatic annotation of subsea pipelines using deep learning. *Sensors*, MDPI, v. 20, n. 3, p. 674, 2020. Disponível em: <https://www.mdpi.com/1424-8220/20/3/674>. Citado na página 19.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016. Disponível em: <http://www.deeplearningbook.org>. Disponível em: <http://www.deeplearningbook.org>. Citado na página 19.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 779–788. Citado na página 19.

VASWANI, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2017. p. 5998–6008. Citado na página 19.

VINYALS, O. et al. Show and tell: A neural image caption generator. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. p. 3156–3164. Citado na página 20.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998. Citado na página 20.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information*

Processing Systems (NeurIPS). [S.l.: s.n.], 2012. p. 1097–1105. Citado na página 21.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015. Citado na página 22.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. ed. Springer Science & Business Media, 2009. Disponível em: <https://web.stanford.edu/~hastie/ElemStatLearn/>. Citado na página 22.

QUINLAN, J. R. Induction of decision trees. *Machine learning*, Springer, v. 1, n. 1, p. 81–106, 1986. Citado na página 22.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IEEE, v. 13, n. 1, p. 21–27, 1967. Citado na página 22.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: A review. *ACM computing surveys (CSUR)*, ACM, v. 31, n. 3, p. 264–323, 1999. Citado na página 23.

ZHU, X. *Semi-Supervised Learning Literature Survey*. [S.l.], 2005. Disponível em: <http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf>. Citado na página 23.

KINGMA, D. P. et al. Semi-supervised learning with deep generative models. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2014. p. 3581–3589. Citado na página 23.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. [S.l.: s.n.], 2019. p. 4171–4186. Citado na página 23.

RADFORD, A. et al. Improving language understanding by generative pre-training. In: . [S.l.: s.n.], 2018. Disponível em: <https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf>. Citado na página 23.

CHEN, T. et al. A simple framework for contrastive learning of visual representations. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2020. p. 1597–1607. Citado na página 23.

GRILL, J.-B. et al. Bootstrap your own latent: A new approach to self-supervised learning. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2020. p. 21271–21284. Citado na página 23.

HU, Y. et al. What can knowledge bring to machine learning? – a survey of low-shot learning for structured data. *arXiv preprint arXiv:2106.06410*, 2021. Disponível em: <https://arxiv.org/abs/2106.06410>. Citado na página 24.

VINYALS, O. et al. Matching networks for one shot learning. In: *Advances in Neural Information Processing Systems*. [s.n.], 2016. v. 29. Disponível em: <https://papers.nips.cc/paper/2016/hash/90e1357833654983612fb05e3ec9148c-Abstract.html>. Citado na página 24.

LAKE, B. M.; SALAKHUTDINOV, R.; TENENBAUM, J. B. One-shot learning by inverting a compositional causal process. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2013. Um dos primeiros trabalhos clássicos em one-shot learning com visão composicional. Citado na página 24.

WANG, Y. et al. Generalizing from a few examples: A survey on few-shot learning. *arXiv preprint arXiv:1904.05046*, 2019. Survey comprehensivo que define o problema FSL e categoriza métodos baseados em conhecimento prévio. Citado na página 25.

SOCHER, R. et al. Zero-shot learning through cross-modal transfer. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2013. Pioneiro no uso de representação textual para reconhecer classes nunca vistas. Citado na página 25.

KOCH, G.; ZEMEL, R.; SALAKHUTDINOV, R. Siamese neural networks for one-shot image recognition. In: *Proceedings of the Deep Learning Workshop at ICML 2015*. International Conference on Machine Learning: [s.n.], 2015. Workshop paper, Deep Learning Workshop. Disponível em: <https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf>. Citado 4 vezes nas páginas 26, 28, 35 e 37.

BROMLEY, J. et al. Signature verification using a "siamese" time delay neural network. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 1993. v. 6, p. 737–744. Citado na página 27.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016. Disponível em: <https://www.deeplearningbook.org/>. Citado na página 27.

AFONJA, T. Accuracy paradox. Medium, 2017. Disponível em: <https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b>. Citado na página 29.

DIGEST, A. S. Stat digest: The idea behind accuracy paradox. Medium, 2023. Disponível em: <https://digestize.medium.com/stat-digest-the-idea-behind-accuracy-paradox-e79daa9fd917>. Citado na página 29.