INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Bahia

Campus
Vitória da Conquista

COEEL
Coordenação de
Engenharia Elétrica

**COORDENAÇÃO DE ENGENHARIA ELÉTRICA - COEEL**

**PROJETO FINAL DE CURSO - PFC**

# Detecção E Classificação De Níveis De Cavitação Em Sistemas Industriais: Uma Abordagem De Clusterização E Comparação De Diferentes Arquiteturas De Redes Neurais Convolucionais

**JONAS SOUZA PINTO**

Vitória da Conquista-BA

5 de março de 2025

**JONAS SOUZA PINTO**

# Detecção E Classificação De Níveis De Cavitação Em Sistemas Industriais: Uma Abordagem De Clusterização E Comparação De Diferentes Arquiteturas De Redes Neurais Convolucionais

Projeto Final de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Bahia, *campus* Vitória da Conquista, como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica.

**Orientador:** Allan de Sousa Soares

**Coorientador:** Carlos Moreno dos Santos

Vitória da Conquista-BA

5 de março de 2025

FICHA CATALOGRÁFICA ELABORADA PELO SISTEMA DE BIBLIOTECAS DO IFBA, COM OS DADOS FORNECIDOS PELO(A) AUTOR(A)

P659 Pinto, Jonas Souza

Detecção e Classificação de Níveis de Cavitação em Sistemas Industriais: Uma Abordagem de Clusterização e Comparação de Diferentes Arquiteturas de Redes Neurais Convolucionais: / Jonas Souza Pinto; orientador Allan de Sousa Soares; coorientador Carlos Moreno dos Santos -- Vitória da Conquista : IFBA, 2025.

108 p.

Trabalho de Conclusão de Curso (Engenharia Elétrica) -- Instituto Federal da Bahia, 2025.

1. Cavitação. 2. Clusterização. 3. Redes Neurais Convolucionais. 4. Monitoramento Preditivo. 5. Aprendizado de Máquina. I. de Sousa Soares, Allan, orient. II. Moreno dos Santos, Carlos, coorient. III. TÍTULO.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA BAHIA
Av. Sérgio Vieira de Mello, 3150 - Bairro Zabelê - CEP 45078-900 - Vitória da Conquista - BA - www.portal.ifba.edu.br

# MENSAGEM

## FOLHA DE APROVAÇÃO PFC

DETECÇÃO E CLASSIFICAÇÃO DE NÍVEIS DE CAVITAÇÃO EM SISTEMAS INDUSTRIAIS: UMA ABORDAGEM DE CLUSTERIZAÇÃO E COMPARAÇÃO DE DIFERENTES ARQUITETURAS DE REDES NEURAIS CONVOLUCIONAIS

JONAS SOUZA PINTO

A presente Monografia de Projeto Final de Curso (doc. SEI 4014918), apresentada em sessão realizada em **dezenove de fevereiro de 2025**, foi avaliada como adequada para a obtenção do Grau de Engenheiro Eletricista, julgada **APROVADA** em sua forma final pela Coordenação do Curso de Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Bahia, Campus Vitoria da Conquista.

BANCA EXAMINADORA

Prof. Me. Allan de Sousa Soares (Orientador)
IFBA campus Vitória da Conquista

Prof. Me. Carlos Moreno dos Santos (Coorientador)
IFBA campus Vitória da Conquista

Prof. Dr. Juan Lieber Marin
IFBA campus Vitória da Conquista

Prof. Dr. Wilton Lacerda Silva
IFBA campus Vitória da Conquista

Vitória da Conquista - Bahia

Documento assinado eletronicamente por **ALLAN DE SOUSA SOARES**, **Professor(a) (Nível Superior)** , em 21/02/2025, às 16:58, conforme decreto nº 8.539/2015.

Documento assinado eletronicamente por **WILTON LACERDA SILVA**, **Professor Efetivo**, em 24/02/2025, às 23:08, conforme decreto nº 8.539/2015.

Documento assinado eletronicamente por **JUAN LIEBER MARIN**, **Professor(a) do Ensino Básico, Técnico e Tecnológico - EBTT**, em 25/02/2025, às 09:22, conforme decreto nº 8.539/2015.

Documento assinado eletronicamente por **CARLOS MORENO DOS SANTOS**, **Coordenador(a) do Curso de Eletromecânica**, em 26/02/2025, às 08:00, conforme decreto nº 8.539/2015.

A autenticidade do documento pode ser conferida no site http://sei.ifba.edu.br/sei/controlador_externo.php?acao=documento_conferir&acao_origem=documento_conferir&id_orgao_acesso_externo=0 informando o código verificador **4014884** e o código CRC **88F1D0BC**.

*A minha família, fonte inesgotável de amor, apoio e inspiração, que sempre acreditou no meu potencial e me incentivou a perseguir meus sonhos. Agradeço também a todas as pessoas que, direta ou indiretamente, contribuíram para o meu crescimento e conhecimento durante esta jornada acadêmica. Este trabalho é dedicado a vocês, com carinho e gratidão …*

*Existe uma teoria que diz que, se um dia alguém descobrir exatamente para que serve o Universo e por que ele está aqui, ele desaparecerá instantaneamente e será substituído por algo ainda mais estranho e inexplicável. Existe uma segunda teoria que diz que isso já aconteceu. [Douglas Adams]*

# AGRADECIMENTOS

Agradeço ao Instituto Federal da Bahia (IFBA) por proporcionar uma educação pública e de qualidade, além de oferecer experiências enriquecedoras por meio de projetos de extensão, ensino e pesquisa. Sou imensamente grato a cada professor, especialmente aos das disciplinas da ênfase em Eletrônica, que despertaram em mim o interesse e o entusiasmo pela área de Sistemas Digitais.

Expresso minha profunda gratidão às professoras Maria das Graças Bittencourt e Joseane Oliveira, que foram fundamentais ao me incentivarem a aplicar o conhecimento em projetos que impactassem positivamente a comunidade. Agradeço também aos colegas do INQ.IFBA, GITI, DAEE e do PET Engenharias pelo aprendizado compartilhado e pela convivência ao longo da graduação.

À minha família, que sempre acreditou no meu potencial e me apoiou incondicionalmente, deixo meu eterno agradecimento. Ao meu melhor amigo Filipe Leto, que sempre demonstrou interesse pelas minhas ideias e contribuiu para desenvolvê-las, deixo minha gratidão especial.

Aos colegas Jonas Machado, Heitor, Estefani, Emanuelle e Kiara, companheiros nas longas madrugadas de estudo, agradeço pela parceria e pela troca de conhecimentos. À minha namorada Ana Beatriz, obrigado por todo o suporte, paciência e compreensão durante a realização deste projeto.

Por fim, registro minha gratidão aos professores orientadores Carlos e Allan pelo suporte e orientação ao longo do desenvolvimento deste trabalho, bem como ao estudante do curso técnico em Eletromecânica, Eduardo Costa, por sua importante contribuição no controle do sistema de bombeamento durante a aquisição de imagens.

A todos vocês, meu sincero e profundo agradecimento.

# RESUMO

Este trabalho propõe uma abordagem para detecção e classificação de níveis de cavitação em sistemas industriais, utilizando técnicas de clusterização e comparação de diferentes arquiteturas de CNNs. A cavitação, um fenômeno que causa danos mecânicos em sistemas de bombeamento, foi analisada através de imagens capturadas em um sistema experimental. A metodologia envolveu a construção de um *dataset* de imagens, onde um sistema de aquisição foi configurado para capturar a formação de bolhas de cavitação em um tubo de acrílico transparente. A iluminação foi controlada para garantir a qualidade das imagens, e um transdutor de pressão foi utilizado para monitorar os níveis de pressão de sucção. Foram exploradas arquiteturas como *VGG16*, *VGG19*, *ResNet50* e *ResNet101*, com a aplicação de filtros como *Prewitt* e *Sobel* para melhorar a detecção de características. O processo de clusterização utilizou *Gaussian Mixture Models* (GMM) para agrupar os dados, e a redução de dimensionalidade foi realizada através de PCA. Os resultados mostraram que a arquitetura *VGG19* alcançou uma acurácia de 88% na classificação supervisionada, enquanto a *ResNet101* obteve 91%. Além disso, as arquiteturas *VGG* consumiram menos memória RAM em comparação com as *ResNet*, especialmente em grandes volumes de dados. A análise de desempenho também revelou que a aplicação de filtros melhorou a qualidade dos clusters, facilitando a separação dos níveis de cavitação. Em dados de teste diversificados, com variações de iluminação e capturados em diferentes dias, o modelo com a *VGG19* alcançou uma precisão média de 79,97%. Este estudo destaca a eficácia das técnicas de visão computacional e aprendizado de máquina no monitoramento preditivo de cavitação, contribuindo para a manutenção eficiente e a redução de custos em ambientes industriais.

**Palavras-chave:** Cavitação, Clusterização, Redes Neurais Convolucionais, Monitoramento Preditivo, Aprendizado de Máquina

# ABSTRACT

This work proposes an approach for the detection and classification of cavitation levels in industrial systems using clustering techniques and a comparison of different convolutional neural network (CNN) architectures. Cavitation, a phenomenon that causes mechanical damage in pumping systems, was analyzed through images captured in an experimental setup. The methodology involved constructing an image dataset, where an acquisition system was configured to capture cavitation bubble formation in a transparent acrylic tube. Lighting was controlled to ensure image quality, and a pressure transducer was used to monitor suction pressure levels. Architectures such as VGG16, VGG19, ResNet50, and ResNet101 were explored, with the application of filters like Prewitt and Sobel to enhance feature detection. The clustering process used GMM to group the data, and dimensionality reduction was performed using PCA. The results showed that the VGG19 architecture achieved 88% accuracy in supervised classification, while ResNet101 obtained 91%. Additionally, the VGG architectures consumed less RAM compared to the ResNet architectures, especially with large volumes of data. Performance analysis also revealed that the application of filters improved cluster quality, facilitating the separation of cavitation levels. On diversified test data, with variations in lighting and captured on different days, the model with VGG19 achieved an average precision of 79.97%. This study highlights the effectiveness of computer vision and machine learning techniques in predictive cavitation monitoring, contributing to efficient maintenance and cost reduction in industrial environments.

**Keywords:** Cavitation, Clustering, Convolutional Neural Networks, Predictive Monitoring, Machine Learning

# Lista de Figuras

# Lista de Tabelas

# Lista de Códigos

# Glossário: Símbolos e Siglas

| Notação | Descrição | Páginas |
|---|---|---|
| $R$ | Raio crítico da bolha ($\mathrm{m}$) | 7 |
| $S$ | Tensão superficial da interface líquido-vapor ($\mathrm{N/m}$) | 7, 25 |
| $W$ | Energia líquida necessária para formar ou colapsar a bolha ($\mathrm{J}$) | 7 |
| $P$ | Pressão do fluido ($\mathrm{Pa}$) | 5 |
| $\rho$ | Densidade do fluido ($\mathrm{kg/m^3}$) | 5 |
| $g$ | Aceleração da gravidade ($\mathrm{m/s^2}$) | 5 |
| $h$ | Altura do fluido (m) | 5, 6 |
| $v$ | Velocidade do fluido ($\mathrm{m/s}$) | 5, 6 |
| *ADAM* | Algoritmo de otimização adaptativo. | xxi, 29 |
| $a(i)$ | Distância média entre $i$ e todos os outros pontos do mesmo *cluster* (coesão) | 24 |
| *API* | Interface de Programação de Aplicações | 28 |
| $b(i)$ | Menor distância média entre $i$ e os pontos dos *clusters* vizinhos (separação) | 24 |
| $b_k$ | Bias do neurônio $k$ | 14 |
| $c_1$ | Primeira componente principal, direção de máxima variância dos dados | 25 |

| Notação | Descrição | Páginas |
|---|---|---|
| $c_2$ | Segunda componente principal, ortogonal a $c_1$ e com segunda maior variância | 25 |
| CNN | *Convolutional Neural Networks* (Redes Neurais Convolucionais) | viii, ix, 1, 14, 15, 41, 74 |
| COEEL | *Coordenação do Curso de Engenharia Elétrica do IFBA campus Vitória da Conquista* | i |
| $d$ | Número de características | 25 |
| $D_{sum}(x, y)$ | Soma das diferenças na posição do pixel | 11 |
| $E$ | Erro em uma rede neural | 14 |
| $\epsilon$ | Termo de estabilidade numérica | 29 |
| $\eta$ | Taxa de aprendizado | 29 |
| $e^{z_i}$ | Exponencial da ativação | 16 |
| *FPS* | Frames por segundo | 41 |
| GMM | *Gaussian Mixture Models* (Modelos de Mistura Gaussiana) | viii, ix, 22, 23, 41, 74 |
| *GND* | Terra ou referência de tensão zero | 35 |
| $G_x$ | Derivada aproximada da intensidade da imagem na direção horizontal | 19, 20 |
| $G_y$ | Derivada aproximada da intensidade da imagem na direção vertical | 19, 20 |
| $I_t(x, y)$ | Valor da intensidade do pixel na posição $(x, y)$ no frame atual | 12 |
| $\hat{I}_{t-1}(x, y)$ | Valor da intensidade do pixel na posição $(x, y)$ no frame anterior | 12 |

| Notação | Descrição | Páginas |
|---|---|---|
| $\hat{I}_{t-2}(x, y)$ | Valor da intensidade do pixel na posição $(x, y)$ no segundo frame anterior | 12 |
| $\lambda_i$ | Autovalores (variância explicada) | 25 |
| *LED* | Diodo emissor de luz | 33 |
| ML | *Machine Learning* (Aprendizado de Máquina) | 1, 2 |
| *mmHg* | Milímetros de mercúrio, unidade de pressão | 36 |
| $\boldsymbol{\mu}_k$ | Vetor de médias da $k$-ésima componente | 23 |
| $n$ | Número de amostras | 25 |
| PCA | *Principal Component Analysis* (Análise de Componentes Principais) | viii, ix, xi, 24, 41, 52–57, 59–64 |
| $\pi_k$ | Peso da $k$-ésima componente | 23 |
| *psi* | Libras por polegada quadrada, unidade de pressão | 35, 36 |
| *ReLU* | Função de ativação *Rectified Linear Unit* | 15 |
| $\hat{s}_t$ | Variância corrigida para viés | 29 |
| $s(i)$ | Coeficiente de silhueta do ponto $i$, variando entre -1 e 1 | 24 |
| $\sigma(\vec{z})_i$ | Probabilidade da classe $i$, calculada a partir da ativação $z_i$ | 17 |
| $\boldsymbol{\Sigma}_k$ | Matriz de covariância da $k$-ésima componente | 23 |

| Notação | Descrição | Páginas |
|---|---|---|
| $\sum_{j=1}^{K} e^{z_j}$ | Soma das exponenciais das ativações de todas as classes | 16 |
| $t$ | Índice do tempo atual | 12 |
| *thread* | Fluxo de execução dentro de um processo ou programa | 45 |
| $\mathbf{u}_i$ | Autovetores (direções principais) | 25 |
| $\hat{v}_t$ | Gradiente corrigido para viés | 29 |
| *Vcc* | Tensão de alimentação positiva | 35 |
| $w_{ki}$ | Pesos sinápticos entre neurônios | 13, 14 |
| $x$ | Valor numérico de entrada para a função | 16 |
| $x_1$ | Primeira dimensão do conjunto de dados bidimensional | 25 |
| $x_2$ | Segunda dimensão do conjunto de dados bidimensional | 25 |
| $\mathbf{x}_t$ | Parâmetros atuais no passo $t$ | 29 |
| $x, y$ | Coordenadas horizontal e vertical do pixel na imagem | 12 |
| $\vec{z}$ | Vetor de entrada | 16 |

# Sumário

# Capítulo 1

# Introdução

## 1.1 Introdução do Trabalho

A cavitação é um fenômeno físico que ocorre em sistemas de bombeamento, resultando na formação e colapso de bolhas de vapor devido a condições de baixa pressão. Esse processo causa danos mecânicos significativos, como erosão, vibrações e perda de eficiência, representando um grande desafio para a manutenção de equipamentos industriais. (BRENNEN, 2011) Esses impactos elevam os custos operacionais e comprometem a confiabilidade dos sistemas.

Com o avanço das tecnologias de aprendizado de máquina (ML) e visão computacional, torna-se possível desenvolver instrumentos inovadores para a detecção precoce da cavitação. Por meio da análise de imagens e do uso de redes neurais convolucionais (CNNs), é viável identificar padrões complexos associados ao fenômeno de forma precisa e eficiente. Essas abordagens não apenas melhoram a compreensão do problema, mas também permitem intervenções preventivas, reduzindo falhas operacionais e otimizando os sistemas industriais (GAISSER; KIRSCHNER; RIEDELBAUCH, 2023).

Este trabalho aborda a relevância de utilizar técnicas de visão computacional para enfrentar os desafios impostos pela cavitação, destacando o potencial de reduzir custos e aumentar a confiabilidade em ambientes industriais. Os detalhes do desenvolvimento e dos resultados obtidos serão discutidos nos capítulos subsequentes.

## 1.2 Objetivo Geral

Desenvolver um sistema eficiente baseado em visão computacional para a detecção e classificação de cavitação em sistemas de bombeamento, utilizando técnicas de ML, como , e algoritmos de clusterização. O objetivo principal é proporcionar um método preciso e robusto para o monitoramento em tempo real, contribuindo para a manutenção preditiva, a redução de custos operacionais e o aumento da confiabilidade de equipamentos industriais.

## 1.3 Objetivos Específicos

- Desenvolver métodos de processamento de imagens para destacar características relevantes associadas à cavitação.
- Explorar e comparar diferentes abordagens de ML para a detecção e classificação de padrões associados à cavitação.
- Implementar técnicas de análise e agrupamento de dados para segmentar níveis de gravidade da cavitação, facilitando sua interpretação.
- Criar uma ferramenta interativa para monitoramento em tempo real, integrando os resultados obtidos pelo modelo proposto.

## 1.4 Justificativa

A cavitação em sistemas de bombeamento é um problema crítico para a indústria, causando danos como erosão, deformação plástica e trincas em componentes metálicos. A análise de (LIU et al., 2023) sobre os efeitos do colapso de bolhas de cavitação em superfícies de aço inoxidável destaca a geração de altas pressões e temperaturas localizadas, que resultam em alterações químicas e estruturais no material. Além disso, a presença de íons metálicos em ambientes aquosos intensifica esses danos, evidenciando a necessidade de métodos avançados para monitorar e mitigar os impactos da cavitação.

A detecção precisa e a análise das características da cavitação são essenciais para o monitoramento e manutenção preditiva. O trabalho de (JU; CHOI, 2022) empregou técnicas de processamento de imagem para identificar padrões de erosão, mostrando o potencial da visão computacional para análise detalhada do fenô-

meno. Essas técnicas podem ser adaptadas para sistemas de bombeamento, possibilitando a identificação precoce de áreas críticas e permitindo intervenções preventivas mais eficazes.

Por fim, o impacto econômico da cavitação reforça a importância do desenvolvimento de soluções inovadoras. Segundo (JU; CHOI, 2022), as taxas de erosão aumentam significativamente com a intensidade do fenômeno, gerando custos elevados com reparos e substituições. A implementação de sistemas baseados em aprendizado de máquina e visão computacional pode reduzir esses custos ao proporcionar um monitoramento contínuo e intervenções preventivas mais eficazes.

# Capítulo 2

# Princípios da Cavitação em Sistemas de Bombeamento

## 2.1 Definição e Fenômeno de Cavitação

A cavitação é um fenômeno físico complexo e multifacetado, caracterizado pela formação e colapso de bolhas de vapor em um líquido devido à redução da pressão local abaixo da pressão de vapor do líquido. Segundo (BRUNETTI, 2008), a pressão de vapor é o ponto crítico no qual o líquido e o vapor coexistem em equilíbrio, sendo dependente não apenas da temperatura, mas também da pressão ambiente e das características moleculares do fluido.

Este fenômeno ocorre frequentemente em sistemas de bombeamento, especialmente em áreas onde há aceleração significativa do fluido, como nas pás de bombas centrífugas, válvulas de controle, e outros componentes hidráulicos. Nestas condições, a redução de pressão em regiões de alta velocidade pode levar à formação de bolhas de vapor, que posteriormente colapsam violentamente. Conforme (BRENNEN, 2011), a formação de cavitação em turbomáquinas ocorre quando a pressão estática cai abaixo da pressão de vapor local, um processo que pode ser intensificado por fatores como temperatura elevada do fluido, presença de gases dissolvidos e rugosidade das superfícies.

A equação de Bernoulli, um princípio fundamental da mecânica dos fluidos, desempenha um papel central no entendimento da cavitação, pois descreve com precisão a relação entre a energia mecânica total de um fluido em movimento e sua pressão, velocidade e altura. Segundo (MUNSON; YOUNG; OKIISHI, 2004), o

fenômeno é matematicamente expressado pela Equação 2.1

$$P + \frac{1}{2}\rho v^2 + \rho g h = \text{constante}, \tag{2.1}$$

Sendo que:

$P \implies$ Pressão do fluido ($\mathrm{Pa}$);

$\rho \implies$ Densidade do fluido ($\mathrm{kg/m^3}$);

$v \implies$ Velocidade do fluido ($\mathrm{m/s}$);

$g \implies$ Aceleração da gravidade ($\mathrm{m/s^2}$);

$h \implies$ Altura do fluido (m);

Conforme a equação de Bernoulli, em um sistema onde ocorre um aumento significativo da velocidade do fluido a pressão estática $P$ diminui para conservar a energia total do sistema. Esta relação fundamental explica por que a cavitação frequentemente ocorre em regiões de alta velocidade do fluido, como nas entradas de bombas e nas áreas de constrição do fluxo. Se essa pressão cair abaixo da pressão de vapor do fluido, inicia-se a formação de bolhas de cavitação (BRUNETTI, 2008). Esse processo ocorre porque o fluido local não consegue mais sustentar sua fase líquida, formando bolhas de vapor que podem crescer rapidamente dependendo das condições locais do escoamento.

A aplicação prática dessa equação em um sistema de bombeamento pode ser descrita entre dois pontos de um escoamento (MUNSON; YOUNG; OKIISHI, 2004), conforme a Equação 2.2. Esta formulação é particularmente útil para análise de sistemas reais, onde as condições de operação podem variar significativamente ao longo do percurso do fluido.

$$P_1 + \frac{1}{2}\rho v_1^2 + \rho g h_1 = P_2 + \frac{1}{2}\rho v_2^2 + \rho g h_2, \tag{2.2}$$

Onde:

$P_1 \implies$ Pressão no ponto 1 ($\mathrm{Pa}$);

$P_2 \implies$ Pressão no ponto 2 ($\mathrm{Pa}$);

$v_1 \implies$ Velocidade no ponto 1 ($\mathrm{m/s}$);

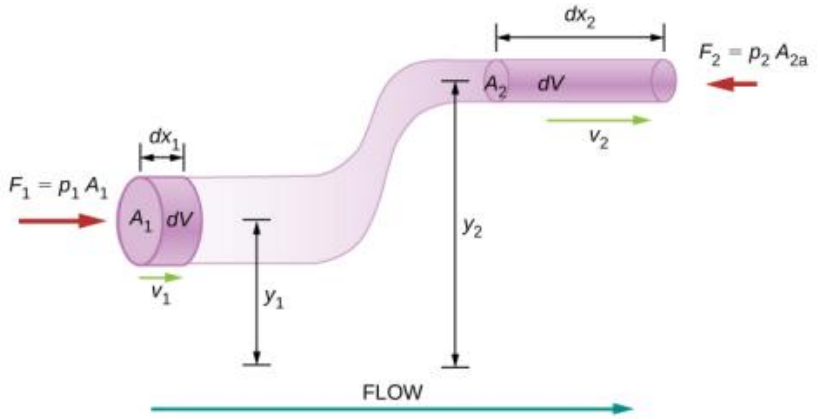
$v_2 \implies$ Velocidade no ponto 2 ($\mathrm{m/s}$);

$h_1 \implies$ Altura no ponto 1 (m);

$h_2 \implies$ Altura no ponto 2 (m).

Segundo (BRENNEN, 2011), se $v_2$ for significativamente maior que $v_1$, a pressão $P_2$ pode cair abaixo do limite necessário para manter o fluido no estado líquido. Este é o ponto crítico onde ocorre a cavitação, um fenômeno que pode ser ainda mais pronunciado em situações onde há variações bruscas de geometria ou em condições de operação fora do ponto ótimo de funcionamento do equipamento. A Figura 2.1 ilustra como o equilíbrio de forças e a continuidade do fluxo influenciam a formação da cavitação em um sistema típico.

**Figura 2.1** – *Esquema combinando o princípio de continuidade e o equilíbrio de forças.*




**Fonte :**(LibreTexts Team, 2024).

## 2.2 Danos Causados pelo Colapso da Cavitação

### 2.2.1 Colapso de Bolhas e Efeitos Mecânicos

O colapso das bolhas de cavitação é um fenômeno destrutivo que ocorre quando as bolhas, formadas em regiões de baixa pressão, são transportadas para áreas de alta pressão e implodem violentamente. Esse processo gera jatos de alta velocidade que podem atingir superfícies com pressões localizadas de até milhares de atmosferas, produzindo ondas de choque microscópicas que se propagam pelo meio fluido. Conforme (BRENNEN, 1995), a energia associada ao processo de formação de uma bolha é composta por dois componentes principais: o trabalho

necessário para deslocar o fluido circundante e a energia armazenada devido à tensão superficial.

Durante o processo de colapso, as bolhas podem atingir velocidades supersônicas, gerando pressões de impacto que podem exceder 1000 MPa em áreas microscópicas. Este fenômeno é particularmente destrutivo devido à natureza concentrada e repetitiva dos impactos, que podem ocorrer milhares de vezes por segundo em uma mesma região. A energia total associada a este processo é representada matematicamente na Equação 2.3.

$$W_{CR} = \frac{4}{3}\pi R_C^2 S \tag{2.3}$$

Onde:

$W \implies$ Energia líquida necessária para formar ou colapsar a bolha ($\mathrm{J}$);

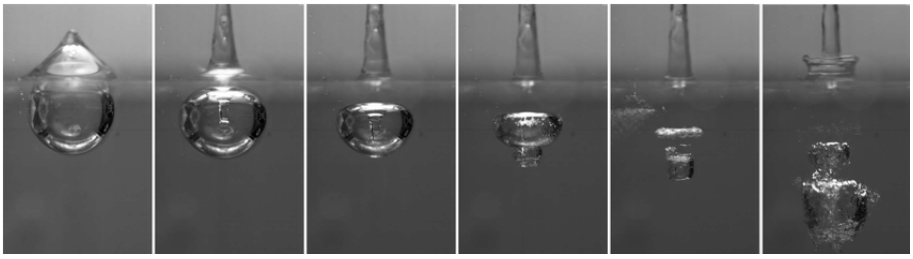
$R \implies$ Raio crítico da bolha ($\mathrm{m}$);

$S \implies$ Tensão superficial da interface líquido-vapor ($\mathrm{N/m}$);

A dinâmica do colapso das bolhas é ainda mais complexa quando ocorre próxima a superfícies sólidas, onde a assimetria do campo de pressão pode resultar na formação de micro-jatos direcionados, intensificando o potencial de dano. Este fenômeno é conhecido como "efeito de proximidade"e pode amplificar significativamente a intensidade do impacto sobre as superfícies metálicas.

A Figura 2.2 demonstra visualmente o processo de colapso e rebote de uma bolha de cavitação, destacando os estágios que resultam em jatos de alta velocidade capazes de causar danos significativos a superfícies sólidas.

**Figura 2.2** – *Sequência do colapso e rebote de uma bolha de cavitação próxima a uma superfície livre.*



**Fonte :**(SUPPONEN et al., 2015).

### 2.2.2 Danos Estruturais e Efeitos de Erosão

A cavitação representa um dos fenômenos mais prejudiciais em turbomáquinas, bombas centrífugas e sistemas hidráulicos, devido à natureza altamente destrutiva dos jatos formados pelo colapso das bolhas ao impactarem superfícies sólidas. Este processo causa uma cascata de efeitos negativos, incluindo desgaste mecânico acelerado, redução significativa da vida útil dos componentes e deterioração progressiva da eficiência operacional do sistema. Segundo (KRELLA, 2023), a repetição cíclica desse fenômeno provoca uma degradação progressiva e sistemática dos materiais, resultando em um aumento exponencial na frequência de manutenções necessárias e, consequentemente, elevando substancialmente os custos operacionais do sistema.

O processo de degradação estrutural ocorre em múltiplas etapas, começando com a formação de microfissuras na superfície do material, que progressivamente se expandem e se interconectam, levando à remoção de partículas microscópicas do material. Este processo é particularmente severo em áreas de alta velocidade do fluido, como nas bordas de ataque das pás de bombas e em regiões de constrição do fluxo.

Conforme (ESCALER et al., 2006), o colapso das bolhas gera ondas de choque e micro-jatos que podem atingir velocidades superiores a 100 m/s, causando erosão severa nas superfícies internas dos equipamentos. A energia liberada durante cada evento de colapso, embora microscópica em escala individual, torna-se significativa quando multiplicada pelos milhares de eventos que ocorrem simultaneamente. As superfícies metálicas expostas a este fenômeno desenvolvem características típicas de fadiga por impacto.

Além dos danos físicos diretos, as vibrações induzidas pela cavitação introduzem um componente adicional de desgaste estrutural. Estas vibrações, que podem atingir frequências na faixa de kHz, propagam-se através da estrutura do equipamento, podendo causar:

- Fadiga mecânica acelerada em componentes estruturais
- Desalinhamento progressivo de eixos e elementos rotativos
- Sobrecarga em mancais e sistemas de vedação
- Ressonância estrutural em casos críticos

## 2.3 Classificação dos Níveis de Cavitação

A classificação dos níveis de cavitação em grupos distintos é uma estratégia importante para a compreensão e análise detalhada do fenômeno. De acordo com (JAIN, 2010), o agrupamento de dados é um método exploratório fundamental para identificar padrões e estruturar informações em conjuntos com características similares.

Esta classificação em níveis específicos se justifica pela necessidade de estabelecer parâmetros objetivos para monitoramento e controle do fenômeno. A categorização permite padronizar a identificação da severidade da cavitação, facilitando tanto o diagnóstico operacional quanto a definição de medidas preventivas. A Tabela 2.1 organiza os níveis de cavitação com base na variação da pressão com as limitações para esse trabalho.

**Tabela 2.1** – *Classificação dos níveis de cavitação.*

| Nível | Intervalo de Pressão ($mmHg$) | Descrição |
|---|---|---|
| Nível 1 | 0 a -300 | Formação inicial de bolhas, sem cavitação significativa. |
| Nível 2 | -300 a -400 | Formação de bolhas mais pronunciada, com possíveis impactos na eficiência. |
| Nível 3 | -400 a -500 | Cavitação moderada, erosão incipiente e queda substancial na eficiência. |
| Nível 4 | -500 a -600 | Cavitação severa, intensa formação de bolhas e danos estruturais significativos. |

**Fonte:** Autoria própria.

O comportamento físico de cada nível está diretamente relacionado à redução da pressão e ao aumento da velocidade do fluido, fatores que devem ser monitorados para evitar danos críticos.

## 2.4 Visão Computacional no Monitoramento de Cavitação

A aplicação de técnicas de visão computacional na análise e detecção da cavitação representa uma importante ferramenta de instrumentação eletrônica no contexto da Indústria 4.0. De acordo com (TONG et al., 2023), a integração de sistemas de fotografia de alta velocidade com análise de sinais de vibração estabelece

um paradigma de monitoramento inteligente, alinhado com os princípios da manufatura avançada. Esta abordagem permite a coleta e processamento de dados em tempo real, contribuindo para a transformação digital dos processos industriais e a implementação de manutenção preditiva.

O monitoramento baseado em visão computacional oferece vantagens significativas sobre métodos convencionais, possibilitando a detecção precoce e não invasiva da formação de bolhas de cavitação. A utilização de câmeras de alta velocidade, capazes de capturar milhares de quadros por segundo, permite observar a dinâmica do fenômeno em detalhes microscópicos, incluindo a formação, crescimento e colapso das bolhas.

# Capítulo 3

# Técnicas de Processamento Digital e Aprendizado Profundo

## 3.1 Extração e Processamento de Frames

A extração e processamento de *frames* em sistemas de detecção requer uma abordagem sistemática e robusta para garantir a qualidade dos dados analisados. Conforme (ELLENFELD et al., 2021), o processo de segmentação de movimento através de diferenciação de *frames* necessita considerar tanto aspectos temporais quanto espaciais do fenômeno observado. A extração de *frames* individuais deve ser realizada considerando a taxa de amostragem adequada para capturar as características dinâmicas do fenômeno estudado.

O processamento dos *frames* extraídos envolve múltiplas etapas de análise e filtragem. De acordo com (ELLENFELD et al., 2021), a subtração de *background* e algoritmos de detecção são fundamentais para isolar os elementos relevantes em cada *frame*. Este processo pode ser realizado através da Equação 3.1 de diferenciação entre *frames*:

$$D_{sum}(x,y) = |I_t(x,y) - \hat{I}_{t-1}(x,y)| + |I_t(x,y) - \hat{I}_{t-2}(x,y)| \tag{3.1}$$

Onde:

$D_{sum}(x,y) \implies$ Soma das diferenças na posição do pixel $(x,y)$;

$I_t(x, y) \implies$ Valor da intensidade do pixel na posição $(x, y)$ no frame atual;

$\hat{I}_{t-1}(x, y) \implies$ Valor da intensidade do pixel na posição $(x, y)$ no frame anterior;

$\hat{I}_{t-2}(x, y) \implies$ Valor da intensidade do pixel na posição $(x, y)$ no segundo frame anterior;

$x, y \implies$ Coordenadas horizontal e vertical do pixel na imagem;

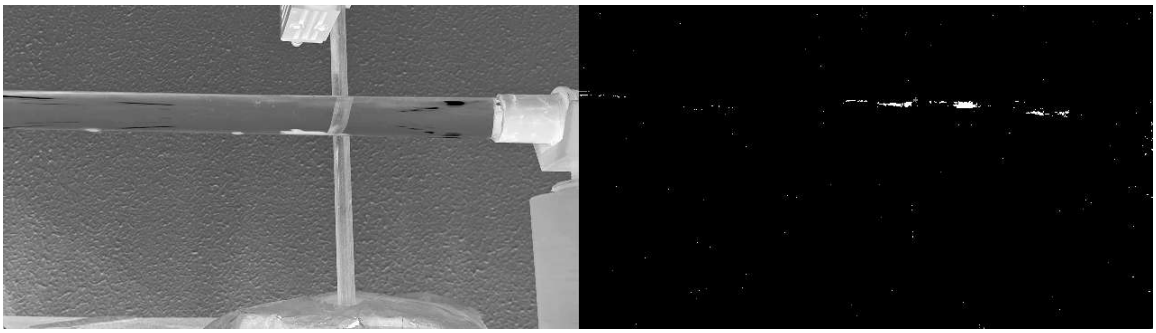
$t \implies$ Índice do tempo atual;

$t - 1 \implies$ Índice do tempo anterior (um frame antes);

$t - 2 \implies$ Índice do segundo tempo anterior (dois frames antes).

Como pode ser observado na Figura 3.1, o processo de diferenciação entre frames permite identificar regiões onde ocorrem mudanças significativas entre frames consecutivos, destacando áreas de potencial movimento. A figura apresenta o frame original em escala de cinza mostrando o tubo de cavitação, e o resultado do processamento após aplicação da diferenciação entre frames, onde são destacadas as regiões onde ocorrem mudanças significativas.

**Figura 3.1** – *Diferenciação entre frames*



**Fonte:** Autoria Própria.

A análise *frame-a-frame* requer considerações específicas sobre o posicionamento e ângulo da câmera (LUFF; HEATH, 2012), pois a qualidade da extração depende significativamente da configuração inicial do sistema de captura. O processamento deve considerar aspectos como iluminação, contraste e resolução espacial, que podem afetar a identificação precisa das bolhas de cavitação.

A integração dos dados processados com sistemas de análise em tempo real demanda uma arquitetura robusta de processamento. (MANJU; VALARMATHIE,

2021) argumenta que a implementação de *frameworks* de análise semântica, incluindo o *OpenCV* (*Open Source Computer Vision Library*) - uma biblioteca de código aberto para processamento de imagens e visão computacional - permite uma extração mais precisa das características relevantes dos *frames*. O *OpenCV* oferece funções otimizadas para processamento de vídeo em tempo real e detecção de objetos (OpenCV Team, 2024).

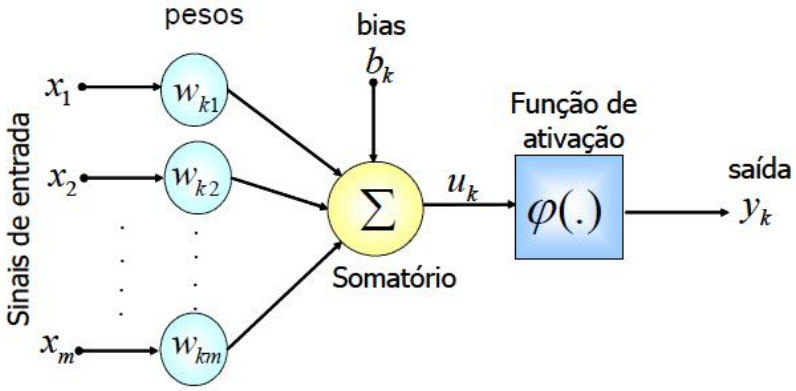
## 3.2 Neurônios Artificiais

### 3.2.1 Ajustes de Pesos e Bias

Conforme conceituado por (HAYKIN, 2009), redes neurais artificiais são sistemas computacionais inspirados no funcionamento do cérebro humano, onde unidades de processamento interconectadas, análogas aos neurônios biológicos, trabalham em conjunto para processar informações. Essas redes aprendem por meio da exposição a exemplos, permitindo que sua estrutura interna seja adaptada para melhorar o desempenho em tarefas específicas (AGGARWAL, 2018).

O neurônio artificial, unidade fundamental das redes neurais, é uma abstração matemática que simula o comportamento básico dos neurônios biológicos (HAYKIN, 2009) . Como ilustrado na Figura 3.2, cada neurônio recebe múltiplos sinais de entrada, processa essas informações por meio de uma função de ativação e produz uma saída. Essa funcionalidade básica é a base para a construção de arquiteturas mais complexas, como as redes multicamadas.

**Figura 3.2** – *Modelo de um Neurônio Artificial*




**Fonte:** (SOARES; SILVA, 2011).

Os pesos sinápticos $w_{ki}$ representam a força das conexões entre os neurônios

em uma rede neural. Conforme descrito por (GURNEY, 1997), Durante o aprendizado os pesos são ajustados iterativamente para permitir que a rede neural aprenda padrões complexos nos dados de entrada e adapte seu comportamento para minimizar o erro de predição.

De acordo com (AGGARWAL, 2018), o bias pode ser visto como um fator que facilita a modelagem de padrões que não cruzam a origem do espaço de entrada, adicionando um valor constante à soma ponderada dos sinais de entrada antes de aplicar a função de ativação.

### 3.2.2 Propagação do Sinal

O processo de propagação do sinal ocorre em duas etapas principais: a propagação direta (*feedforward*) e a retropropagação do erro (*backpropagation*). Na propagação direta, os sinais de entrada são processados pelos neurônios, camada por camada, até gerar uma saída final (AGGARWAL, 2018). Já na retropropagação, o erro $E$ é propagado no sentido inverso, ajustando os pesos $w_{ki}$ e bias $b_k$ para minimizar o erro em futuras iterações (HAYKIN, 2009).

Durante esse processo, cada neurônio artificial atua como um processador elementar que recebe, transforma e transmite sinais, permitindo que a rede como um todo desenvolva representações cada vez mais refinadas e abstratas das informações de entrada. A combinação da propagação direta com a retropropagação do erro cria um sistema robusto de aprendizado que se adapta continuamente.

## 3.3 Redes Neurais Convolucionais e Arquiteturas Pré-Treinadas

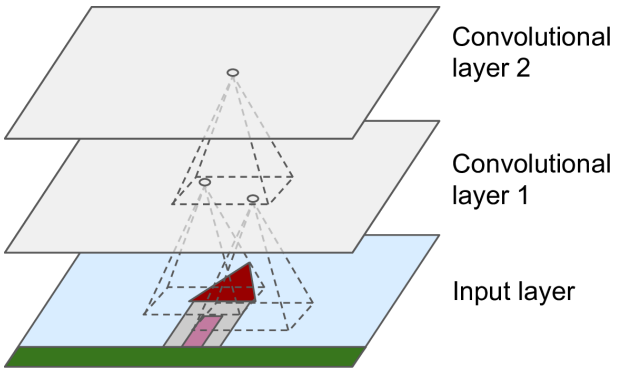
### 3.3.1 Arquitetura Geral das Redes Neurais Convolucionais

A arquitetura das Redes Neurais Convolucionais (CNNs) é estruturada em uma sequência hierárquica de camadas, onde cada uma desempenha um papel específico no processamento dos dados (AGGARWAL, 2018). Essa organização possibilita que as *CNNs* sejam aplicadas de forma eficiente a problemas envolvendo da-

dos espaciais, como imagens e vídeos, devido à sua capacidade de capturar tanto padrões locais quanto globais.

Conforme ilustrado na Figura 3.3, cada neurônio em uma camada convolucional conecta-se apenas a uma região específica da camada anterior, denominada *campo receptivo*, de modo que características visuais cada vez mais complexas sejam extraídas em estágios sucessivos.

**Figura 3.3** – *Estrutura das camadas de uma CNN.*



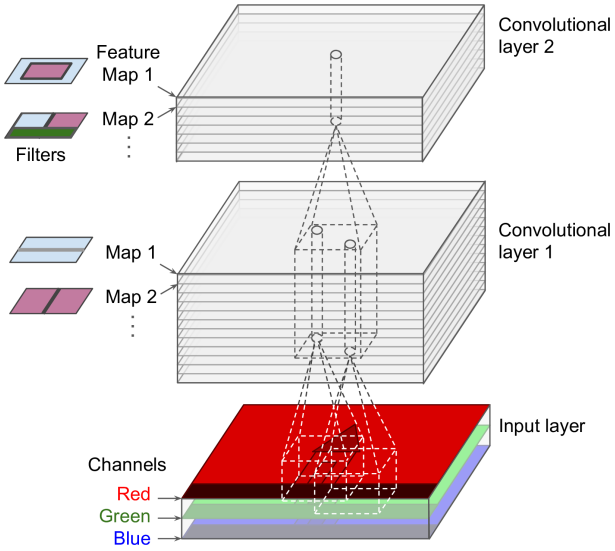

**Fonte:** (GéRON, 2017).

O processamento tem início nas camadas convolucionais, que aplicam filtros (*kernels*) diretamente à entrada (CHOLLET, 2017). Como mostra a Figura 3.4, diferentes filtros são aplicados à imagem de entrada para gerar *mapas de características* (*feature maps*) que enfatizam padrões específicos, como bordas e texturas. Cada filtro é responsável por detectar um tipo de característica local, produzindo um mapa correspondente que conserva informações espaciais relevantes.

Entre os componentes fundamentais das CNNs, destaca-se a função de ativação *ReLU* (*Rectified Linear Unit*), que conforme conceitua (GéRON, 2017), a função de ativação não-linear é responsável por introduzir a capacidade de aprendizado de padrões complexos na rede. Sua natureza não-saturante para valores positivos previne o problema do desvanecimento do gradiente durante o treinamento.

Segundo (CHOLLET, 2017), a *ReLU* apresenta melhor convergência em redes profundas quando comparada a funções sigmoides tradicionais, permitindo um treinamento mais rápido e eficaz de arquiteturas complexas. Além disso, sua implementação introduz propriedades importantes como a esparsidade de ativação e a resistência ao desvanecimento do gradiente, problemas comuns em arquiteturas anteriores. A função *ReLU* pode ser matematicamente expressa pela Equação 3.2:

**Figura 3.4** – *Aplicação de filtros em imagens RGB e geração de mapas de características.*




**Fonte:** (GéRON, 2017).

$$\text{ReLU}(x) = \max(0, x), \tag{3.2}$$

Onde:

$x \implies$ Valor numérico de entrada para a função.

Em cenários de classificação multiclasse, a função *softmax* (Equação 3.3) é frequentemente aplicada à saída da rede para produzir uma distribuição de probabilidade sobre as classes. Essa função normaliza as saídas da última camada, transformando os *scores* brutos em probabilidades que somam 1(AGGARWAL, 2018).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.3}$$

Onde:

$\vec{z} \implies$ Vetor de entrada, cujos componentes $z_i$ representam a ativação da rede para cada classe $i$;

$e^{z_i} \implies$ Exponencial da ativação;

$\sum_{j=1}^{K} e^{z_j} \implies$ Soma das exponenciais das ativações de todas as classes;

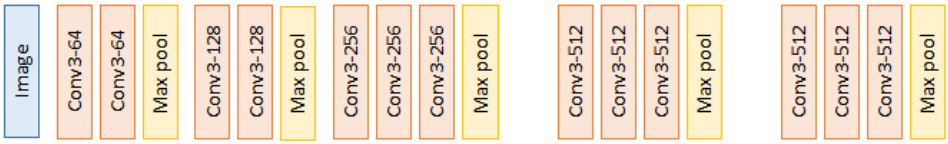
$\sigma(\vec{z})_i \implies$ Probabilidade da classe $i$, calculada a partir da ativação $z_i$.

## 3.3.2 Arquiteturas Pré-Treinadas

### 3.3.2.1 Arquitetura *VGG* (*Visual Geometry Group)*

Conforme descrito por (SIMONYAN; ZISSERMAN, 2015), o principal diferencial da arquitetura *VGG (*Visual Geometry Group) é o uso exclusivo de filtros convolucionais muito pequenos, de tamanho $3 \times 3$, com *stride* 1, e camadas de *max-pooling* de tamanho $2 \times 2$. Essa abordagem simplificada permitiu explorar a profundidade das redes como fator importante para o desempenho em tarefas de classificação e localização de imagens. A Figura 3.5 ilustra a estrutura geral da arquitetura VGG16, destacando suas camadas convolucionais e totalmente conectadas.

**Figura 3.5** – *Estrutura da arquitetura VGG16.*

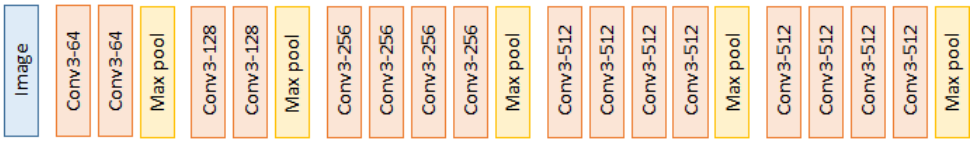



**Fonte:** (SHADEED; TAWFEEQ; MAHMOUD, 2020).

As variantes mais conhecidas da arquitetura *VGG* são a VGG16 e a VGG19., que possuem 16 e 19 camadas com pesos treináveis, respectivamente. De acordo com (SHADEED; TAWFEEQ; MAHMOUD, 2020), a VGG16 consiste em 13 camadas convolucionais seguidas por 3 camadas totalmente conectadas, totalizando aproximadamente 138 milhões de parâmetros. A VGG19, por sua vez, adiciona três camadas convolucionais à estrutura da VGG16, tornando a rede mais profundas,

A principal diferença entre as arquiteturas está na capacidade da VGG19 de capturar características mais complexas devido às suas camadas adicionais. No entanto, isso também resulta em maior custo computacional durante o treinamento e inferência. A Figura 3.6 apresenta a arquitetura do modelo VGG19.

**Figura 3.6** – *Estrutura da arquitetura VGG19.*



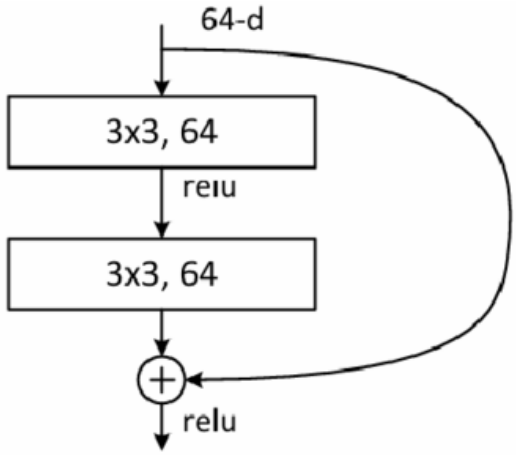

**Fonte:** (SHADEED; TAWFEEQ; MAHMOUD, 2020).

#### 3.3.2.2 Arquitetura *ResNet* (*Residual Networks)*

De acordo com (HE et al., 2015), a arquitetura *ResNet* (*Residual Networks*) foi projetada para resolver problemas críticos no treinamento de redes neurais profundas, como o desvanecimento de gradientes e a degradação do desempenho com o aumento da profundidade. O principal avanço da ResNet está na introdução de conexões residuais (*residual connections*), que permitem que camadas aprendam funções residuais em vez de mapeamentos diretos.

Essas conexões utilizam atalhos (*shortcuts*) que conectam diretamente a entrada de uma camada à sua saída, como ilustrado na Figura 3.7. Isso reduz a complexidade do aprendizado e facilita a otimização, permitindo o treinamento de redes com centenas de camadas (LIANG, 2020).

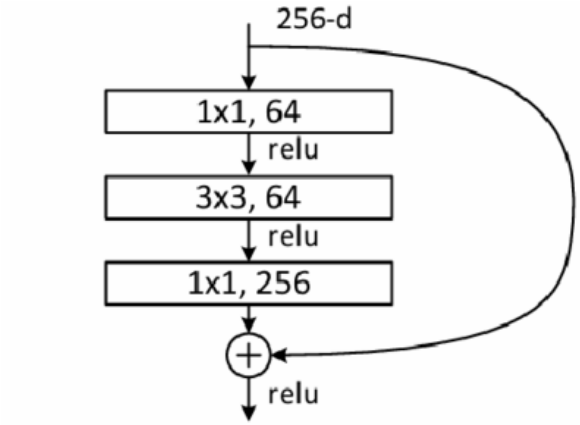
**Figura 3.7** – *Bloco residual básico utilizado na arquitetura ResNet.*




**Fonte:** (AGGARWAL, 2018).

As variantes mais conhecidas da arquitetura *ResNet* são a ResNet50 e a ResNet101, que possuem 50 e 101 camadas treináveis, respectivamente. Ambas utilizam blocos residuais básicos e do tipo *bottleneck*, como ilustrado na Figura 3.8. De acordo com (HE et al., 2015), o bloco *bottleneck* combina convoluções $1 \times 1$ para redução e restauração dimensional com convoluções $3 \times 3$ para extração de características. Essa estrutura reduz o número de parâmetros sem comprometer o desempenho. Além disso, conforme destacado por (ROUSSEAU; DRUMETZ; FABLET, 2020), essa configuração permite que as redes ResNet sejam mais robustas e estáveis em tarefas de classificação em larga escala.

**Figura 3.8** – *Bloco do tipo bottleneck utilizado em arquiteturas mais profundas como a ResNet101.*




**Fonte:**(AGGARWAL, 2018).

## 3.4 Filtros de Borda

### 3.4.1 Filtro *Prewitt*

Conforme conceituado por (FILHO; NETO, 1999), os filtros de borda são projetados para detectar transições abruptas na intensidade dos pixels, conhecidas como bordas. Essas transições geralmente correspondem a limites entre objetos ou variações de textura, sendo essenciais para tarefas como segmentação e reconhecimento de padrões.

Os filtros de borda utilizam operadores baseados em gradiente para calcular as mudanças na intensidade em diferentes direções, destacando bordas horizontais, verticais ou diagonais (GONZALEZ; WOODS, 2008). Segundo (AHMED, 2018), esses operadores são amplamente empregados em aplicações como análise estrutural e detecção de contornos em imagens complexas.

Dentre os operadores clássicos de detecção de bordas, o filtro *Prewitt* é amplamente reconhecido por sua simplicidade e eficiência computacional. Ele utiliza máscaras convolucionais $3 \times 3$ para calcular as derivadas aproximadas da intensidade da imagem nas direções horizontal ($G_x$) e vertical ($G_y$). Conforme descrito por (FILHO; NETO, 1999) e (AHMED, 2018), as máscaras do operador *Prewitt* são definidas pelas matrizes 3.4.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{3.4}$$

Essas máscaras são aplicadas à imagem original por meio de convolução, gerando gradientes que representam a magnitude e a direção das bordas. A combinação das convoluções $G_x$ e $G_y$ é usada para calcular a magnitude do gradiente pela fórmula 3.5.

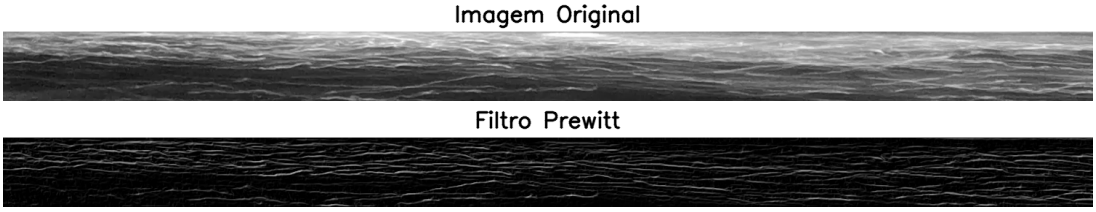
$$|G| = \sqrt{G_x^2 + G_y^2} \tag{3.5}$$

Além disso, a direção da borda pode ser determinada pela Equação 3.6.

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{3.6}$$

A Figura 3.9 ilustra o resultado da aplicação do filtro *Prewitt* em uma imagem contendo bolhas de cavitação. Na imagem original, as bolhas são visíveis, mas suas bordas não estão bem definidas. Após a aplicação do filtro *Prewitt*, as bordas das bolhas são destacadas, facilitando sua análise.

**Figura 3.9** – *Exemplo da aplicação do filtro Prewitt.*




**Fonte:** Autoria Própria.

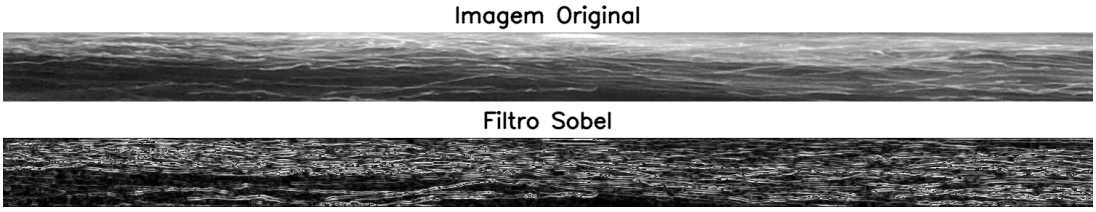
### 3.4.2 Filtro Sobel

O filtro *Sobel* é utilizado devido à sua eficiência em combinar suavização e diferenciação para a detecção de bordas (GONZALEZ; WOODS, 2008). Este operador também utiliza máscaras convolucionais $3 \times 3$ para calcular as derivadas aproximadas da intensidade da imagem nas direções horizontal ($G_x$) e vertical ($G_y$) (FILHO; NETO, 1999). As máscaras do filtro *Sobel* são definidas pelas matrizes 3.6 :

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{3.7}$$

Essas máscaras são aplicadas à imagem original por meio de convolução, gerando gradientes que representam a intensidade das bordas em diferentes direções. A Figura 3.10 ilustra a aplicação do filtro em uma imagem contendo bolhas de cavitação. Na imagem original (acima), as bolhas são visíveis, mas suas bordas não estão bem definidas. Após a aplicação do filtro *Sobel* (abaixo), as bordas das bolhas são claramente destacadas, facilitando sua análise.

**Figura 3.10** – *Exemplo da aplicação do filtro Sobel.*




**Fonte:** Autoria Própria.

## 3.5 Técnicas de Agrupamento de Dados

### 3.5.1 Clusterização de Imagens

(BISHOP, 2006) define que a *clusterização* de imagens consiste em agrupar pixels ou regiões visuais segundo características em comum, podendo incluir atributos de intensidade, cor e textura. Esse procedimento se apoia, em grande parte, na suposição de que partes homogêneas na imagem devem ser reunidas em grupos de propriedades semelhantes, auxiliando no processo de *segmentação* e análise visual mais robusta.

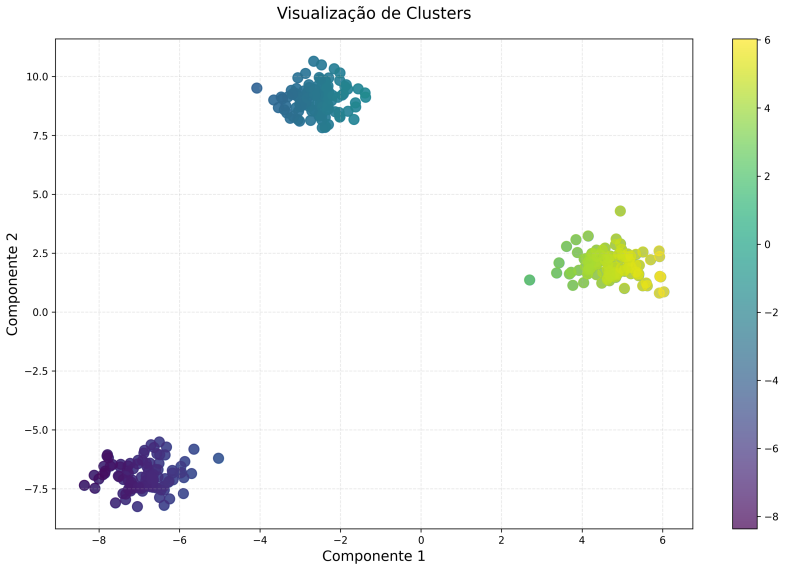
Além disso, conforme salientado por (GéRON, 2017), a eficácia desses agrupamentos está intrinsecamente ligada à qualidade dos dados de entrada e à correta parametrização dos métodos de clusterização, aspectos que são frequentemente negligenciados em implementações superficiais.

Conforme afirmam (EVERITT et al., 2011), a análise de densidade é frequentemente utilizada para enfatizar regiões nas quais os pixels apresentam maior con-

centração de valores similares no *dataset*. Nesse contexto, métodos baseados em *kernel density estimation* são aplicados para estimar a distribuição local dos dados.

Na Figura 3.11, observa-se a distribuição de três grupos distintos de dados, onde cada cluster é representado por uma cor diferente na escala mostrada pela barra lateral. Esta visualização simplificada permite identificar claramente a separação natural entre os grupos, demonstrando como a análise de clusters pode revelar padrões estruturais nos dados.

**Figura 3.11** – *Visualização de Clusters*




**Fonte:** Autoria Própria.

## 3.5.2 Mistura Gaussiana

Os modelos de mistura gaussiana (*Gaussian Mixture Models* - GMM) utilizados em modelagem probabilística para representar distribuições de dados complexas, especialmente aquelas que apresentam múltiplos picos ou modos (EVERITT et al., 2011). Conforme descrito por (BISHOP, 2006), a ideia central do GMM é combinar várias distribuições gaussianas individuais para formar uma única distribuição mais rica e flexível.A densidade conjunta do modelo é expressa pela Equação 3.8:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \tag{3.8}$$

Onde:

$\pi_k \implies$ Peso da $k$-ésima componente, satisfazendo $\sum_{k=1}^{K} \pi_k = 1$;

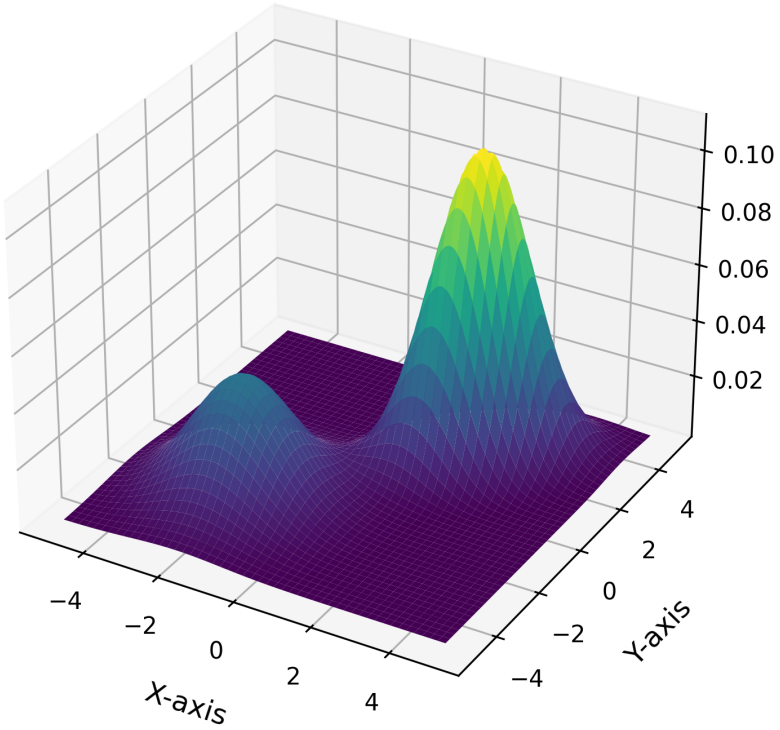
$\boldsymbol{\mu}_k \implies$ Vetor de médias da $k$-ésima componente;

$\boldsymbol{\Sigma}_k \implies$ Matriz de covariância da $k$-ésima componente.

Como destacado pelo estudo de (LU et al., 2024), os GMMs são particularmente úteis em aplicações como segmentação de imagens e reconhecimento de padrões. Cada componente gaussiana pode ser interpretada como representando um agrupamento específico nos dados, o que torna o modelo adequado para tarefas de aprendizado não supervisionado. Além disso, a capacidade do GMM de modelar distribuições multimodais permite capturar estruturas complexas nos dados, mesmo em situações onde os agrupamentos não são claramente definidos.

Figura 3.12 ilustra visualmente o conceito de mistura gaussiana em duas dimensões. Nela, cada "colina"tridimensional representa uma distribuição gaussiana individual, enquanto a superfície combinada reflete a densidade de probabilidade total resultante.

**Figura 3.12** – *Mistura Gaussiana com duas componentes*




**Fonte:** Autoria Própria.

### 3.5.3 Coeficiente de Silhueta

Conforme conceitua (EVERITT et al., 2011), o coeficiente de silhueta é uma métrica utilizada para avaliar a qualidade de agrupamentos em métodos de *clusterização*. Ele mede a coesão interna dos *clusters* e a separação entre eles, fornecendo uma visão quantitativa sobre o quão bem os dados estão alocados em seus respectivos grupos.

Essa métrica é particularmente útil em situações onde o número ideal de *clusters* não é conhecido previamente, permitindo que diferentes configurações sejam comparadas (HAN; KAMBER; PEI, 2011). A fórmula para calcular o coeficiente de silhueta para um ponto $i$ é expressa pela Equação 3.9:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{3.9}$$

Onde:

$s(i) \implies$ Coeficiente de silhueta do ponto $i$, variando entre -1 e 1;

$a(i) \implies$ Distância média entre $i$ e todos os outros pontos do mesmo *cluster* (coesão);

$b(i) \implies$ Menor distância média entre $i$ e os pontos dos *clusters* vizinhos (separação).

## 3.6 Análise de Componentes Principais (PCA)

De acordo com (GéRON, 2017), a análise de Componentes Principais(PCA) é uma técnica estatística que reduz a dimensionalidade de um conjunto de dados ao projetá-lo em novas variáveis (componentes) que concentram a maior parte da variância existente. Dessa forma, torna-se mais fácil identificar padrões e relações importantes, bem como eliminar ruídos que possam prejudicar análises mais aprofundadas (BEN-HUR; GUYON, 2003). Em essência, a PCA transforma variáveis originais possivelmente correlacionadas em variáveis não correlacionadas, conhecidas como componentes principais.

O PCA é iniciado pelo cálculo da matriz de covariância para os dados previ-

amente centrados. Considerando um conjunto de dados $\mathbf{X} \in \mathbb{R}^{n \times d}$, onde $n$ representa o número de amostras e $d$ o número de características, a matriz de covariância é matematicamente definida pela Equação 3.10 (BISHOP, 2006).

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{3.10}$$

Onde:

S $\Longrightarrow$ Matriz de covariância;

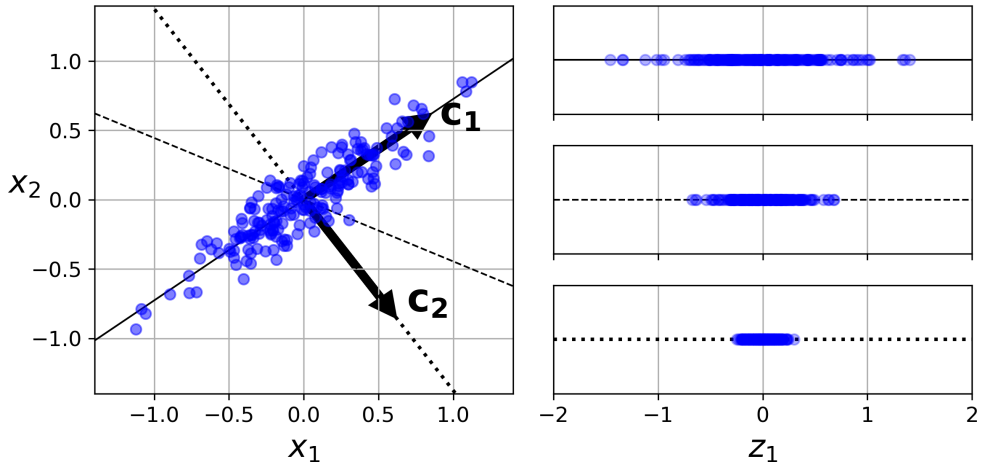
$\mathbf{u}_i$ $\Longrightarrow$ Autovetores (direções principais);

$\lambda_i$ $\Longrightarrow$ Autovalores (variância explicada).

Segundo (BEN-HUR; GUYON, 2003), as direções de baixa variância frequentemente correspondem a ruídos ou variações irrelevantes. Assim, descartá-las pode melhorar significativamente a qualidade dos dados analisados.

A imagem 3.13 ilustra esse processo em um conjunto de dados bidimensional ($x_1, x_2$), onde as componentes principais ($c_1$ e $c_2$) são calculadas. A primeira componente principal ($c_1$) é a direção que preserva a maior variância dos dados, enquanto a segunda componente principal ($c_2$) é ortogonal a $c_1$ e captura a maior parte da variância remanescente.

**Figura 3.13** – *Processo de projeção e análise de componentes principais (PCA).*




**Fonte:** (GéRON, 2017).

Após a projeção, como mostrado na Figura 3.13, observa-se que a maior parte da variância é mantida na direção de $c_1$, enquanto a projeção em $c_2$ ou em direções alternativas preserva uma quantidade significativamente menor de variação.

# Capítulo 4

# Sistema de Detecção Baseado em Aprendizado

## 4.1 Aprendizado por Transferência em Redes Convolucionais

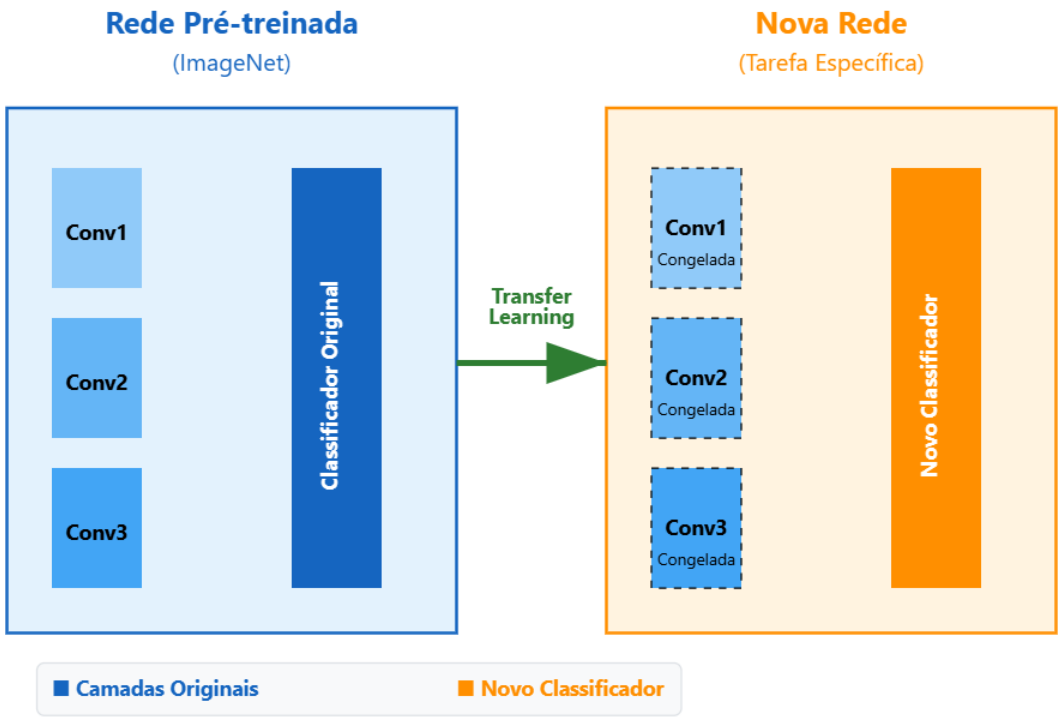
O aprendizado por transferência é uma área de estudo cujo objetivo é reutilizar conhecimentos previamente adquiridos em um domínio-fonte para aprimorar o desempenho em um domínio-alvo, mesmo que haja diferenças entre tarefas e distribuições de dados. Conforme discutido por (RIBANI; MARENGONI, 2019) essa técnica busca reduzir o custo de coleta e rotulagem de dados, aproveitando representações mais abstratas que podem ser adaptadas a novas aplicações.

(ALI et al., 2023) Conceitua que há diferentes estratégias de aprendizado por transferência, incluindo a extração de características, o ajuste fino (*fine-tuning*) e as abordagens baseadas em parâmetros. Na extração de características, camadas intermediárias de uma rede pré-treinada são utilizadas como descritores para uma nova tarefa, evitando treinar toda a rede desde o início . Já o *fine-tuning* consiste em ajustar os pesos de alguns blocos convolucionais, mantendo as primeiras camadas fixas ou parcialmente fixas.

A Figura 4.1 ilustra o processo de aprendizado por transferência aplicado a uma rede convolucional. Inicialmente, a rede pré-treinada (à esquerda) é composta por várias camadas convolucionais e um classificador original. Durante o aprendizado por transferência, as camadas convolucionais são congeladas para

reutilizar as representações previamente aprendidas, enquanto um novo classificador (à direita) é treinado para uma tarefa específica. Essa abordagem reduz significativamente o esforço computacional e a necessidade de grandes volumes de dados anotados, como destacado por (RIBANI; MARENGONI, 2019).

**Figura 4.1** – *Transferência de aprendizado com redes convolucionais.*




**Fonte:** Autoria Própria.

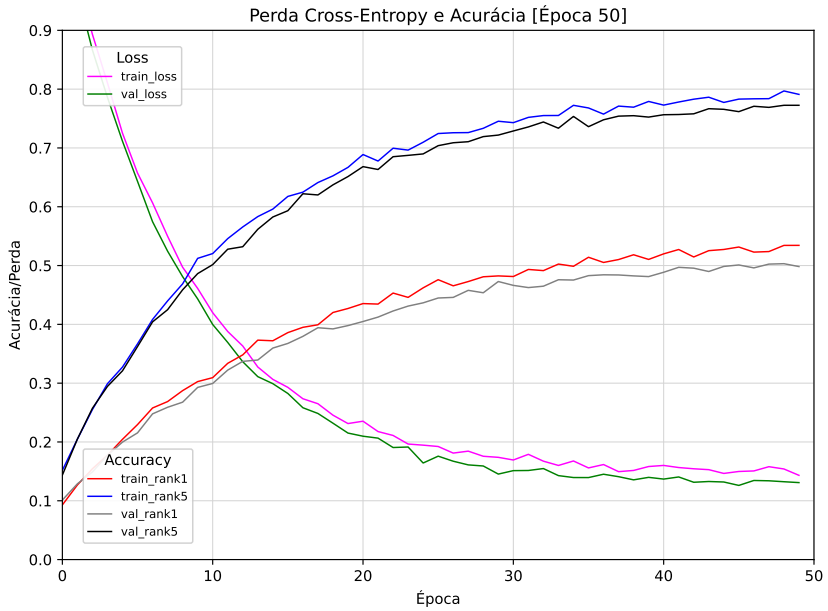
O banco de dados *ImageNet* destaca-se como um dos pilares do aprendizado por transferência em visão computacional, pois contém milhões de imagens rotuladas em mais de mil classes (ROSEBROCK, 2017). O *ImageNet* segue a taxonomia do *WordNet*, agrupando categorias em diversos níveis de granularidade, permitindo a captura de feições derivadas de uma ampla diversidade de objetos.

Além de sua abrangência, o *ImageNet* tornou-se uma referência importante após o surgimento de arquiteturas profundas, tais como *VGG*, que evidenciaram ganhos expressivos em classificação de imagens. A Figura 4.2 apresenta os gráficos de perda (*cross-entropy*) e acurácia para o modelo *VGG* treinado no *ImageNet*.

Nesse contexto, *RANK 1* refere-se à acurácia em que a classe prevista pelo modelo é exatamente a classe correta na primeira tentativa, enquanto *RANK 5* representa a acurácia considerando as cinco previsões mais prováveis do modelo, sendo que a correta deve estar entre elas. Esses valores são amplamente utiliza-

dos para avaliar a precisão de modelos em tarefas de classificação, especialmente em domínios com múltiplas categorias(ROSEBROCK, 2017).

**Figura 4.2** – *Perda e acurácia do VGG no ImageNet.*




**Fonte:** Adaptado de (ROSEBROCK, 2017).

Na Figura 4.2, observa-se que, ao longo de 50 épocas, a perda do modelo diminui progressivamente tanto no conjunto de treino quanto no de validação, enquanto a acurácia para *RANK 1* e *RANK 5* aumenta consistentemente, indicando a eficácia do aprendizado por transferência aplicado ao modelo.

Nesse contexto, bibliotecas como Keras e TensorFlow têm grande importância para o *transfer learning*, pois oferecem *API*s de alto nível e modelos pré-treinados amplamente utilizados em visão computacional, como *VGG*, *ResNet* e *Inception* (TEAM, 2015) . Essas ferramentas simplificam tanto a extração de características quanto o *fine-tuning*, facilitando a configuração rápida das camadas congeladas ou ajustáveis, além de fornecerem suporte para manipulação de grandes bases de dados como o *ImageNet*.

## 4.2 Otimização Baseada em Gradiente

### 4.2.1 Otimizador *ADAM*

O otimizador *ADAM (Adaptive Moment Estimation)* se destaca por usar médias móveis dos gradientes e suas variâncias para fazer ajustes adaptativos. A principal vantagem do *ADAM* é sua capacidade de manter a estabilidade durante o treinamento, independente da escala dos gradientes (ZHANG et al., 2021). Ele faz isso normalizando as atualizações dos parâmetros e incluindo um termo de estabilidade numérica para evitar erros de cálculo. A Equação 4.1 descreve esse processo.

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \frac{\hat{v}_t}{\sqrt{\hat{s}_t} + \epsilon} \tag{4.1}$$

Onde:

$\mathbf{x}_t \implies$ Parâmetros atuais no passo $t$;

$\eta \implies$ Taxa de aprendizado;

$\hat{v}_t \implies$ Gradiente corrigido para viés;

$\hat{s}_t \implies$ Variância corrigida para viés;

$\epsilon \implies$ Termo de estabilidade numérica.

Essa combinação de termos permite que o *ADAM* adapte dinamicamente as atualizações dos parâmetros, tornando o algoritmo eficiente em uma ampla gama de cenários.

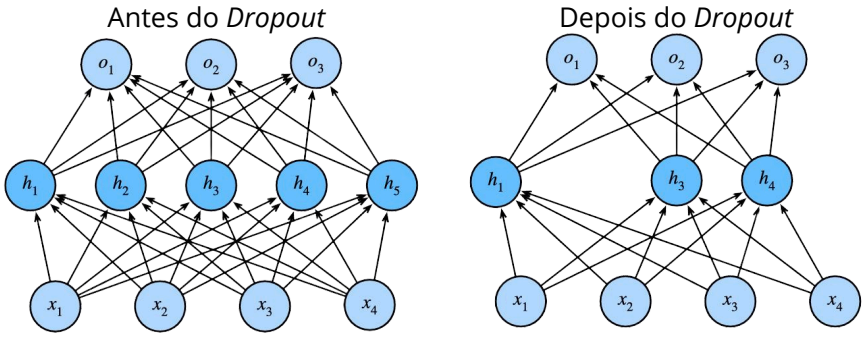
## 4.3 *Dropout* como Método de Regularização

O *dropout* é uma técnica de regularização amplamente utilizada em redes neurais profundas para reduzir o *overfitting*, que ocorre quando o modelo aprende excessivamente os detalhes e ruídos do conjunto de treinamento, comprometendo sua capacidade de generalizar para novos dados (GéRON, 2017). Durante o treinamento, ele desativa aleatoriamente uma fração dos neurônios em cada camada, removendo-os temporariamente da rede. Isso força o modelo a não depender

excessivamente de combinações específicas de neurônios, promovendo maior robustez e generalização (PRINCE, 2023).

A Figura 4.3 ilustra o funcionamento do *dropout*. À esquerda, temos uma rede neural com todos os neurônios ativos. À direita, observa-se a aplicação do *dropout*, na qual alguns neurônios (em cinza) são desativados aleatoriamente durante o treinamento. Esse mecanismo força a rede a depender menos de neurônios individuais ou padrões específicos de ativação, promovendo maior resiliência e generalização.

**Figura 4.3** – *Dropout: antes e após.*




**Fonte:** Adaptado de (ZHANG et al., 2021) .

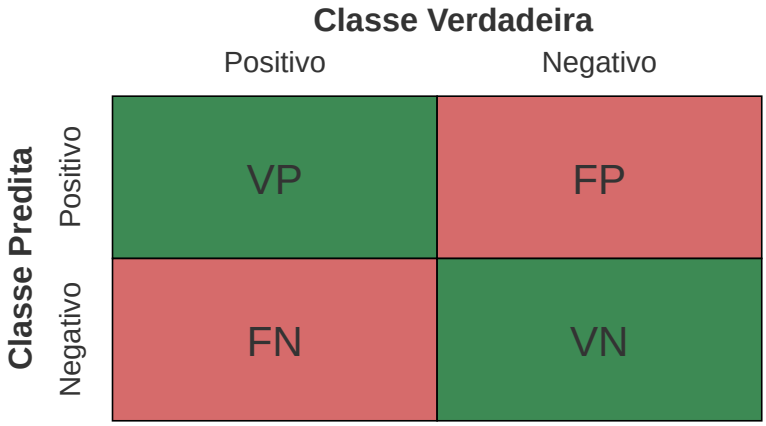
Em termos práticos, os valores de *dropout* variam entre 0 e 1. Um valor de 0 significa que nenhum neurônio é desativado, enquanto um valor de 1 desativa todos os neurônios. Valores intermediários, como 0.5, desativam metade dos neurônios, promovendo um equilíbrio entre regularização e capacidade de aprendizado do modelo (ZHANG et al., 2021).

## 4.4 Métricas de Avaliação

### 4.4.1 Matriz de Confusão

A matriz de confusão é uma ferramenta para avaliar o desempenho de modelos de classificação, sendo amplamente utilizada para analisar os resultados de predições em diferentes classes (VISA et al., 2011). Como mostrado na Figura 4.4.1, a matriz organiza os resultados em quatro categorias principais: Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN), permitindo uma análise detalhada do comportamento do classificador.

**Figura 4.4** – *Matriz de Confusão para Classificação Binária*




**Fonte:** Autoria Própria

.

Para problemas multiclasse, a matriz expande-se para uma dimensão n x n, onde n é o número de classes, permitindo visualizar as classificações corretas na diagonal principal e os erros de classificação entre as diferentes classes nas demais posições (VISA et al., 2011).

A partir desta estrutura, (KULKARNI; CHONG; BATARSEH, 2020) explicam que através da matriz de confusão é possível calcular importantes métricas de avaliação de desempenho. A Acurácia (Equação 4.2) representa a proporção de predições corretas (tanto positivas quanto negativas) em relação ao total de predições.

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.2}$$

A Precisão (Equação 4.3) mede a proporção de predições positivas corretas entre todas as predições positivas feitas.

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{4.3}$$

O Recall (Equação 4.4), também conhecido como sensibilidade ou taxa de verdadeiro positivo, indica a proporção de casos positivos reais que foram corretamente identificados.

$$\text{Recall} = \frac{VP}{VP + FN} \tag{4.4}$$

O F1-score (Equação 4.5) é a média harmônica entre precisão e recall, fornecendo uma métrica balanceada que considera tanto falsos positivos quanto falsos negativos.

$$F1 = 2 * \frac{\text{precisão} * \text{recall}}{\text{precisão} + \text{recall}} \tag{4.5}$$

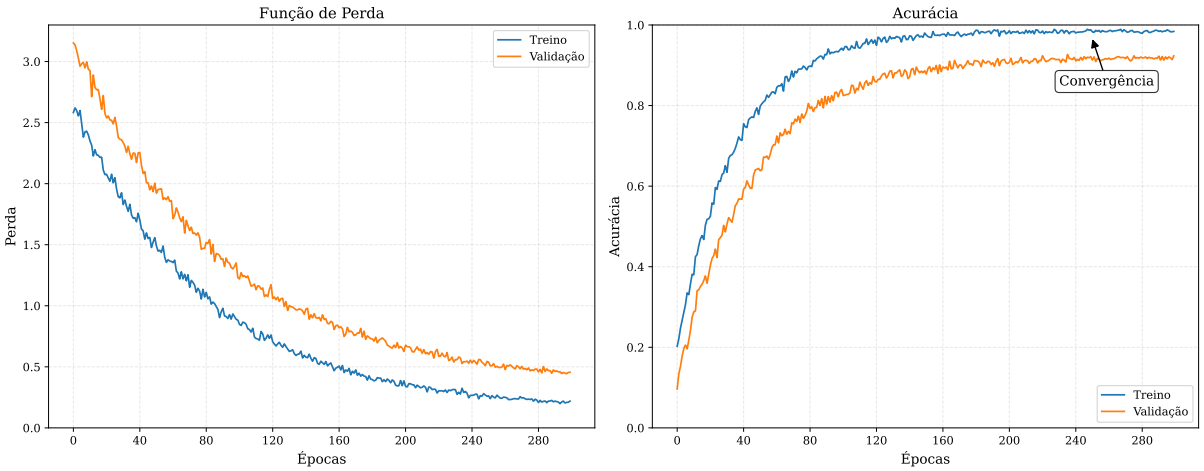
### 4.4.2 Curvas de Acuracia e Perda

Conforme (VIERING; LOOG, 2021), a curva de acurácia indica a proporção de predições corretas do modelo, aumentando conforme ele aprende a classificar os dados. Já a curva de perda mede o erro entre predições e valores reais, diminuindo ao longo do treinamento à medida que os parâmetros são ajustados.

Curvas suaves indicam aprendizado estável, enquanto a convergência reflete o desempenho máximo dado os dados e hiperparâmetros (IBRAHIM, 2023). Comparar curvas de treino e validação permite identificar problemas de generalização: grandes diferenças sugerem sobreajuste, enquanto curvas próximas podem indicar subajuste (VIERING; LOOG, 2021).

Na figura 4.5, observa-se um modelo ideal, onde as curvas convergem suavemente sem discrepâncias significativas, refletindo boa generalização e evitando sobreajuste ou subajuste.

**Figura 4.5** – *Convergência e Generalização do Modelo*




**Fonte:** Autoria Própria.

# Capítulo 5

# Metodologia

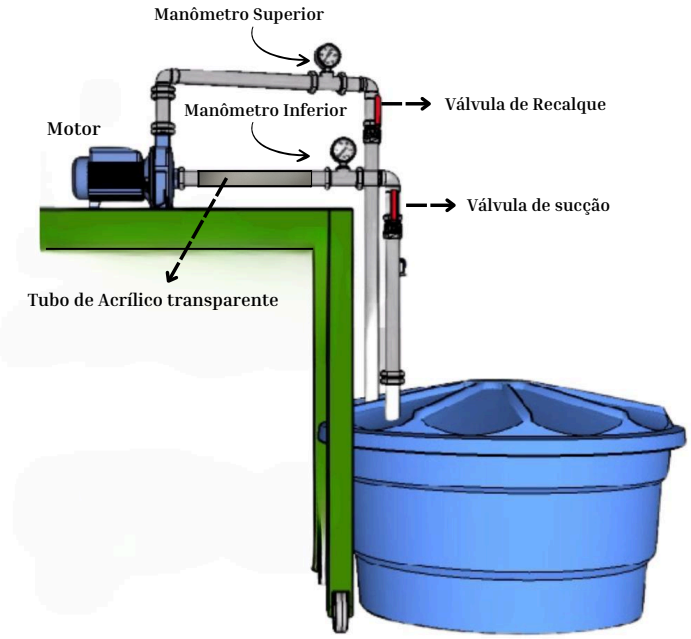
## 5.1 Construção do *Dataset* de Imagens

### 5.1.1 Configuração e Posicionamento do Sistema de Aquisição

O sistema de aquisição de imagens, esquematizado na Figura 5.1, foi desenvolvido para investigar o fenômeno da cavitação. O fluido, impulsionado por uma bomba centrífuga acionada por motor, percorre um tubo de acrílico transparente. Manômetros e válvulas de controle, integrantes do sistema, possibilitam o monitoramento e o ajuste preciso da pressão e do fluxo, variáveis que influenciam diretamente a formação da cavitação. A transparência do tubo de acrílico permite a visualização e captura óptica do escoamento e das bolhas de cavitação.

A captura das imagens das bolhas de cavitação demandou atenção à iluminação. Uma fonte de *LED* foi posicionada abaixo do tubo de acrílico, com a iluminação ambiente reduzida ao mínimo. O contraste de luz aumentou a visibilidade das bolhas, destacando detalhes e melhorando a definição e a qualidade das imagens.

**Figura 5.1** – *Diagrama Esquemático do Sistema de Aquisição.*




**Fonte:** Autoria Própria.

A Figura 5.2 apresenta o sistema de aquisição em operação e detalha o posicionamento da câmera utilizada para registrar as imagens. Fixada em um tripé para maior estabilidade, a câmera foi posicionada em frente ao tubo de acrílico, enquadrando a região onde a formação da cavitação é mais provável. A utilização do tripé permitiu o ajuste fino da altura, ângulo e foco, minimizando distorções ópticas e otimizando a nitidez das imagens capturadas.

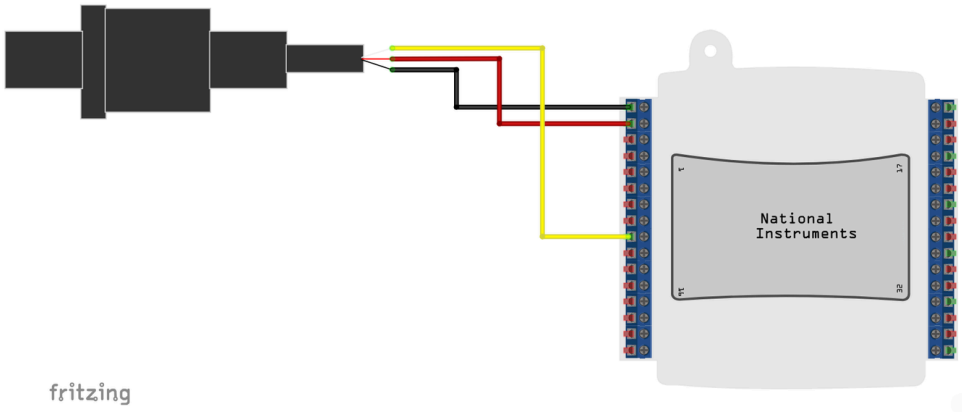
**Figura 5.2** – *Configuração do Posicionamento da Câmera para Captura de Imagens.*


**Fonte:** Autoria Própria.

### 5.1.2 Controle dos Níveis de Pressão de Sucção

Para controlar e monitorar a pressão de sucção do sistema, um transdutor de pressão foi conectado a uma placa de aquisição de dados *National Instruments USB-6341*. Este transdutor converte a pressão física, que afeta a formação da cavitação, em um sinal elétrico analógico. A placa de aquisição, por sua vez, digitaliza esse sinal para que possa ser processado e analisado pelo computador. A Figura 5.3 ilustra as conexões entre o transdutor, a placa de aquisição e o restante do sistema.

**Figura 5.3** – *Diagrama de conexões do transdutor de pressão.*




**Fonte:** Autoria Própria.

As conexões entre os componentes foram feitas da seguinte forma:

- O *Vcc* do sensor foi conectado ao pino *5V* da placa;
- O *GND* do sensor foi conectado ao pino *GND* da placa;
- A saída analógica do sensor foi conectada à entrada de leitura *A7* da placa.

O código em *LabVIEW*, mostrado na Figura 5.4, implementa o seguinte cálculo para converter a tensão lida pelo transdutor em unidades de pressão (mmHg – milímetros de mercúrio). A Equação 5.1 descreve a relação entre a tensão do transdutor e a pressão em *psi* (libras por polegada quadrada). Os valores de tensão máxima (4,5V) e mínima (0,5V), assim como a pressão máxima correspondente (173 psi), foram obtidos do *datasheet* do fabricante (STUDIO, 2017). A Equação 5.1

representa a relação entre tensão e pressão:

$$\text{Pressão (psi)} = \frac{(\text{Valor lido (V)} - \text{Tensão mínima (V)}) \cdot \text{Pressão máxima (psi)}}{(\text{Tensão máxima (V)} - \text{Tensão mínima (V)})} \quad (5.1)$$

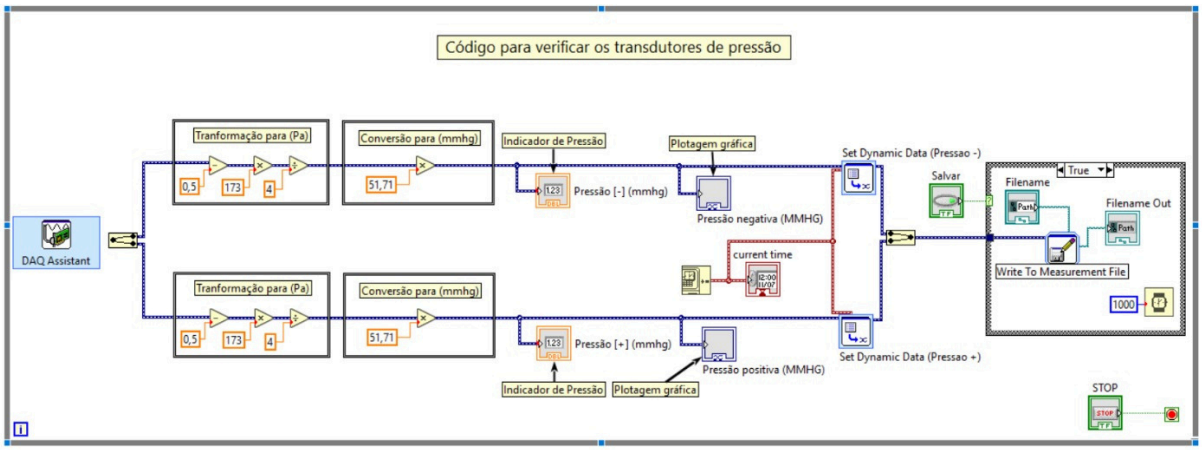
Substituindo os valores na Equação 5.1, obtém-se a Equação 5.2, implementada no código:

$$\text{Pressão (psi)} = \frac{(\text{Valor lido (V)} - 0,5) \cdot 173}{4} \quad (5.2)$$

A pressão em *psi* é então convertida para *mmHg*, unidade padrão utilizada para medir a pressão de sucção no contexto deste trabalho e comumente empregada em estudos de cavitação, multiplicando-se o valor obtido na Equação 5.2 pelo fator de conversão 51,71, apresentado na Equação 5.3.

$$\text{Pressão (mmHg)} = \text{Pressão (psi)} \cdot 51,71 \quad (5.3)$$

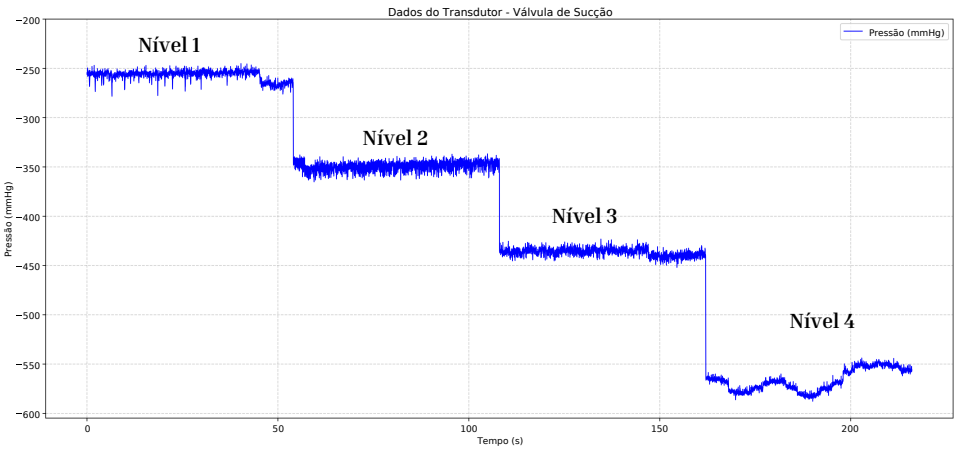
**Figura 5.4** – *Código em LabVIEW para aquisição e processamento dos dados.*




**Fonte:** Autoria Própria.

Após a conversão da pressão para *mmHg*, é possível analisar a variação dos níveis de pressão ao longo do tempo. A Figura 5.5 apresenta os dados adquiridos pelo transdutor, demonstrando os diferentes níveis de pressão medidos e como se comportam durante a operação do sistema para cada nível de cavitação. Além disso, foi nesses pontos que foram capturadas as imagens, permitindo correlacionar visualmente os efeitos da cavitação com as variações de pressão registradas.

**Figura 5.5** – *Variação da pressão de sucção ao longo do tempo.*

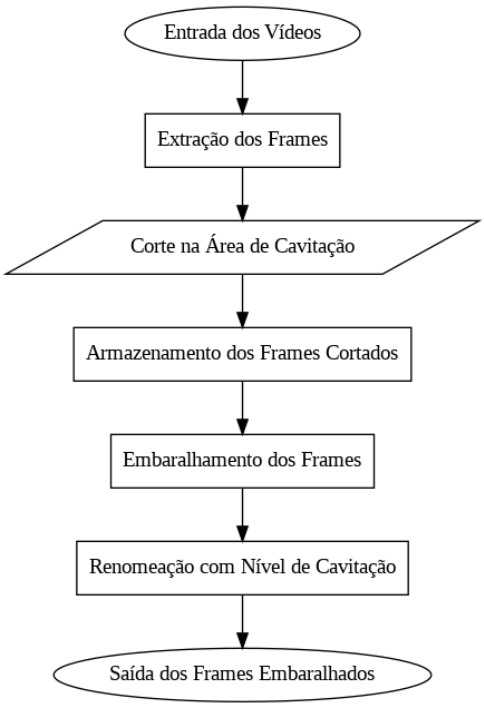



**Fonte:** Autoria Própria.

### 5.1.3 Extração e Processamento dos Frames

Para a construção do *dataset* de imagens utilizado no treinamento do modelo de classificação de cavitação, os vídeos gravados durante os experimentos foram processados. O processo, ilustrado no fluxograma da Figura 5.6, demonstra as etapas sequenciais do processamento.

**Figura 5.6** – *Fluxograma do processo de construção do dataset.*

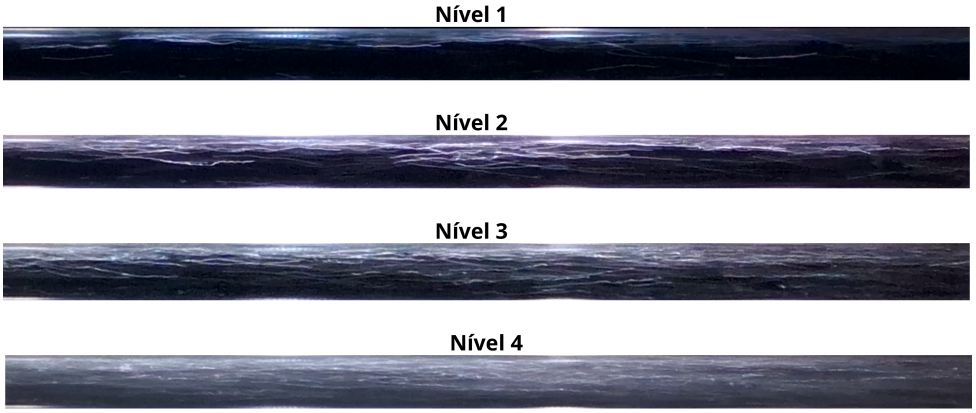



**Fonte:** Autoria Própria.

Conforme ilustrado no fluxograma da Figura 5.6, a construção do *dataset* inicia-se extraindo os *frames* de cada vídeo. Cada *frame* é então recortado, focando no tubo de acrílico onde as bolhas de cavitação se formam, garantindo que apenas a área relevante seja analisada. Os *frames* recortados são embaralhados aleatoriamente para evitar vieses no treinamento, e finalmente renomeados incorporando a informação do nível de cavitação extraída do nome original do vídeo. Por exemplo, 1_0001.png, onde "1"representa o nível de cavitação e "0001"é um índice sequencial.

Os cortes obtidos após o processamento ilustram os diferentes níveis de cavitação capturados nos experimentos. A Figura 5.7 apresenta exemplos de cortes para os quatro níveis de cavitação observados. As imagens exibem os diferentes níveis de cavitação registrados durante os experimentos, onde é possível observar as características visuais das bolhas formadas em cada condição de pressão aplicada. Além disso, os cortes também permitem remover ruídos do ambiente externo que não sejam do tubo.

**Figura 5.7** – *Cortes para os diferentes níveis de cavitação*




**Fonte:** Autoria Própria.

## 5.2 Infraestrutura e Ferramentas Experimentais

### 5.2.1 Bibliotecas e *softwares*

A linguagem *Python* foi utilizada neste trabalho devido à sua importância na ciência de dados, oferecendo uma vasta gama de bibliotecas e ferramentas que facilitam a manipulação, análise e visualização de dados, além de suportar técnicas avançadas de *machine learning* e inteligência artificial (PATIL; MAHANDULE;

GUNJAL, 2024). Para o desenvolvimento do sistema de detecção de cavitação, foi necessário utilizar um conjunto específico de bibliotecas e *softwares* computacionais. A seguir, são apresentadas as principais bibliotecas e *frameworks* utilizados no desenvolvimento deste trabalho.

- ***Thonny IDE***: Ambiente de desenvolvimento integrado utilizado para escrita e *debug* do código, escolhido por sua interface intuitiva e recursos de *debug* que facilitam o desenvolvimento.
- ***TensorFlow.Keras***: *Framework* de *deep learning* que forneceu as implementações das arquiteturas *VGG* e *ResNet*, além das camadas necessárias para construção e treinamento das redes neurais.
- ***Scikit-learn***: Biblioteca que disponibilizou ferramentas essenciais para *clusterização*, métricas de avaliação e pré-processamento dos dados, incluindo *PCA* e *GMM*.
- ***OpenCV***: Biblioteca de visão computacional utilizada para processamento de imagens e vídeos em tempo real, incluindo a implementação do filtro *Prewitt*.
- ***Random***: Módulo responsável pela aleatorização dos *frames* extraídos dos vídeos, garantindo uma distribuição aleatória das imagens para o processo de treinamento.
- ***NumPy***: Biblioteca fundamental para operações numéricas e manipulação eficiente de *arrays* multidimensionais durante o processamento das imagens.
- ***Matplotlib***: Biblioteca de visualização que permitiu a geração de gráficos para análise dos resultados e visualização dos *clusters*.
- ***Flask***: *Framework web* utilizado para criar a interface de monitoramento em tempo real do sistema de detecção de cavitação.

### 5.2.2 Recursos Computacionais para Processamento das Imagens

A implementação do sistema de detecção de cavitação demandou recursos computacionais significativos, especialmente durante a execução dos algoritmos

de clusterização. Para estas etapas, que incluem a extração e processamento dos *frames* dos vídeos, aplicação do filtro *Prewitt* e execução dos algoritmos de agrupamento não supervisionado, foi utilizado um computador local com configurações básicas. Na tabela 5.1 são apresentadas as especificações técnicas do equipamento utilizado, que influenciaram diretamente na execução dos algoritmos e na escolha das estratégias de processamento.

**Tabela 5.1** – *Configurações do hardware utilizado para processamento e clusterização*

| **Componente** | **Especificação** |
|---|---|
| Processador | Intel Core i5 (12ª geração) |
| Sistema Operacional | Windows 11 64-bit |
| Memória RAM | 8 GB DDR4 |
| Armazenamento | SSD 256 GB |

**Fonte:** Autoria Própria.

Para o treinamento do modelo de aprendizado supervisionado, foram utilizados créditos computacionais da plataforma *Google Colab Pro*, que disponibilizou recursos de processamento mais robustos. A utilização de *GPUs* NVIDIA T4 permitiu acelerar significativamente o processo de treinamento das redes neurais convolucionais, possibilitando a execução de múltiplos experimentos com diferentes arquiteturas e parâmetros em um tempo reduzido.

### 5.2.3 Especificações do Sistema de Bombeamento

O sistema de bombeamento utilizado no experimento é equipado com um motor elétrico *WEG*, cujas especificações técnicas são apresentadas na Tabela 5.2.

**Tabela 5.2** – *Especificações técnicas do motor elétrico*

| **Parâmetro** | **Especificação** |
|---|---|
| Potência | 0,37 kW (1/2 cv) |
| Tensão | 127/220 V |
| Corrente | 8,80/3,40 A |
| Frequência | 60 Hz |
| Rotação | 3510 RPM |
| Ip /In | 1,30 A |

**Fonte:** Autoria Própria.

### 5.2.4 Especificações da Câmera

Para a aquisição das imagens do fenômeno de cavitação, foi utilizada uma câmera que permitiu capturar a formação das bolhas. O equipamento foi selecionado considerando sua capacidade de gravação em alta definição para o estudo do fenômeno, cujas especificações são apresentadas na Tabela 5.3.

**Tabela 5.3** – *Especificações técnicas da câmera utilizada*

| **Parâmetro** | **Especificação** |
|---|---|
| Resolução | 50 Megapixels (8165 x 6124) |
| Resolução de Vídeo | Full HD (1920 x 1080) |
| Taxa de Quadros | 30 *FPS* |

**Fonte:** Autoria Própria.

## 5.3 Implementação da Clusterização de Imagens

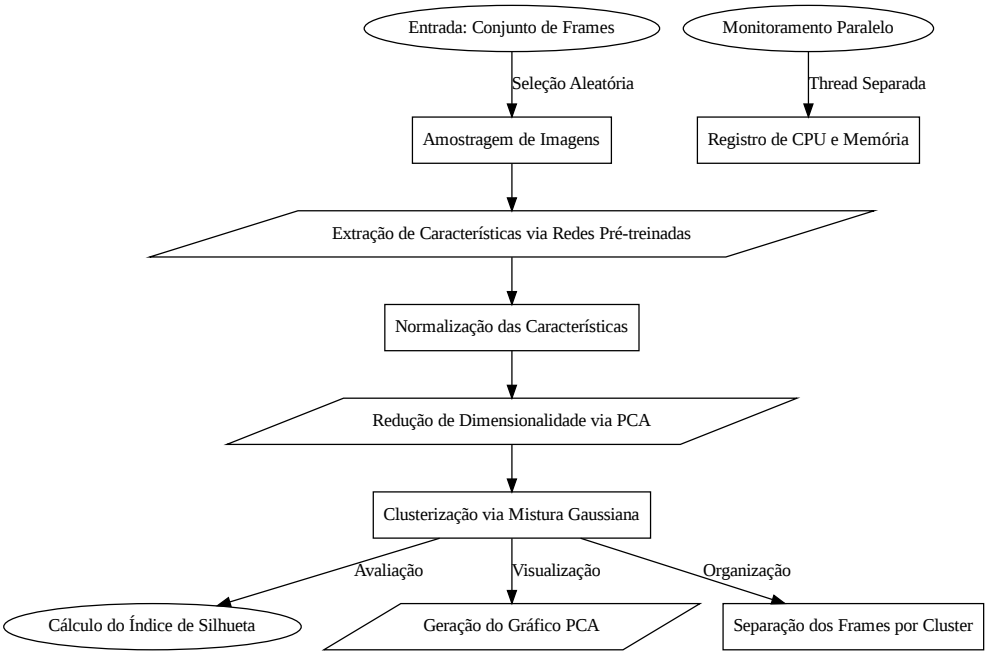
A clusterização de imagens é uma etapa que permite agrupar e classificar padrões visuais semelhantes nos diferentes estados da cavitação, conforme ilustrado no fluxograma 5.8. O processo se integra ao sistema proposto utilizando características extraídas por arquiteturas de redes neurais convolucionais pré-treinadas como *VGG* e *ResNet* para segmentar os dados de forma não supervisionada.

O método implementado utiliza as CNNs como extratores de características das imagens, seguido por uma redução dimensional via PCA com 30 componentes, número definido para preservar a variância significativa dos dados enquanto elimina ruídos. Os dados processados são então agrupados através do modelo de *Gaussian Mixture Model* (GMM), que identifica e separa os padrões em *clusters* distintos. O sistema inclui monitoramento de recursos computacionais e avaliação da qualidade dos agrupamentos através do coeficiente de silhueta, permitindo uma categorização sistemática das diferentes manifestações da cavitação.

Para avaliar o desempenho e a escalabilidade dos modelos, foram realizados experimentos com diferentes proporções do conjunto de dados. O *dataset* completo contava com 19.200 imagens, sendo testadas quatro configurações: 25% (4.800 imagens), 50% (9.600 imagens), 75% (14.400 imagens) e 100% (19.200 imagens). Em cada configuração, manteve-se uma distribuição equilibrada entre os níveis de cavitação, garantindo que cada nível fosse representado com a mesma

quantidade de imagens. Por exemplo, na configuração de 25%, foram utilizadas 1.200 imagens de cada nível, evitando assim qualquer viés na análise dos resultados.

**Figura 5.8** – *Fluxograma do processo de clusterização para separação de níveis de cavitação*




**Fonte:** Autoria Própria.

Os dois modelos com métricas superiores na análise não supervisionada foram selecionados para a fase de classificação supervisionada. A metodologia consistiu em utilizar os agrupamentos gerados pela clusterização como base de dados rotulada para o treinamento supervisionado, estabelecendo assim uma ponte sistemática entre as duas abordagens. Esta estratégia metodológica híbrida possibilita não apenas a validação cruzada dos padrões identificados, mas também um aproveitamento otimizado da base de dados em ambas as etapas do processo.

## 5.4 Rotulagem Supervisionada do Conjunto de Dados

### 5.4.1 Conjunto de Treinamento

O treinamento supervisionado foi realizado utilizando as arquiteturas *VGG19* e *ResNet101* através de transferência de aprendizado (*transfer learning*), onde as camadas convolucionais foram mantidas congeladas e apenas as camadas superiores foram treinadas para a classificação dos quatro níveis de cavitação, conforme ilustrado na Figura 5.9. O conjunto de dados utilizado para o treinamento consistiu em 9.600 imagens, que foram obtidas do processo anterior de clusterização quando utilizados 50% dos dados totais (19.200 imagens). A distribuição entre as classes não se manteve perfeitamente equilibrada devido à natureza do processo de clusterização, que não atingiu 100% de precisão na separação dos níveis.

**Figura 5.9** – *Arquitetura do modelo de classificação supervisionada implementada*


**Fonte:** Autoria Própria.

A implementação do treinamento incluiu técnicas de aumento de dados (*data augmentation*) para melhorar a generalização do modelo, com transformações como espelhamento horizontal e rotações. As camadas treináveis foram configuradas com uma arquitetura específica para o problema, incluindo uma camada de *Global Average Pooling*, seguida por *Dropout* para prevenção de sobreajuste e uma camada densa final com ativação *softmax*. Os parâmetros específicos utilizados no treinamento podem ser observados na Tabela 5.4, que detalha as configurações escolhidas para otimizar o desempenho dos modelos.

### 5.4.2 Conjunto de Validação

O conjunto de validação, composto por 3.200 imagens completamente distintas e independentes das utilizadas no treinamento, foi utilizado para avaliar o desempenho dos modelos durante o processo de aprendizado e evitar o sobreajuste (*overfitting*). Este conjunto independente permite verificar se o modelo está generalizando adequadamente os padrões aprendidos para dados nunca antes vistos, fornecendo métricas importantes como acurácia e perda de validação para ajustar os hiperparâmetros.

**Tabela 5.4** – *Parâmetros de Treinamento*

| **Parâmetro** | **Especificação** |
|---|---|
| Taxa de Aprendizado | 0,0001 |
| Otimizador | *Adam* |
| Épocas | 50 |
| Tamanho do Lote | 32 |
| Taxa de *Dropout* | 0,5 |
| Função de Perda | Entropia Cruzada Categórica |
| Tamanho da Imagem | 224 x 224 pixels |
| Aumento de Dados | Espelhamento Horizontal |

**Fonte:** Autoria Própria.

### 5.4.3 Conjunto de Teste

O conjunto de teste foi composto por vídeos capturados em diferentes dias, mantendo as mesmas condições de pressão dos experimentos anteriores, mas garantindo a independência das amostras utilizadas nas etapas de treinamento e validação. Foi realizado um compilado de vídeos, reunindo diferentes registros para ampliar a representatividade da amostragem. Os vídeos foram processados frame a frame utilizando os modelos .h5 gerados durante o treinamento, permitindo avaliar o desempenho em um cenário próximo ao real de operação. Esta abordagem de teste simula a aplicação prática do sistema, onde o modelo precisa classificar corretamente os níveis de cavitação em novos conjuntos de dados, mesmo que sob condições controladas similares.

Para viabilizar o monitoramento remoto do sistema, foi desenvolvida uma interface web utilizando *HTML*. A integração com o sistema de classificação foi feita através do framework *Flask*, que atua como servidor web, processando requisições em tempo real. Esta arquitetura permite exibir instantaneamente os resultados da classificação na interface web, incluindo o nível de cavitação detectado, a precisão da classificação e o horário da análise, possibilitando o monitoramento remoto através de qualquer dispositivo com acesso à internet.

# Capítulo 6

# Resultados e Discussões

## 6.1 Monitoramento de Recursos Computacionais na Clusterização

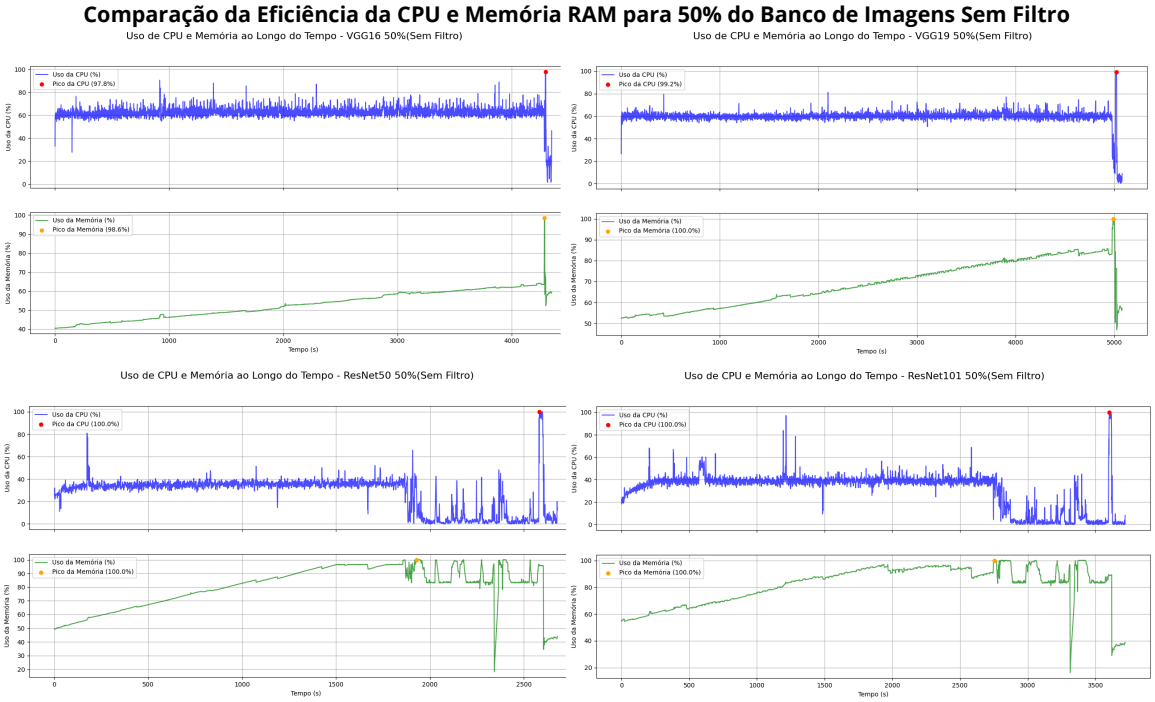
### 6.1.1 Monitoramento de Recursos no Processamento de 50% do *Dataset*

A análise foi realizada utilizando um conjunto de 9.600 imagens, representando 50% do *dataset* total, distribuídas uniformemente entre os diferentes níveis de cavitação. O monitoramento dos recursos computacionais foi executado através de uma *thread* paralela, registrando o consumo de CPU e memória RAM durante todo o processo de clusterização.

#### 6.1.1.1 Análise Sem Filtro

Analisando a Figura 6.1, observa-se um padrão distinto entre as famílias de arquiteturas. As redes *VGG* (*VGG16* e *VGG19*) apresentam picos mais frequentes de utilização de CPU, oscilando entre 60% e 85%, caracterizando um comportamento mais intensivo em processamento. Em contrapartida, as arquiteturas *ResNet* demonstram um padrão mais estável de CPU, porém com crescimento contínuo no consumo de memória RAM ao longo do tempo de execução.

**Figura 6.1** – *Consumo computacional das arquiteturas sem aplicação de filtros*




**Fonte:** Autoria Própria.

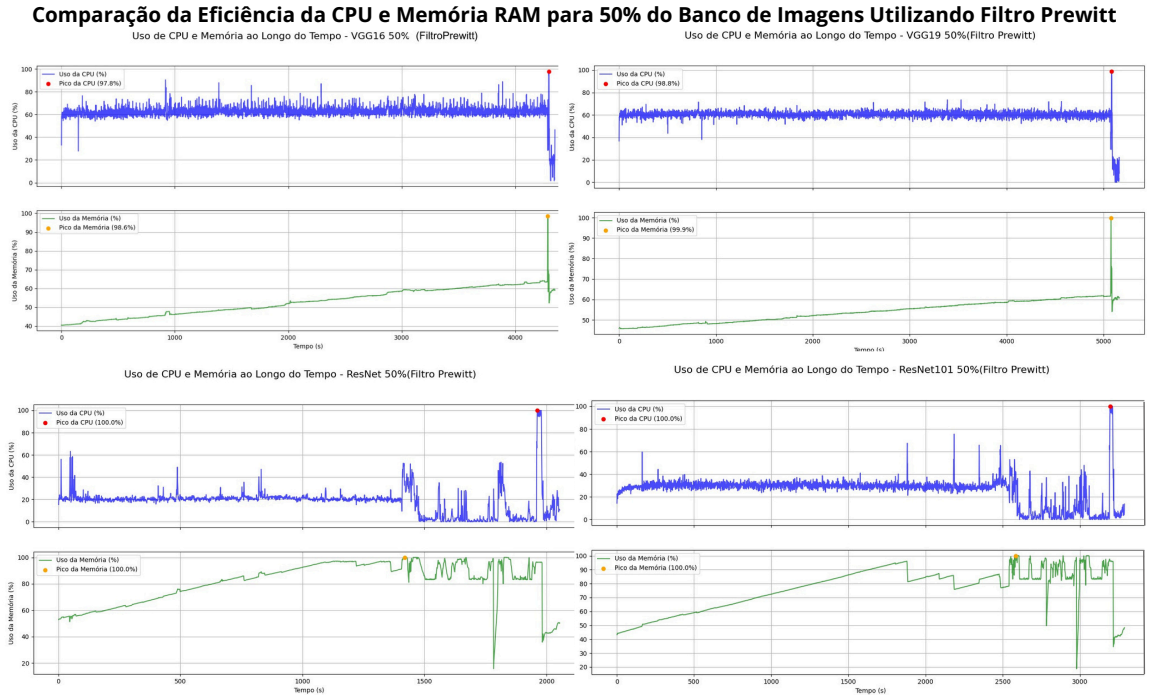
### 6.1.1.2 Análise com Filtro Prewitt

Na Figura 6.2, o padrão se mantém, com as arquiteturas *VGG* apresentando maior variabilidade no uso de CPU, especialmente a *VGG19* com picos frequentes acima de 80%. As *ResNet*s, particularmente a *ResNet101*, mostram uma curva ascendente mais acentuada no consumo de memória RAM, chegando a utilizar aproximadamente 25% mais memória que as *VGG*s no final do processamento.

### 6.1.1.3 Análise com Filtro Sobel

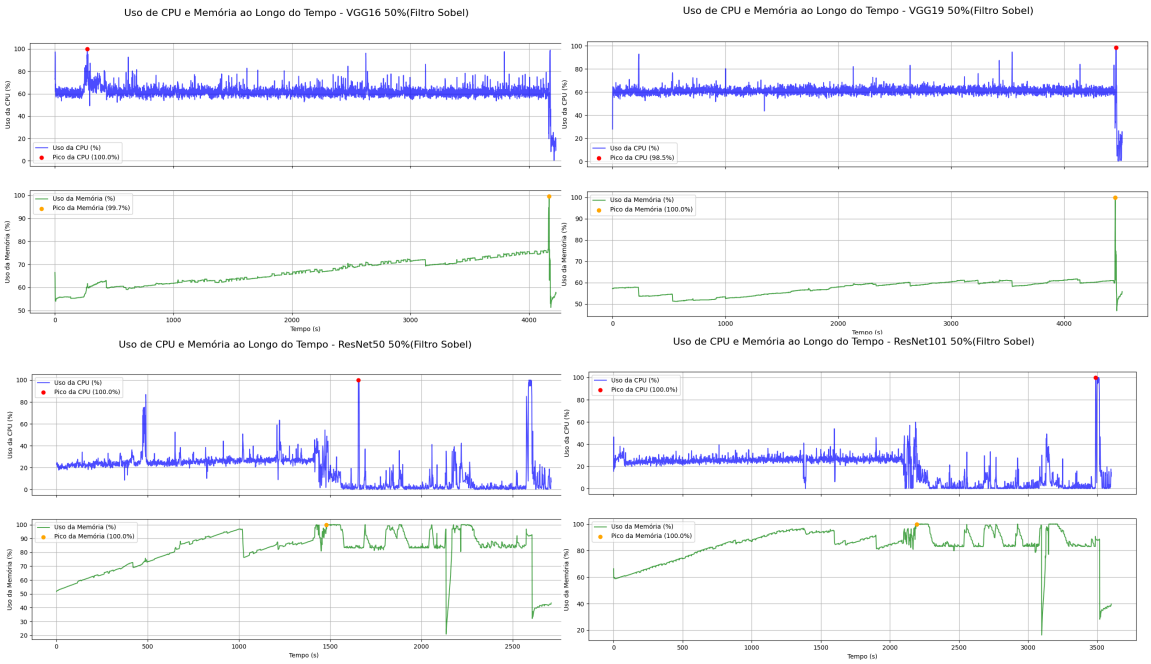
A Figura 6.3 confirma a tendência observada, onde as arquiteturas *VGG* mantêm um padrão de maior consumo de CPU com oscilações significativas, enquanto as *ResNet*s apresentam um crescimento mais expressivo no consumo de memória RAM. A *ResNet101* continua sendo a arquitetura com maior demanda de memória, enquanto a *VGG16* mantém o perfil mais econômico em termos de consumo de RAM.

**Figura 6.2** – *Desempenho de CPU e RAM das Arquiteturas com filtro Prewitt*




**Fonte:** Autoria Própria.

**Figura 6.3** – *Desempenho de CPU e RAM das Arquiteturas com filtro Sobel*



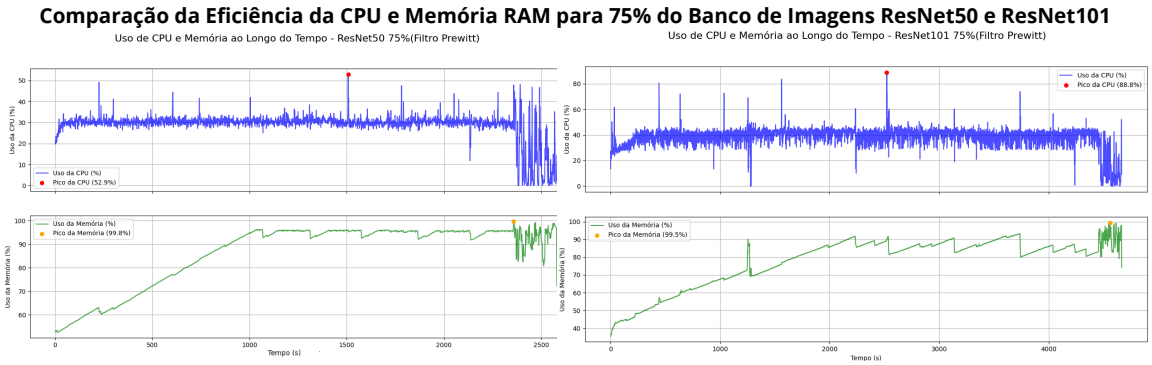

**Fonte:** Autoria Própria.

### 6.1.2 Restrições Computacionais na Execução da ResNet50 e ResNet101 com *Datasets* Extensos

Durante a análise com 75% (14.400 imagens) e 100% (19.200 imagens) do conjunto de dados, as arquiteturas *ResNet50* e *ResNet101* apresentaram falhas críticas de alocação de memória, como evidenciado na Figura 6.4. O monitoramento demonstrou um crescimento abrupto no consumo de memória RAM, atingindo rapidamente o limite do *hardware* disponível antes de apresentar o erro Unable to allocate 10.8 GiB for an array with shape (14400, 100352).

Este comportamento ocorreu devido à tentativa de processamento de imagens de alta resolução em larga escala. A complexidade das arquiteturas *ResNet*, que possuem maior número de camadas e parâmetros treináveis, resultou em uma demanda exponencial de memória para armazenamento de *arrays* multidimensionais durante o processamento. O gráfico de monitoramento mostra claramente o momento em que o sistema atinge seu limite, com a curva de memória RAM apresentando um crescimento vertical acentuado seguido pela interrupção abrupta do processamento, indicando a falha na alocação de memória.

**Figura 6.4** – *Monitoramento do erro de alocação de memória nas arquiteturas ResNet50 e ResNet101 com 75% do dataset*




**Fonte:** Autoria Própria.

### 6.1.3 Monitoramento de Recursos no Processamento do *Dataset* Completo

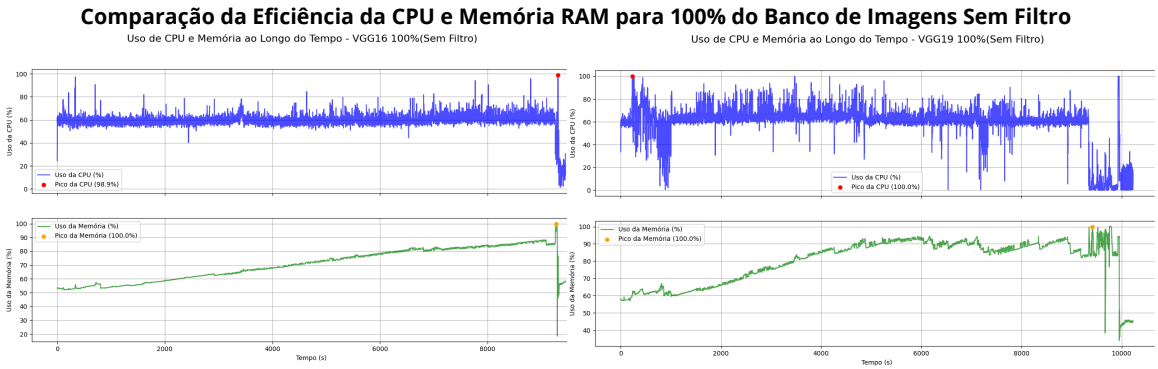
A análise com o conjunto completo de dados (19.200 imagens) foi realizada apenas com as arquiteturas *VGG16* e *VGG19*, devido às limitações de memória previamente identificadas nas arquiteturas *ResNet*. O monitoramento dos recursos

computacionais seguiu o mesmo protocolo estabelecido para 50% do *dataset*.

### 6.1.3.1 Análise Sem Filtro

Como evidenciado na Figura 6.5, as arquiteturas *VGG* mantiveram o padrão de consumo observado anteriormente, porém com maior intensidade. A *VGG16* apresentou picos de CPU mais frequentes, oscilando entre 70% e 90% de utilização, enquanto o consumo de memória RAM mostrou um crescimento mais gradual ao longo do processamento.

**Figura 6.5** – *Monitoramento de CPU e RAM das arquiteturas VGG sem aplicação de filtros para o dataset completo*




**Fonte:** Autoria Própria.

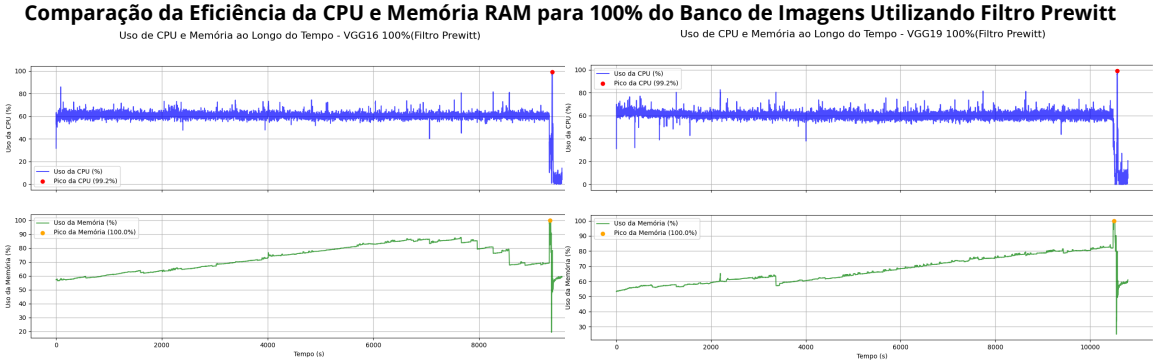
### 6.1.3.2 Análise com Filtro Prewitt

Na Figura 6.15, observa-se que a aplicação do filtro *Prewitt* resultou em um padrão de consumo mais intenso para ambas as arquiteturas. A *VGG19* demonstrou picos de CPU mais frequentes e elevados, ultrapassando 85% de utilização em diversos momentos, enquanto a curva de memória RAM apresentou um comportamento ascendente mais acentuado em comparação com o cenário sem filtro.

### 6.1.3.3 Análise com Filtro Sobel

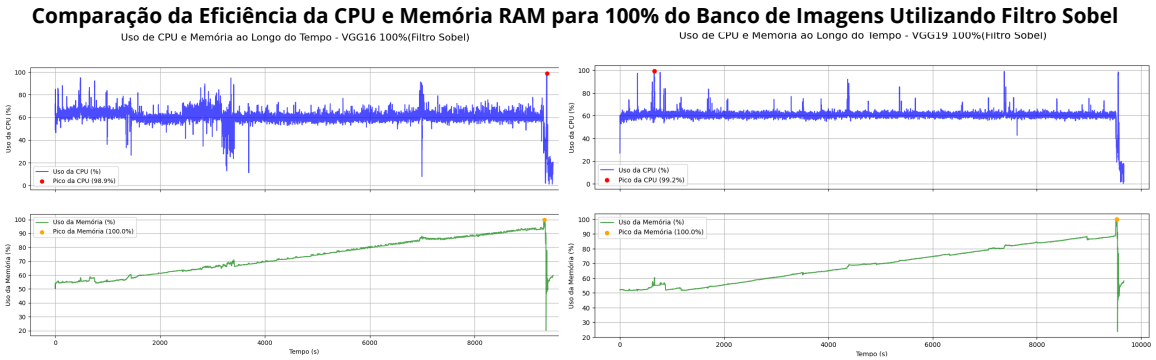
A Figura 6.7 demonstra que o filtro *Sobel* gerou um padrão de consumo similar ao *Prewitt*, com a *VGG19* apresentando maior variabilidade no uso de CPU. O consumo de memória RAM manteve-se controlado para ambas as arquiteturas, embora com níveis mais elevados em comparação aos testes com 50% do *dataset*.

**Figura 6.6** – *Consumo computacional das arquiteturas VGG com filtro Prewitt para o dataset completo*




**Fonte:** Autoria Própria.

**Figura 6.7** – *Consumo computacional das arquiteturas VGG com filtro Sobel para o dataset completo*




**Fonte:** Autoria Própria.

### 6.1.4 Análise de Tempos de Processamento em Função do Volume de Dados

Para uma análise detalhada do desempenho de tempo das arquiteturas, foram realizados testes com diferentes proporções do conjunto de dados. Os tempos de processamento foram medidos em segundos e organizados de acordo com o volume de dados e o tipo de filtro aplicado.

- **Análise com 25% do *Dataset*:** Como evidenciado na Tabela 6.1, com 4.800 imagens, a arquitetura *ResNet50* demonstrou o melhor desempenho temporal em todas as configurações, especialmente no processamento sem filtro (790,7 segundos). As arquiteturas *VGG* apresentaram tempos significativamente maiores, com a *VGG19* requerendo mais de 2.500 segundos em todas as configurações.

**Tabela 6.1** – *Tempos de processamento para 25% do dataset (em segundos)*

| Arquitetura | Prewitt | Sobel | Sem Filtro |
|---|---|---|---|
| *VGG16* | 2217.6 | 2197.9 | 2273.3 |
| *ResNet50* | 954.9 | 838.3 | 790.7 |
| *ResNet101* | 1188.3 | 1463.4 | 1181.2 |
| *VGG19* | 2562.7 | 2314.1 | 2500.7 |

**Fonte:** Autoria Própria.

- **Análise com 50% do *Dataset*:**
  No processamento de 9.600 imagens, observa-se na Tabela 6.2 um aumento proporcional no tempo de processamento. A *ResNet50* manteve o melhor desempenho, com tempos próximos a 2.200 segundos com filtro *Prewitt*, porém as arquiteturas *ResNet* apresentaram um aumento significativo no tempo de processamento, chegando a mais de 230% em relação aos testes com 25% do *dataset*, devido ao crescimento expressivo no consumo de memória RAM.

**Tabela 6.2** – *Tempos de processamento para 50% do dataset (em segundos)*

| Arquitetura | Prewitt | Sobel | Sem Filtro |
|---|---|---|---|
| *VGG16* | 4602.7 | 4180.5 | 4299.5 |
| *ResNet50* | 2195.9 | 2608.9 | 2605.5 |
| *ResNet101* | 3217.4 | 3521.5 | 3263.3 |
| *VGG19* | 5088.9 | 4563.8 | 5026.9 |

**Fonte:** Autoria Própria.

- **Análise com 75% do *Dataset*:** Com 14.400 imagens, apenas as arquiteturas *VGG* completaram o processamento, devido às limitações de memória das *ResNet*s. A *VGG16* manteve tempos menores em todas as configurações.

**Tabela 6.3** – *Tempos de processamento para 75% do dataset (em segundos)*

| Arquitetura | Prewitt | Sobel | Sem Filtro |
|---|---|---|---|
| *VGG16* | 6912.5 | 6833.4 | 6942.4 |
| *VGG19* | 7486.3 | 7771.3 | 7727.4 |

**Fonte:** Autoria Própria.

- **Análise com 100% do *Dataset*** No processamento do conjunto completo de 19.200 imagens, a *VGG16* manteve melhor desempenho que a *VGG19*, com diferença média de aproximadamente 800 segundos entre as arquiteturas.

**Tabela 6.4** – *Tempos de processamento para 100% do dataset (em segundos)*

| Arquitetura | Prewitt | Sobel | Sem Filtro |
|---|---|---|---|
| *VGG16* | 9125.9 | 9380.7 | 9316.9 |
| *VGG19* | 9961.5 | 10.127.5 | 9953.7 |

**Fonte:** Autoria Própria.

### 6.1.5 Síntese Comparativa de Eficiência das Arquiteturas e Filtros

A análise comparativa das arquiteturas demonstrou um claro compromisso entre tempo de processamento e consumo de recursos computacionais. As arquiteturas *ResNet* apresentaram melhor desempenho temporal com conjuntos menores de dados, com a *ResNet50* processando 25% do *dataset* em 790,7 segundos sem filtro, porém foram limitadas pela alta demanda de memória RAM em volumes maiores, impossibilitando o processamento acima de 50% do conjunto de dados.

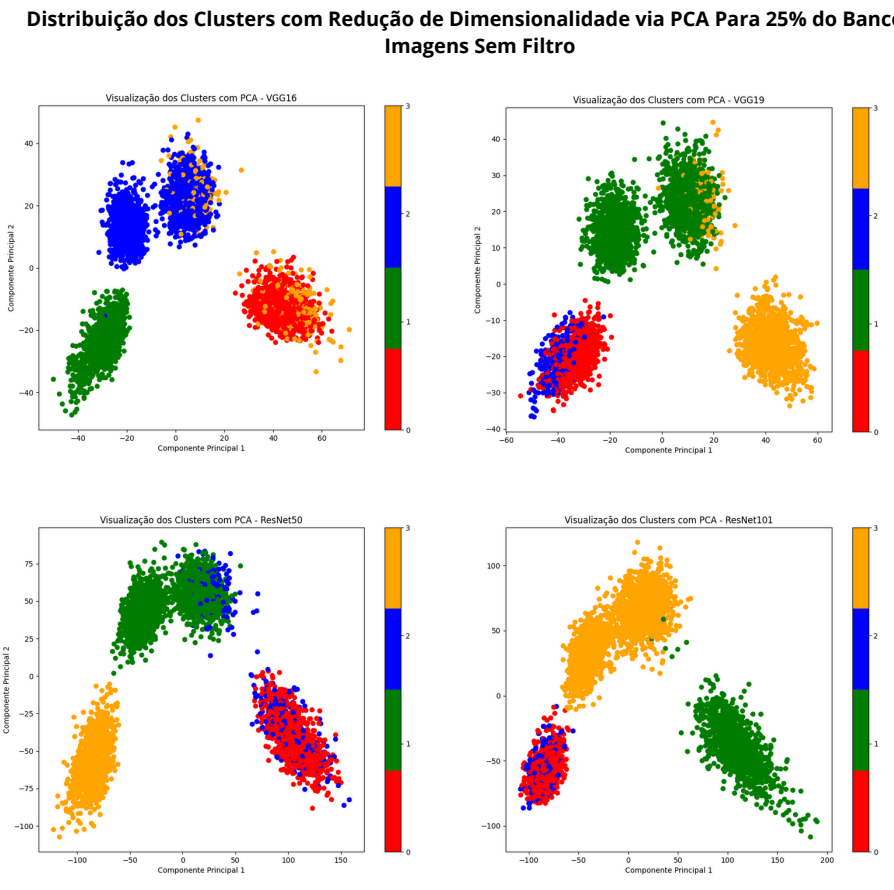
As arquiteturas *VGG*, especialmente a *VGG16*, demonstraram maior estabilidade e escalabilidade, conseguindo processar o *dataset* completo com consumo controlado de memória RAM, embora com tempos de processamento significativamente maiores. Em relação aos filtros, o *Prewitt* apresentou melhor equilíbrio entre consumo de recursos e tempo de processamento, especialmente quando aplicado à arquitetura *VGG16*, que se mostrou a opção mais viável para processamento em larga escala.

## 6.2 Distribuição e Comportamento dos *Clusters* em Múltiplas Escalas de Dados

### 6.2.1 Análise dos *Clusters* para 25% do *Dataset*

- **Sem utilização de filtro:** A análise com 25% do *dataset* (4.800 imagens) demonstrou comportamentos distintos para cada arquitetura, como pode ser observado na Figura 6.8, que apresenta a distribuição espacial dos clusters após redução dimensional via PCA. As Tabelas 6.5, 6.6, 6.7 e 6.8 apresentam os resultados detalhados para cada arquitetura, demonstrando índices de predominância global similares, variando entre 72,56% e 75,00% .

**Figura 6.8** – *Visualização bidimensional dos clusters via PCA para 25% do dataset sem filtro*




**Fonte:** Autoria Própria.

**Tabela 6.5** – *Distribuição dos clusters para arquitetura VGG16 sem filtro - 25% imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 970 | 0 | 0 | 0 | 970 | 100.00% | % |
| **1** | 0 | 0 | 0 | 1199 | 1199 | 100.00% | % |
| **2** | 0 | 1051 | 1200 | 1 | 2252 | 53.28% | % |
| **3** | 230 | 149 | 0 | 0 | 379 | 60.68% | % |
| **Predominância Global** | | | | | | 74.97% | % |

**Fonte:** Autoria Própria.

**Tabela 6.6** – *Distribuição dos clusters para arquitetura VGG19 sem filtro - 25% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 916 | 916 | 100.00% | % |
| **1** | 0 | 1083 | 1200 | 0 | 2283 | 52.56% | % |
| **2** | 0 | 0 | 0 | 284 | 284 | 100.00% | % |
| **3** | 1200 | 117 | 0 | 0 | 1317 | 91.11% | % |
| **Predominância Global** | | | | | | 72.56% | % |

**Fonte:** Autoria Própria.

**Tabela 6.7** – *Distribuição dos clusters para arquitetura ResNet50 sem filtro - 25% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 949 | 0 | 0 | 0 | 949 | 100.00% | % |
| **1** | 0 | 1057 | 1200 | 0 | 2257 | 53.16% | % |
| **2** | 251 | 143 | 0 | 0 | 394 | 63.70% | % |
| **3** | 0 | 0 | 0 | 1200 | 1200 | 100.00% | % |
| **Predominância Global** | | | | | | 75.00% | % |

**Fonte:** Autoria Própria.

**Tabela 6.8** – *Distribuição dos clusters para arquitetura ResNet101 sem filtro - 25% das imagens*

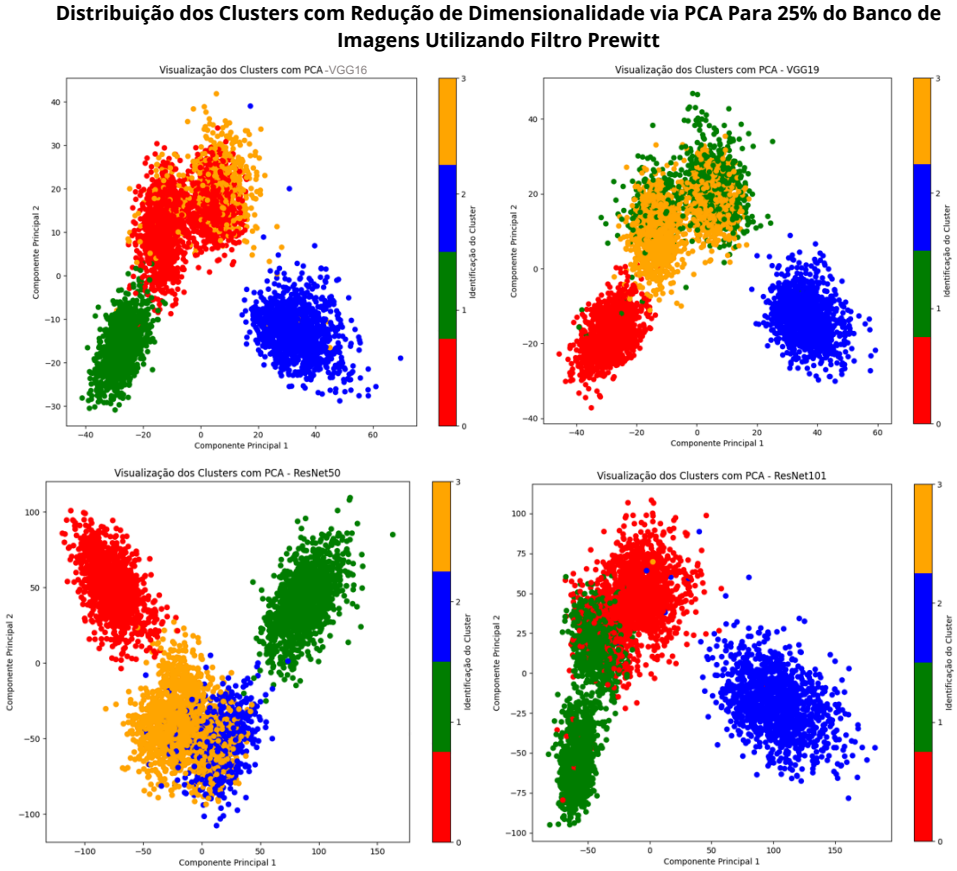
| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 1 | 927 | 928 | 100.00% | % |
| **1** | 1200 | 7 | 0 | 0 | 1207 | 99.42% | % |
| **2** | 0 | 0 | 0 | 273 | 273 | 100.00% | % |
| **3** | 0 | 1193 | 1199 | 0 | 2392 | 50.12% | % |
| **Predominância Global** | | | | | | 74.98% | % |

**Fonte:** Autoria Própria.

- **Com Filtro *Prewitt*:** A aplicação do filtro *Prewitt* resultou em diferentes padrões de agrupamento, como pode ser observado na Figura 6.9, que apresenta a distribuição espacial dos clusters após redução dimensional via PCA. As Tabelas 6.9, 6.11 e 6.10 apresentam os resultados detalhados para cada arquitetura. A arquitetura *ResNet101* não conseguiu realizar a separação adequada em 4 clusters, resultando em um cluster com apenas uma amostra (Cluster 3), impossibilitando o cálculo de sua predominância global. As demais arquiteturas demonstraram índices de predominância global variando entre 82,08% e 85,83%, superiores aos

obtidos sem filtro.

**Figura 6.9** – *Visualização bidimensional dos clusters via PCA para 25% do dataset - Filtro Prewitt*




**Fonte:** Autoria Própria.

**Tabela 6.9** – *Dist. dos clusters para arquitetura VGG16 com filtro Prewitt - 25% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 732 | 1093 | 3 | 1828 | 59.79% | % |
| **1** | 0 | 0 | 7 | 1194 | 1201 | 99.41% | % |
| **2** | 1188 | 3 | 0 | 0 | 1191 | 99.78% | % |
| **3** | 12 | 465 | 100 | 3 | 580 | 80.17% | % |
| **Predominância Global** | | | | | | 82.08% | % |

**Fonte:** Autoria Própria.

**Tabela 6.10** – *Dist. dos clusters para arquitetura VGG19 com filtro Prewitt - 25% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância | % |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 11 | 1190 | 1201 | 99.08% | % |
| **1** | 2 | 566 | 195 | 10 | 773 | 73.22% | % |
| **2** | 1198 | 0 | 0 | 0 | 1198 | 100.00% | % |
| **3** | 0 | 634 | 994 | 0 | 1628 | 61.05% | % |
| **Predominância Global** | | | | | | 82.25% | % |

**Fonte:** Autoria Própria.

**Tabela 6.11** – *Dist. dos clusters para arquitetura ResNet50 com filtro Prewitt - 25% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância | % |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 3 | 1197 | 1200 | 99.75% | % |
| **1** | 1196 | 1 | 0 | 0 | 1197 | 99.91% | % |
| **2** | 4 | 573 | 43 | 0 | 620 | 92.45% | % |
| **3** | 0 | 626 | 1154 | 3 | 1783 | 64.66% | % |
| **Predominância Global** | | | | | | 85.83% | % |

**Fonte:** Autoria Própria.

### 6.2.2 Análise dos *Clusters* para 50% do *Dataset*

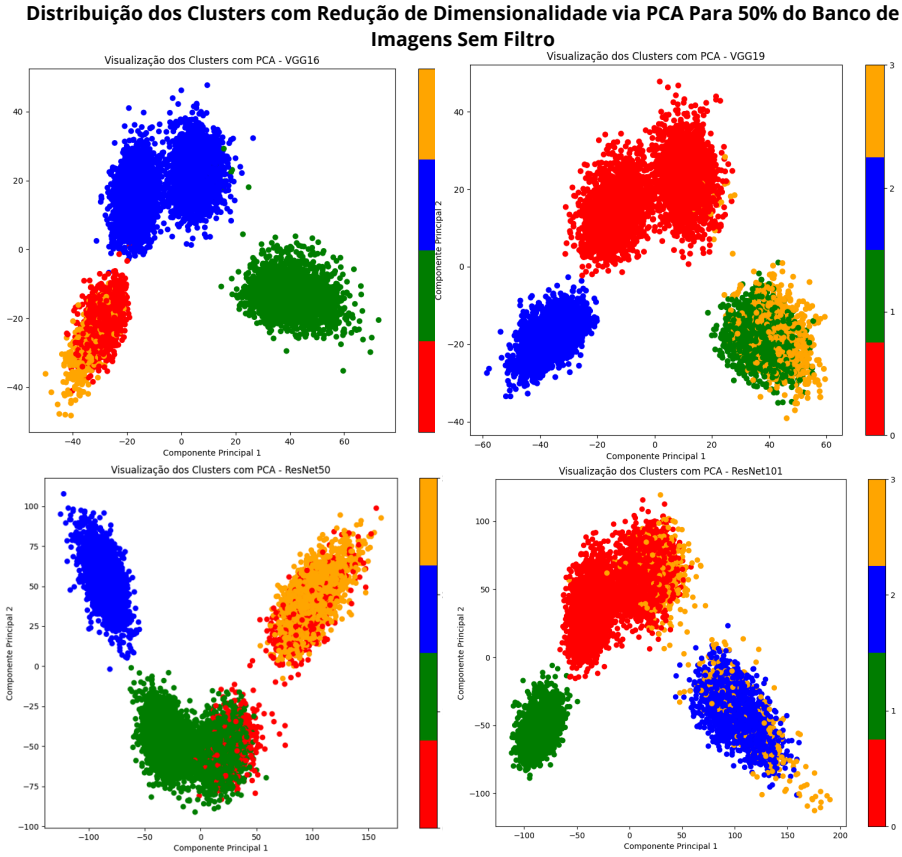
- **Sem utilização de filtro:** A análise com 50% do *dataset* (9.600 imagens) confirmou padrões de agrupamento similares aos de 25%. A Figura 6.10 ilustra a distribuição espacial dos clusters após PCA, enquanto as Tabelas 6.12, 6.13, 6.15 e 6.14 detalham os resultados, com predominância global entre 74,94% e 75,00%. Todas as arquiteturas identificaram bem os níveis extremos de cavitação.

**Tabela 6.12** – *Distribuição dos clusters para arquitetura VGG16 sem filtro - 50% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância | % |
|---|---|---|---|---|---|---|---|
| **0** | 2400 | 16 | 0 | 0 | 2416 | 99.33% | % |
| **1** | 0 | 0 | 4 | 1973 | 1977 | 99.79% | % |
| **2** | 0 | 2384 | 2396 | 1 | 4781 | 50.10% | % |
| **3** | 0 | 0 | 0 | 426 | 426 | 100.00% | % |
| **Predominância Global** | | | | | | 74.95% | % |

**Fonte:** Autoria Própria.

**Figura 6.10** – *Visualização bidimensional dos clusters via PCA para 50% do dataset Sem Filtro*




**Fonte:** Autoria Própria.

**Tabela 6.13** – *Distribuição dos clusters para arquitetura VGG19 sem filtro - 50% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 2376 | 2400 | 0 | 4776 | 50.25% | % |
| **1** | 1532 | 0 | 0 | 0 | 1532 | 100.00% | % |
| **2** | 0 | 0 | 0 | 2400 | 2400 | 100.00% | % |
| **3** | 868 | 24 | 0 | 0 | 892 | 97.30% | % |
| **Predominância Global** | | | | | | 75.00% | % |

**Fonte:** Autoria Própria.

**Tabela 6.14** – *Distribuição dos clusters para arquitetura ResNet101 sem filtro - 50% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância | % |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 2048 | 2396 | 1 | 4445 | 53.90% | % |
| **1** | 0 | 0 | 1 | 2399 | 2400 | 99.95% | % |
| **2** | 2011 | 0 | 0 | 0 | 2011 | 100.00% | % |
| **3** | 389 | 352 | 3 | 0 | 744 | 52.28% | % |
| **Predominância Global** | | | | | | 74.94% | % |

**Fonte:** Autoria Própria.

**Tabela 6.15** – *Distribuição dos clusters para arquitetura ResNet50 sem filtro - 50% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância | % |
|---|---|---|---|---|---|---|---|
| **0** | 485 | 360 | 0 | 0 | 845 | 57.39% | % |
| **1** | 0 | 2040 | 2400 | 0 | 4440 | 54.05% | % |
| **2** | 0 | 0 | 0 | 2400 | 2400 | 100.00% | % |
| **3** | 1915 | 0 | 0 | 0 | 1915 | 100.00% | % |
| **Predominância Global** | | | | | | 75.00% | % |

**Fonte:** Autoria Própria.

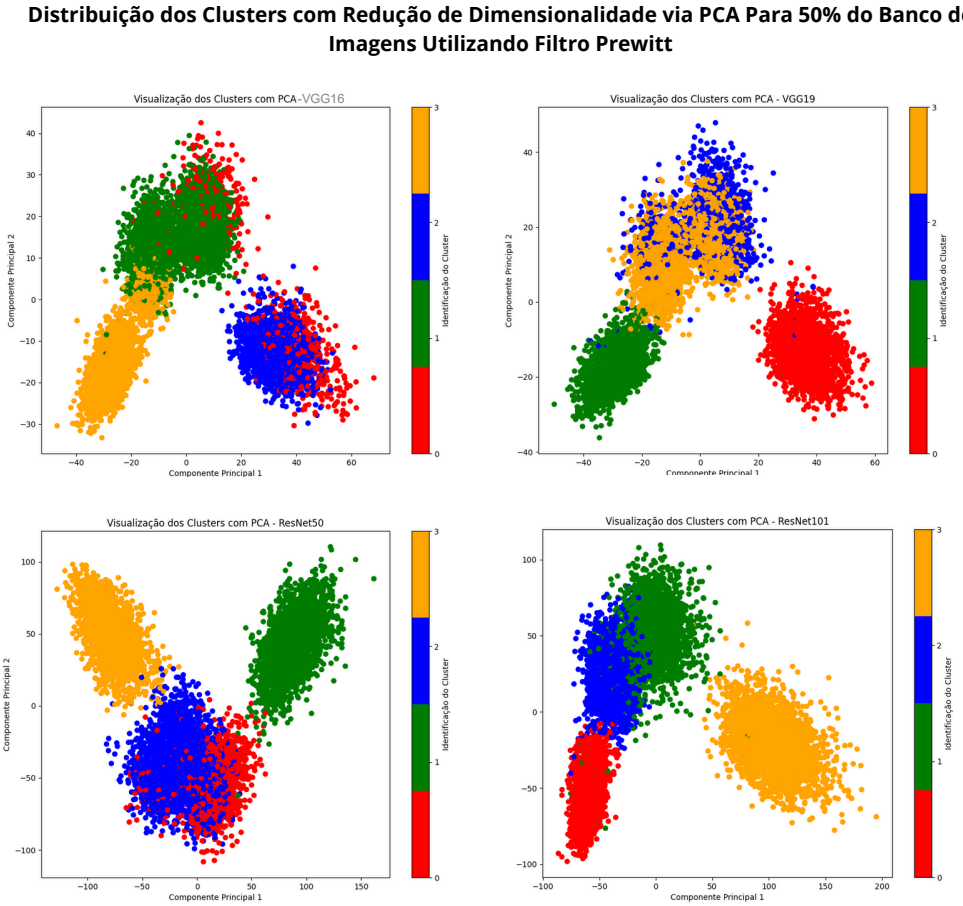
- **Com Filtro *Prewitt*:** A aplicação do filtro *Prewitt* em 50% do *dataset* gerou diferentes padrões de agrupamento, como mostrado na Figura 6.11. As Tabelas 6.16, 6.17, 6.18 e 6.19 detalham os resultados, com predominância global entre 71,61% e 93,45%. Nota-se que a arquitetura *ResNet101* obteve o melhor desempenho, apresentando predominância superior a 86% em todos os clusters.

**Tabela 6.16** – *Dist. dos clusters para arquitetura VGG16 com filtro Prewitt - 50% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância | % |
|---|---|---|---|---|---|---|---|
| **0** | 510 | 316 | 29 | 0 | 855 | 59.64% | % |
| **1** | 0 | 2083 | 1864 | 8 | 3955 | 52.66% | % |
| **2** | 1890 | 1 | 0 | 0 | 1891 | 99.94% | % |
| **3** | 0 | 0 | 507 | 2392 | 2899 | 82.51% | % |
| **Predominância Global** | | | | | | 71.61% | % |

**Fonte:** Autoria Própria.

**Figura 6.11** – *Visualização bidimensional dos clusters via PCA para 50% do dataset - Filtro Prewitt*




**Fonte:** Autoria Própria.

**Tabela 6.17** – *Dist. dos clusters para arquitetura VGG19 com filtro Prewitt - 50% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% |
|---|---|---|---|---|---|---|
| **0** | 2389 | 3 | 0 | 0 | 2392 | 99.87% % |
| **1** | 0 | 0 | 16 | 2379 | 2395 | 99.35% % |
| **2** | 11 | 836 | 270 | 18 | 1135 | 73.65% % |
| **3** | 0 | 1561 | 2114 | 3 | 3678 | 57.47% % |
| **Predominância Global** | | | | | | 80.39% % |

**Fonte:** Autoria Própria.

**Tabela 6.18** – *Dist. dos clusters para arquitetura ResNet50 com filtro Prewitt - 50% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 4 | 1202 | 148 | 0 | 1354 | 88.77% | % |
| **1** | 2396 | 5 | 0 | 0 | 2401 | 99.79% | % |
| **2** | 0 | 1193 | 2245 | 5 | 3443 | 65.20% | % |
| **3** | 0 | 0 | 7 | 2395 | 2402 | 99.70% | % |
| **Predominância Global** | | | | | | 85.81% | % |

**Fonte:** Autoria Própria.

**Tabela 6.19** – *Dist. dos clusters para arquitetura ResNet101 filtro Prewitt - 50% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 7 | 2358 | 2365 | 99.70% | % |
| **1** | 7 | 2089 | 261 | 15 | 2372 | 88.06% | % |
| **2** | 0 | 309 | 2132 | 27 | 2468 | 86.38% | % |
| **3** | 2393 | 2 | 0 | 0 | 2395 | 99.91% | % |
| **Predominância Global** | | | | | | 93.45% | % |

**Fonte:** Autoria Própria.

### 6.2.3 Análise dos *Clusters* para 75% do *Dataset*

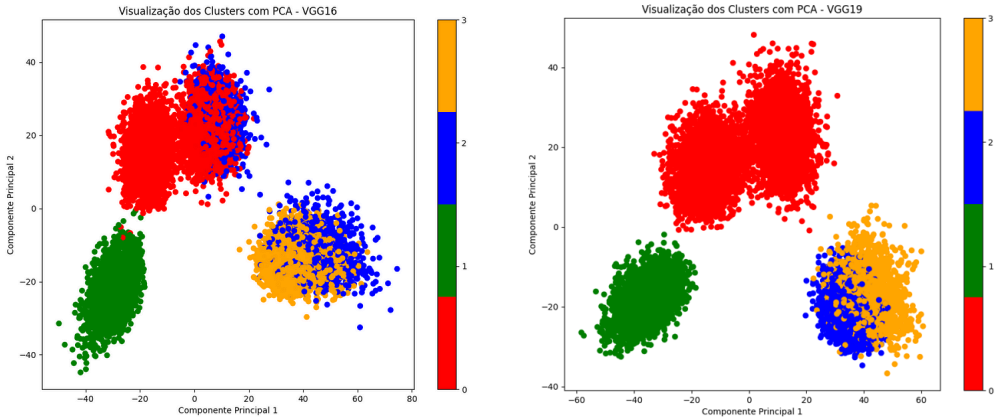
- **Sem utilização de filtro:** A análise com 75% do *dataset* (14.400 imagens) foi realizada apenas para *VGG16* e *VGG19*, devido às limitações de memória das arquiteturas *ResNet*. A Figura 6.12 mostra a distribuição espacial dos clusters após PCA, mantendo padrões de separação similares às análises anteriores. As Tabelas 6.20 e 6.21 apresentam predominância global de 74,93% e 74,99%, respectivamente.

**Tabela 6.20** – *Distribuição dos clusters para arquitetura VGG16 sem filtro - 75% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância (%) |
|---|---|---|---|---|---|---|
| **0** | 0 | 2813 | 3594 | 4 | 6411 | 56.05 |
| **1** | 0 | 0 | 4 | 3596 | 3600 | 99.88 |
| **2** | 1067 | 787 | 2 | 0 | 1856 | 57.48 |
| **3** | 2533 | 0 | 0 | 0 | 2533 | 100.00 |
| **Predominância Global** | | | | | | 74.93 |

**Fonte:** Autoria Própria.

**Figura 6.12** – *Visualização bidimensional dos clusters via PCA para 75% do dataset Sem Filtro*




**Fonte:** Autoria Própria.

**Tabela 6.21** – *Distribuição dos clusters para arquitetura VGG19 sem filtro - 75% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância (%) |
|---|---|---|---|---|---|---|
| **0** | 0 | 3596 | 3600 | 1 | 7197 | 50.02 |
| **1** | 0 | 0 | 0 | 3599 | 3599 | 100.00 |
| **2** | 1762 | 0 | 0 | 0 | 1762 | 100.00 |
| **3** | 1838 | 4 | 0 | 0 | 1842 | 99.78 |
| **Predominância Global** | | | | | | 74.99 |

**Fonte:** Autoria Própria.

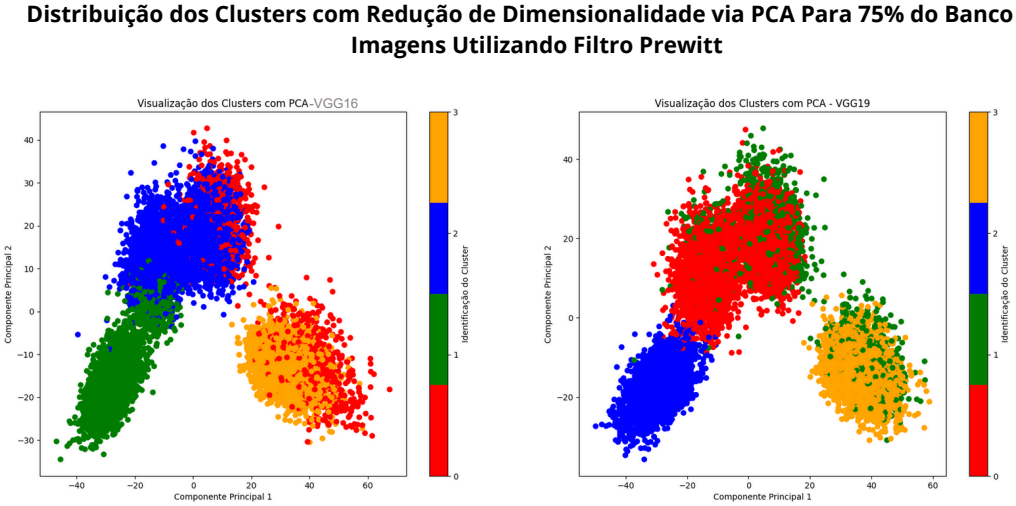
**Com Filtro *Prewitt*:** A aplicação em 75% dos dados destacou diferenças nos agrupamentos (Figura 6.13). As Tabelas 6.22 e 6.23 indicam melhor desempenho global da *VGG19* (76,11%) em relação à *VGG16* (70,60%).

**Tabela 6.22** – *Dist. dos clusters para arquitetura VGG16 com filtro Prewitt - 75% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 915 | 624 | 32 | 0 | 1571 | 58.24% | % |
| **1** | 0 | 0 | 660 | 3591 | 4251 | 84.47% | % |
| **2** | 0 | 2976 | 2908 | 9 | 5893 | 50.50% | % |
| **3** | 2685 | 0 | 0 | 0 | 2685 | 100.00% | % |
| **Predominância Global** | | | | | | 70.60% | % |

**Fonte:** Autoria Própria.

**Figura 6.13** – *Visualização bidimensional dos clusters via PCA para 75% do dataset - Filtro Prewitt*




**Fonte:** Autoria Própria.

**Tabela 6.23** – *Dist. dos clusters para arquitetura VGG19 com filtro Prewitt - 75% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% |
|---|---|---|---|---|---|---|
| **0** | 0 | 2753 | 3420 | 8 | 6181 | 55.33% % |
| **1** | 489 | 847 | 158 | 9 | 1503 | 56.35% % |
| **2** | 0 | 0 | 22 | 3583 | 3605 | 99.38% % |
| **3** | 3111 | 0 | 0 | 0 | 3111 | 100.00% % |
| **Predominância Global** | | | | | | 76.11% % |

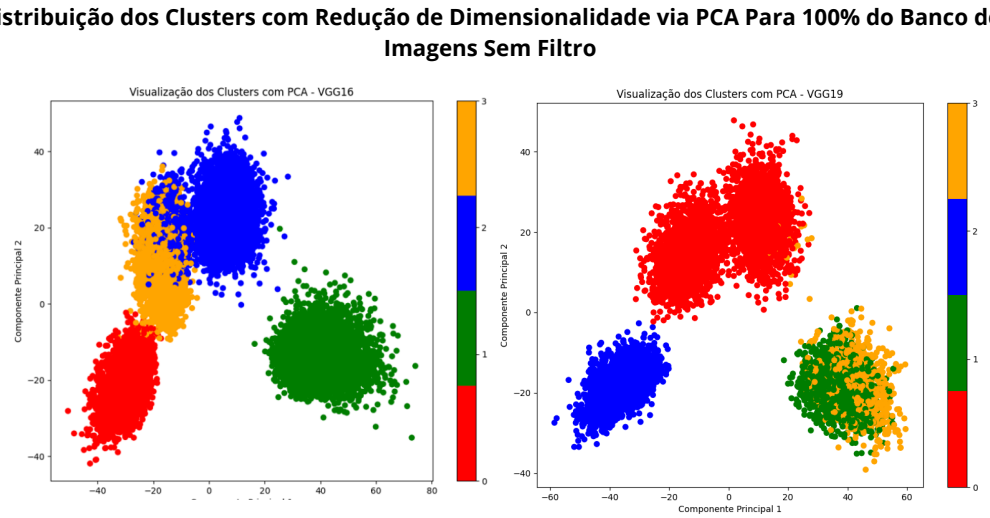
**Fonte:** Autoria Própria.

### 6.2.4 Análise dos *Clusters* para 100% do *Dataset*

- **Sem utilização de filtro:** A análise com o conjunto completo de dados (19.200 imagens) revelou um fenômeno notável com a arquitetura *VGG16*, como pode ser observado na Figura 6.14. As Tabelas 6.24 e 6.25 demonstram que, diferentemente das análises anteriores, a *VGG16* alcançou um expressivo aumento em seu desempenho (92,00%), enquanto a *VGG19* manteve-se estável (74,98%). Este comportamento pode ser atribuído à capacidade da *VGG16* em refinar seus padrões de agrupamento com o aumento do volume de dados, mantendo alta predominância em todos os clusters (80,37% a 99,93%). Um aspecto relevante foi a capacidade da *VGG16* em distinguir os níveis intermediários de cavitação (níveis

2 e 3) com maior eficiência, alcançando predominâncias de 80,37% e 94,69% respectivamente, níveis que tradicionalmente apresentavam maior sobreposição nas análises anteriores.

**Figura 6.14** – *Visualização bidimensional dos clusters via PCA para 100% do dataset Sem Filtro*




**Fonte:** Autoria Própria.

**Tabela 6.24** – *Distribuição dos clusters para arquitetura VGG16 sem filtro - 100% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 4 | 4793 | 4797 | 99.91% | % |
| **1** | 4800 | 3 | 0 | 0 | 4803 | 99.93% | % |
| **2** | 0 | 4396 | 1120 | 2 | 5518 | 80.37% | % |
| **3** | 0 | 401 | 3676 | 5 | 4082 | 94.69% | % |
| **Predominância Global** | | | | | | 92.00% | % |

**Fonte:** Autoria Própria.

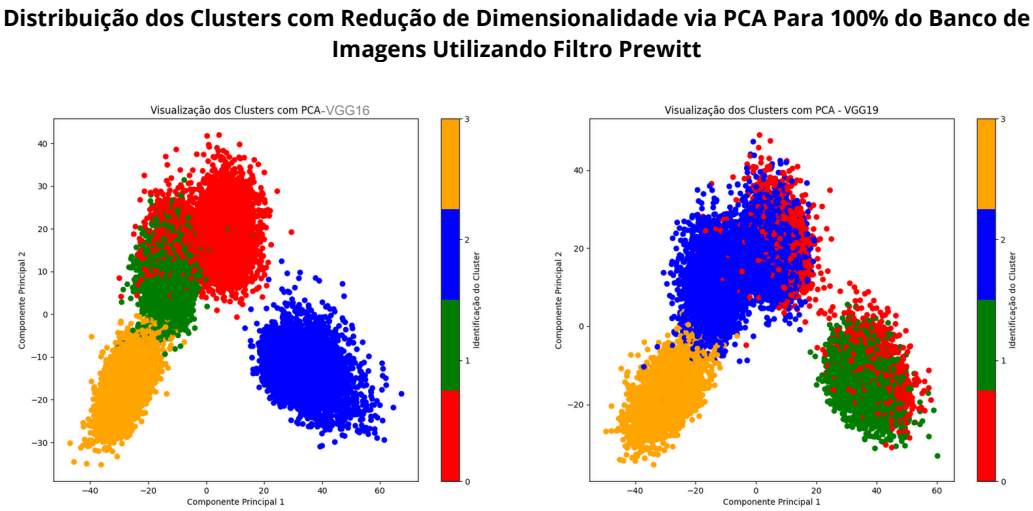
**Tabela 6.25** – *Distribuição dos clusters para arquitetura VGG19 sem filtro - 100% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 4041 | 4041 | 100.00% | % |
| **1** | 0 | 4798 | 4799 | 2 | 9599 | 49.99% | % |
| **2** | 4800 | 2 | 0 | 0 | 4802 | 99.95% | % |
| **3** | 0 | 0 | 1 | 757 | 758 | 99.78% | % |
| **Predominância Global** | | | | | | 74.98% | % |

**Fonte:** Autoria Própria.

- **Com Filtro *Prewitt*:** A aplicação do filtro *Prewitt* no conjunto completo de dados revelou padrões interessantes de agrupamento, como ilustrado na Figura 6.15. As Tabelas 6.26 e 6.27 demonstram que a arquitetura *VGG16* manteve seu excelente desempenho, alcançando uma predominância global ainda maior (92,93%), com três clusters apresentando predominância superior a 98%. Em contraste, a *VGG19* apresentou uma redução em sua capacidade de separação, com predominância global de 74,72%, tendo dificuldade particular na distinção dos níveis intermediários.

**Figura 6.15** – *Visualização bidimensional dos clusters via PCA para 100% do dataset com Filtro Prewitt*




**Fonte:** Autoria Própria.

**Tabela 6.26** – *Dist. dos clusters para arquitetura VGG16 com filtro Prewitt - 100% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância | % |
|---|---|---|---|---|---|---|---|
| **0** | 5 | 4756 | 1280 | 2 | 6043 | 78.70% | % |
| **1** | 0 | 39 | 3508 | 14 | 3561 | 98.51% | % |
| **2** | 4795 | 5 | 0 | 0 | 4800 | 99.89% | % |
| **3** | 0 | 0 | 12 | 4784 | 4796 | 99.74% | % |
| **Predominância Global** | | | | | | 92.93% | % |

**Fonte:** Autoria Própria.

**Tabela 6.27** – *Dist. dos clusters para arquitetura VGG19 com filtro Prewitt - 100% das imagens*

| Cluster | Nível 1 | Nível 2 | Nível 3 | Nível 4 | Total | Predominância% | |
|---|---|---|---|---|---|---|---|
| **0** | 1116 | 1162 | 51 | 1 | 2330 | 52.10% | % |
| **1** | 3684 | 0 | 0 | 0 | 3684 | 100.00% | % |
| **2** | 0 | 3638 | 4722 | 20 | 8380 | 56.34% | % |
| **3** | 0 | 0 | 27 | 4779 | 4806 | 99.43% | % |
| **Predominância Global** | | | | | | 74.72% | % |

**Fonte:** Autoria Própria.

## 6.3 Avaliação da Coesão dos Clusters Através do Coeficiente de Silhueta

Conforme explicado na Seção 3.5.3, o coeficiente de silhueta é uma métrica que varia de -1 a 1, onde valores mais próximos de 1 indicam melhor qualidade dos agrupamentos. As Tabelas 6.28, 6.29 e 6.30 apresentam os coeficientes obtidos para cada arquitetura, filtro e proporção do conjunto de dados.

- **Sem utilização de filtro:**

**Tabela 6.28** – *Coeficientes de silhueta para arquiteturas sem filtro*

| Arquitetura | 25% | 50% | 75% | 100% |
|---|---|---|---|---|
| *VGG16* | 0.1811 | 0.1981 | 0.1948 | 0.1999 |
| *VGG19* | 0.1623 | 0.1853 | 0.2013 | 0.1928 |
| *ResNet50* | 0.1815 | 0.1857 | - | - |
| *ResNet101* | 0.1895 | 0.1926 | - | - |

**Fonte:** Autoria Própria.

- **Com Filtro *Prewitt*:**

**Tabela 6.29** – *Coeficientes de silhueta para arquiteturas com filtro Prewitt*

| Arquitetura | 25% | 50% | 75% | 100% |
|---|---|---|---|---|
| *VGG16* | 0.2631 | 0.2584 | 0.2246 | 0.2302 |
| *VGG19* | 0.2560 | 0.2173 | 0.2191 | 0.2547 |
| *ResNet50* | 0.2760 | 0.2599 | - | - |
| *ResNet101* | 0.2609 | 0.2558 | - | - |

**Fonte:** Autoria Própria.

- **Com Filtro *Sobel*:**

**Tabela 6.30** – *Coeficientes de silhueta para arquiteturas com filtro Sobel*

| Arquitetura | 25% | 50% | 75% | 100% |
|---|---|---|---|---|
| *VGG16* | 0.1774 | 0.1742 | 0.1846 | 0.1369 |
| *VGG19* | 0.1548 | 0.1627 | 0.1522 | 0.1416 |
| *ResNet50* | 0.2298 | 0.2160 | - | - |
| *ResNet101* | 0.2565 | 0.2301 | - | - |

**Fonte:** Autoria Própria.

Analisando globalmente os resultados, observa-se que o filtro *Prewitt* proporcionou a melhor separação entre clusters, com coeficientes consistentemente superiores aos demais cenários, atingindo 0,275 com a *ResNet50* em 25% dos dados. É notável que as arquiteturas *ResNet*, apesar das limitações de processamento com grandes volumes de dados, apresentaram os melhores coeficientes de silhueta em suas execuções (variando de 0,25 a 0,27), sugerindo maior capacidade de extração de características discriminantes. O filtro *Sobel* mostrou comportamento inconsistente, com bom desempenho nas *ResNets* mas queda significativa nas arquiteturas *VGG* com maiores volumes de dados, atingindo os menores valores observados (0,136 para *VGG16* e 0,141 para *VGG19* com 100% dos dados).

## 6.4 Análise Comparativa dos Modelos Supervisionados para Detecção de Cavitação

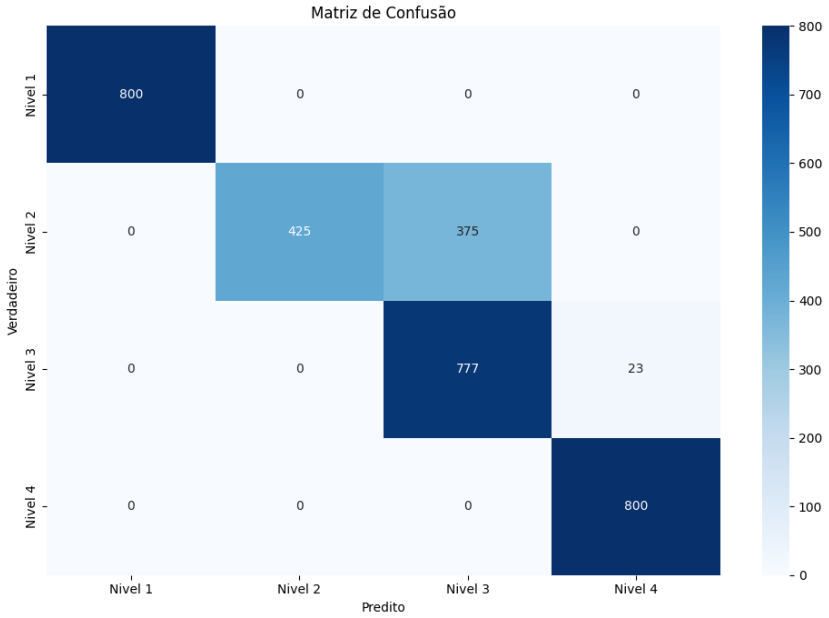
Os modelos *VGG19* e *ResNet101* demonstraram desempenho superior quando treinados com 50% dos dados de *clustering*, evidenciando a eficácia desta abordagem para a detecção de cavitação. O filtro *Prewitt* estabeleceu-se como a alternativa mais adequada para o pré-processamento das imagens, proporcionando características mais distintivas para a classificação. Com base nestas constatações, o treinamento foi conduzido utilizando essas configurações e a metodologia citada na secção 5.4.1.

### 6.4.1 Desempenho do Modelo *VGG19*

A matriz de confusão apresentada na Figura 6.16 demonstra detalhadamente o comportamento do modelo *VGG19* na classificação dos diferentes níveis de cavitação. Em seguida, a Tabela 6.31 apresenta as métricas quantitativas de desem-

penho do modelo.

**Figura 6.16** – *Matriz de Confusão do modelo VGG19*




**Fonte:** Autoria Própria.

A matriz revela que o modelo obteve classificação perfeita para o Nível 1, com todas as 800 amostras corretamente identificadas. Para o Nível 2, observa-se uma divisão nas predições, onde 421 amostras foram classificadas corretamente, enquanto 379 foram erroneamente atribuídas ao Nível 3. O Nível 3 apresentou 777 classificações corretas, com apenas 23 amostras confundidas com o Nível 4. Por fim, o Nível 4 também demonstrou excelente performance, com 800 classificações precisas.

O modelo *VGG19* apresentou boa classificação, com variações entre os níveis de cavitação. Alguns níveis foram classificados com alta precisão, enquanto outros, como o Nível 2, apresentaram desafios, com precisão perfeita, mas baixa recuperação. Apesar disso, as médias gerais do modelo permaneceram consistentes e satisfatórias, indicando uma capacidade geral sólida de classificação, com nuances de performance entre os níveis de complexidade.

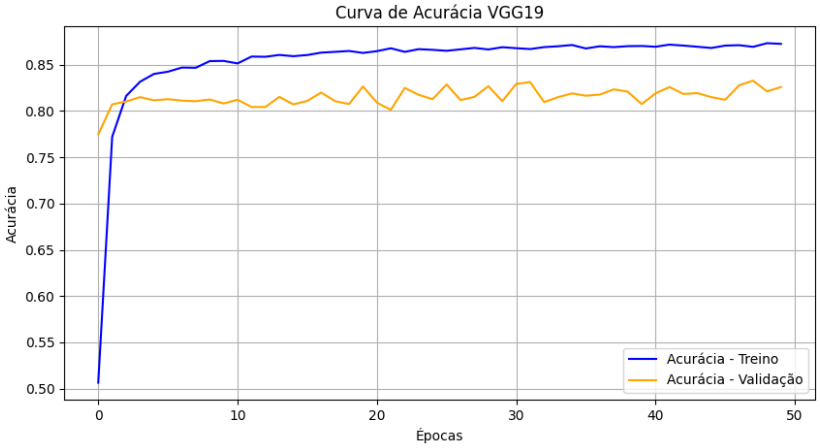
A acurácia do *VGG19* (Figura 6.17) evoluiu consistentemente, com a validação acompanhando o treinamento, indicando bom equilíbrio entre generalização e aprendizado. A convergência ocorreu perto da época 20, estabilizando-se em seguida. A perda (Figura 6.18) diminuiu inicialmente, estabilizando-se gradualmente. A similaridade entre curvas de treinamento e validação sugere ausência de *overfitting* significativo e adequada generalização dos dados.

**Tabela 6.31** – *Relatório de Classificação do modelo VGG19*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Nivel 1 | 1.00 | 1.00 | 1.00 | 800 |
| Nivel 2 | 1.00 | 0.53 | 0.69 | 800 |
| Nivel 3 | 0.67 | 0.97 | 0.80 | 800 |
| Nivel 4 | 0.97 | 1.00 | 0.99 | 800 |
| accuracy | | | 0.88 | 3200 |

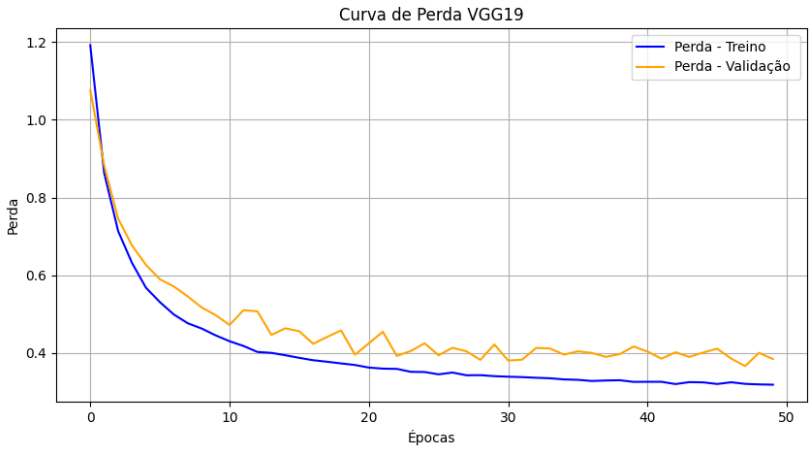
**Fonte:** Autoria Própria.

**Figura 6.17** – *Curva de Acurácia do modelo VGG19*




**Fonte:** Autoria Própria.

**Figura 6.18** – *Curva de Perda do modelo VGG19*

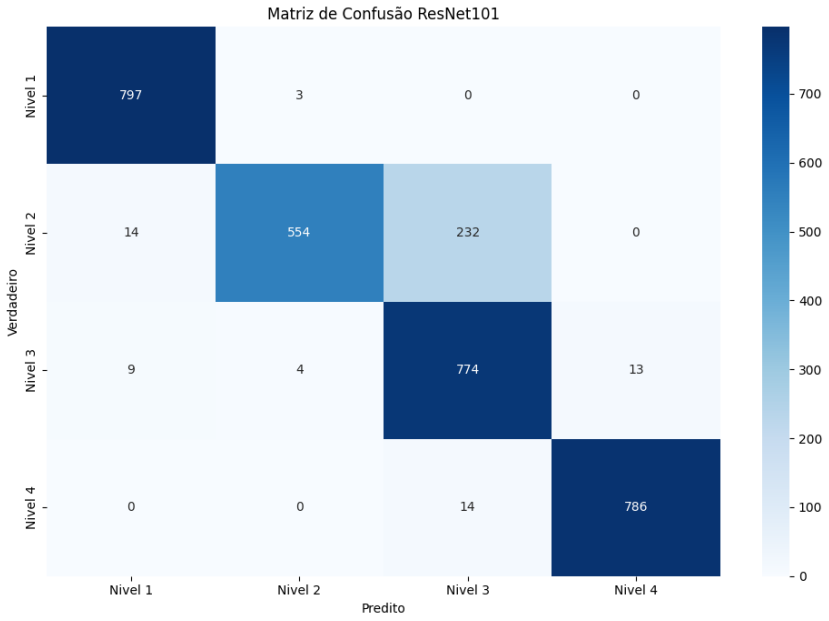



**Fonte:** Autoria Própria.

### 6.4.2 Desempenho do Modelo *ResNet101*

O modelo *ResNet101* demonstrou um desempenho superior na classificação dos níveis de cavitação, conforme evidenciado pela matriz de confusão na Figura 6.19 e pelas métricas detalhadas na Tabela 6.32.

**Figura 6.19** – *Matriz de Confusão do modelo ResNet101*




**Fonte:** Autoria Própria.

A matriz de confusão mostra que o modelo classificou corretamente todas as amostras do Nível 1 e Nível 4 (800 cada). No Nível 2, 551 amostras foram acertadas, mas 249 foram confundidas com o Nível 3, que por sua vez teve 777 classificações corretas e apenas 23 erros para o Nível 4.

**Tabela 6.32** – *Relatório de Classificação do modelo ResNet101*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Nivel 1 | 0.97 | 1.00 | 0.98 | 800 |
| Nivel 2 | 0.99 | 0.69 | 0.81 | 800 |
| Nivel 3 | 0.76 | 0.97 | 0.85 | 800 |
| Nivel 4 | 0.98 | 0.98 | 0.98 | 800 |
| accuracy | | | 0.91 | 3200 |

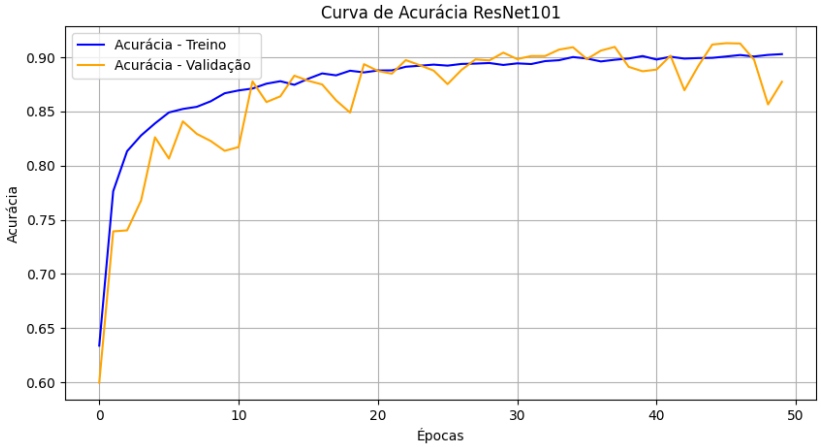
**Fonte:** Autoria Própria.

O relatório de classificação destaca o bom desempenho do *ResNet101* nos níveis 1 e 4, com alta *precision* e *recall*. O nível 2 apresenta uma lacuna significativa no *recall* (*69%*), indicando dificuldade em identificar todas as amostras corretamente.

Já o nível 3 equilibra *precision* (*76%*) e *recall* (*97%*), resultando em um desempenho geral sólido, com acurácia de 91%.

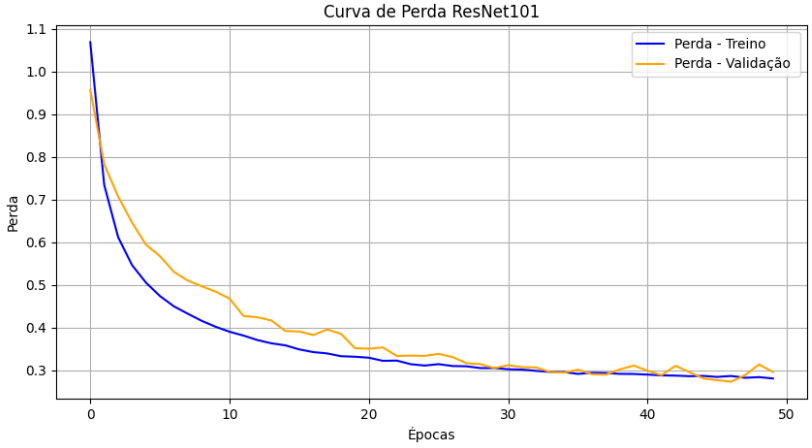
A curva de acurácia do modelo *ResNet101*, apresentada na Figura 6.20, demonstra uma evolução estável durante o treinamento, com convergência eficiente e bom equilíbrio entre as métricas de treinamento e validação. A curva de perda, ilustrada na Figura 6.21, exibe uma redução consistente do erro, indicando um processo de aprendizagem adequado sem sinais significativos de *overfitting*.

**Figura 6.20** – *Curva de Acurácia do modelo ResNet101*




**Fonte:** Autoria Própria.

**Figura 6.21** – *Curva de Perda do modelo ResNet101*

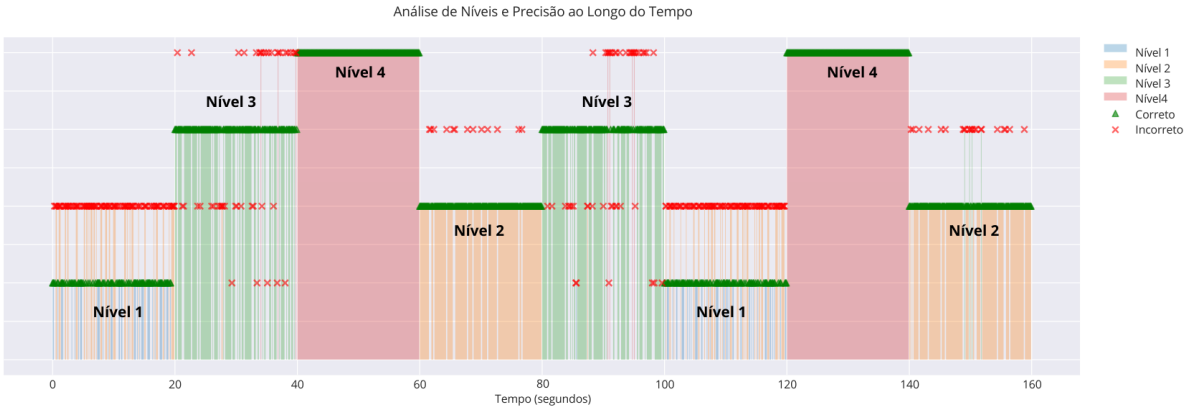



**Fonte:** Autoria Própria.

## 6.5 Análise da Classificação em Dados de Teste

Seguindo a metodologia do capítulo 5.4.3, os modelos treinados foram testados com dados compilados de diferentes dias e iluminações. Para uma análise mais ampla, foram usadas sequências de vídeos capturadas em distintos períodos operacionais, permitindo avaliar o desempenho em variadas condições ambientais e estados do sistema. O modelo *VGG19* foi escolhido para os testes práticos por consumir menos memória que o *ResNet101*, aspecto essencial para aplicações em tempo real. A Figura 6.22 mostra a precisão para cada nível de cavitação, evidenciando a eficácia do modelo.

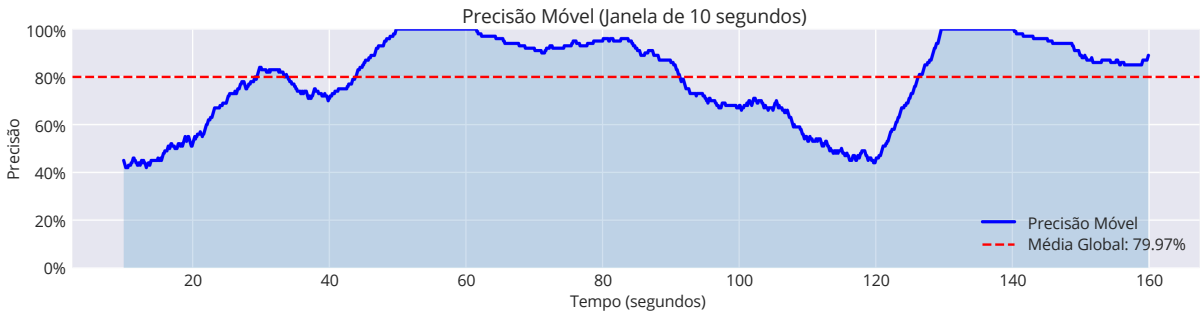
**Figura 6.22** – *Precisão por Nível de Cavitação*




**Fonte:** Autoria Própria.

A análise temporal do sistema pode ser observada através da Tabela 6.33 e da Figura 6.23, que apresentam o comportamento das classificações ao longo do período de teste.

**Figura 6.23** – *Precisão por Nível de Cavitação*




**Fonte:** Autoria Própria.

A distribuição geral dos níveis de cavitação, apresentada na Figura 6.24, demonstra uma predominância do Nível 2, seguido pelos Níveis 4, 3 e 1, respectiva-

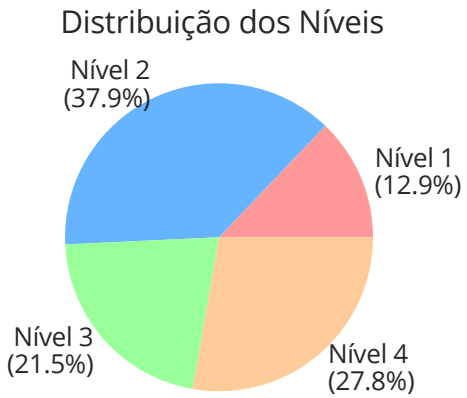
**Tabela 6.33** – *Análise de Classificação por Segmento Temporal*

| Segmento | Período (s) | Classificações Corretas | Classificações Incorretas |
|---|---|---|---|
| 1 | 0.0-20.0 (Nível 1) | 96 | 104 |
| 2 | 20.0-40.0 (Nível 3) | 155 | 45 |
| 3 | 40.0-60.0 (Nível 4) | 200 | 0 |
| 4 | 60.0-80.0 (Nível 2) | 187 | 13 |
| 5 | 80.0-100.0 (Nível 3) | 154 | 46 |
| 6 | 100.0-120.0 (Nível 1) | 89 | 111 |
| 7 | 120.0-140.0 (Nível 4) | 200 | 0 |
| 8 | 140.0-160.0 (Nível 2) | 178 | 22 |

**Fonte:** Autoria Própria.

mente. Esta distribuição não uniforme reflete as características reais do sistema em operação.

**Figura 6.24** – *Distribuição dos Níveis de Cavitação*




**Fonte:** Autoria Própria.

A análise integrada dos resultados revela padrões interessantes: o Nível 4 apresentou classificação perfeita em seus segmentos (3 e 7), como pode ser observado na Figura 6.23. O principal desafio do modelo foi a tendência em classificar erroneamente o Nível 1 como Nível 2, evidenciado pelos períodos de 0-20s e 100-120s, onde há uma predominância de marcadores vermelhos (X). Entretanto, o modelo manteve alta precisão na identificação dos Níveis 3 e 4, demonstrando sua adequação para detecção de estados mais críticos de cavitação. Esta variação na precisão entre os níveis pode estar relacionada às características específicas de cada estado de cavitação e às diferentes condições de iluminação e operação durante a captura dos vídeos, resultando em uma média global de precisão de 79,97% nos dados de teste.

## 6.6 Interface Web para Monitoramento Remoto em Tempo Real

A interface web desenvolvida permite o monitoramento em tempo real dos níveis de cavitação, como apresentado na Figura 6.25. O sistema exibe a imagem atual do escoamento na parte superior, seguida pelo nível de cavitação detectado, precisão da classificação e horário da análise.

**Figura 6.25** – *Interface Web para Monitoramento de Cavitação*


**Fonte:** Autoria Própria.

A interface apresenta um design funcional, com elementos visuais que facilitam a interpretação dos resultados. O nível de cavitação é exibido em destaque, utilizando um código de cores que varia de acordo com a severidade (azul para Níveis 1,2,3 e vermelho para Nível 4). A precisão da classificação é apresentada em porcentagem, permitindo avaliar a confiabilidade da detecção em tempo real.

O sistema inclui uma seção de informações adicionais que contextualiza a aplicação, explicando sua importância para o monitoramento de sistemas hidráulicos e prevenção de danos em equipamentos. A interface foi desenvolvida utilizando *HTML* e o *framework Flask*, garantindo responsividade e facilidade de acesso através de diferentes dispositivos.

# Capítulo 7

# Considerações Finais

Este trabalho demonstrou que técnicas de visão computacional e redes neurais convolucionais (CNNs), são eficientes para detecção e classificação de cavitação em sistemas de bombeamento. A combinação de aprendizado supervisionado e não supervisionado, com o filtro *Prewitt*, permitiu identificar e categorizar padrões visuais de cavitação. A metodologia híbrida, integrando clusterização por *Gaussian Mixture Models* (GMM) e treinamento supervisionado, foi robusta, especialmente com a *VGG19* para extração de características, alcançando coeficientes de silhueta de até 0,2547 com o filtro *Prewitt*. Desenvolveu-se uma interface web intuitiva para monitoramento em tempo real, avançando em soluções práticas e acessíveis para ambientes industriais.

Os resultados mostraram um compromisso entre precisão do treinamento e consumo de recursos computacionais. A *ResNet101* teve maior acurácia no treinamento (91% contra 88% da *VGG19*), mas a *VGG19* foi superior em eficiência de memória e escalabilidade, devido à sua arquitetura mais simples, que demanda menos memória *RAM*. As arquiteturas *ResNet* atingiam rapidamente o limite de memória ao processar grandes volumes de dados, resultando em falhas. A *VGG16* e, especialmente, a *VGG19* processaram o conjunto completo de dados (19.200 imagens), mantendo o consumo de memória controlado. Nos testes de validação, a *VGG19* alcançou uma precisão média de 79,97%, demonstrando robustez e capacidade de generalização, mesmo com acurácia de treinamento ligeiramente inferior. Essa característica, associada à eficiência no uso de memória, sugere que a *VGG19* é uma alternativa mais adequada para sistemas industriais de monitoramento contínuo, onde estabilidade e operação com recursos limitados são cruciais.

# Capítulo 8

# Sugestões para Trabalhos Futuros

Com base nos resultados e nas limitações encontradas neste trabalho, sugere-se as seguintes direções para pesquisas futuras:

- Realizar a clusterização em computadores com maior capacidade de memória RAM, permitindo a análise completa do *dataset* com as arquiteturas *ResNet*.
- Conduzir o treinamento supervisionado com as arquiteturas *ResNet50* e VGG16, utilizando o conjunto completo de dados para comparação direta.
- Efetuar testes de desempenho em tempo real com todas as arquiteturas (*ResNet50*, *ResNet101*, VGG16 e VGG19) em condições variadas.
- Investigar a incorporação dos modelos de detecção de cavitação em sistemas embarcados, visando aplicações industriais de monitoramento.
- Aprimorar a interface web, incluindo funcionalidades adicionais, como gráficos históricos e alertas configuráveis, melhorando sua usabilidade.
- Expandir o *dataset*, incluindo imagens com diferentes condições de iluminação e tipos de bombas, para aumentar a generalização.
- Avaliar o uso de outras técnicas de pré-processamento de imagens, como diferentes filtros e métodos de realce, para otimizar a extração.

# REFERÊNCIAS


AGGARWAL, C. C. *Neural Networks and Deep Learning - A Textbook*. Springer, 2018. 12 – 14, 21 – 24, 63– 65, 116 – 120, 315-321 p. ISBN 978-3-319-94462-3. Disponível em: <https://doi.org/10.1007/978-3-319-94463-0>. ( Citado 5 vezes nas páginas 13, 14, 16, 18, and 19. )

AHMED, A. Comparative study among sobel, prewitt and canny edge detection operators used in image processing. *Journal of Theoretical and Applied Information Technology*, v. 96, p. 9, 10 2018. ( Citado na página 19. )

ALI, A. et al. Transfer learning: A new promising techniques. *Mesopotamian Journal of Big Data*, v. 2023, p. 31–32, 02 2023. ( Citado na página 26. )

BEN-HUR, A.; GUYON, I. Detecting stable clusters using principal component analysis. *Methods in molecular biology (Clifton, N.J.)*, v. 224, p. 159–82, 02 2003. ( Citado 2 vezes nas páginas 24 and 25. )

BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. 21–25, 110 - 113, 423–429, 560 – 583 p. ISBN 0387310738. ( Citado 3 vezes nas páginas 21, 22, and 25. )

BRENNEN, C. E. *Cavitation and Bubble Dynamics*. New York: Oxford University Press, 1995. 1-15, 34-47, 100-125 p. ISBN 0-19-509409-3. Disponível em: <https://www.researchgate.net/publication/36721045_Cavitation_and_Bubble_Dynamics>. ( Citado na página 6. )

BRENNEN, C. E. An introduction to cavitation fundamentals. *Proceedings of WIMRC Forum 2011*, California Institute of Technology, p. 1–12, 2011. Invited Lecture, University of Warwick, UK. Disponível em: <https://www.researchgate.net/publication/279685077_An_Introduction_to_Cavitation_Fundamentals>. ( Citado 3 vezes nas páginas 1, 4, and 6. )

BRUNETTI, F. *Mecânica dos Fluidos*. 2ª edição revisada. ed. São Paulo: Pearson Prentice Hall, 2008. 8-9, 18-24, 125-136 p. Livro dedicado à compreensão da mecânica dos fluidos com exercícios resolvidos e aplicações práticas. ISBN 978-85-7605-182-4. ( Citado 2 vezes nas páginas 4 and 5. )

CHOLLET, F. *Deep Learning with Python*. [S.l.]: Manning, 2017. 204, 205, 255 p. ISBN 9781617294433. ( Citado na página 15. )

ELLENFELD, M. et al. Deep fusion of appearance and frame differencing for motion segmentation. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. [S.l.: s.n.], 2021. p. 4334–4344. ( Citado na página 11. )

ESCALER, X. et al. Detection of cavitation in hydraulic turbines. *Mechanical Systems and Signal Processing*, v. 20, n. 4, p. 983–1007, 2006. ISSN 0888-3270. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0888327004001475>. ( Citado na página 8. )

EVERITT, B. et al. *Cluster Analysis, 5th Edition*. John Wiley & Sons, 2011. 15 –20,43,49,143, 426 – 430 p. Disponível em: <https://books.google.com.br/books?id=3RFPzQEACAAJ>. ( Citado 3 vezes nas páginas 21, 22, and 24. )

FILHO, O. M.; NETO, H. V. *Processamento Digital de Imagens*. Rio de Janeiro: Brasport, 1999. 37–38, 83–95 p. ISBN 8574520098. ( Citado 2 vezes nas páginas 19 and 20. )

GAISSER, L.; KIRSCHNER, O.; RIEDELBAUCH, S. Cavitation detection in hydraulic machinery by analyzing acoustic emissions under strong domain shifts using neural networks. *Physics of Fluids*, AIP Publishing LLC, v. 35, n. 2, p. 027128, 2023. ( Citado na página 1. )

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. Upper Saddle River, N.J.: Prentice Hall, 2008. 120–135, 723–726 p. ISBN 9780131687288 013168728X 9780135052679 013505267X. ( Citado 2 vezes nas páginas 19 and 20. )

GURNEY, K. *An Introduction to Neural Networks*. [S.l.]: Taylor & Francis, Inc., 1997. 24 – 33 p. ISBN 1857286731. ( Citado na página 14. )

GéRON, A. *Hands-on machine learning with Scikit-Learn and TensorFlow*. Sebastopol, CA: O'Reilly Media, 2017. 222 – 231, 240– 245, 256 – 261, 331, 357– 359, 434, 437 – 438 p. ISBN 978-1491962299. ( Citado 6 vezes nas páginas 15, 16, 21, 24, 25, and 29. )

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2011. 99 – 106, 487 – 491 p. ISBN 0123814790, 9780123814791. ( Citado na página 24. )

HAYKIN, S. S. *Neural networks and learning machines*. Third. [S.l.]: Pearson Education, 2009. 23 – 26, 85,224 p. ( Citado 2 vezes nas páginas 13 and 14. )

HE, K. et al. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. Disponível em: <http://arxiv.org/abs/1512.03385>. ( Citado na página 18. )

IBRAHIM, M. *A Deep Dive Into Learning Curves in Machine Learning*. 2023. Accessed: 2025-01-15. Disponível em: <https://shre.ink/bvwO>. ( Citado na página 32. )

JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, v. 31, n. 8, p. 651–666, 2010. ISSN 0167-8655. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167865509002323>. ( Citado na página 9. )

JU, H.-j.; CHOI, J.-s. Experimental study of cavitation damage to marine propellers based on the rotational speed in the coastal waters. *Machines*, v. 10, p. 793, 09 2022. ( Citado 2 vezes nas páginas 2 and 3. )

KRELLA, A. K. Degradation and protection of materials from cavitation erosion: A review. *Materials*, v. 16, n. 5, 2023. ISSN 1996-1944. Disponível em: <https://www.mdpi.com/1996-1944/16/5/2058>. ( Citado na página 8. )

KULKARNI, A.; CHONG, D.; BATARSEH, F. A. 5 - foundations of data imbalance and solutions for a data democracy. In: BATARSEH, F. A.; YANG, R. (Ed.). *Data Democracy*. [S.l.]: Academic Press, 2020. p. 83–106. ISBN 978-0-12-818366-3. ( Citado na página 31. )

LIANG, J. Image classification based on resnet. *Journal of Physics: Conference Series*, v. 1634, p. 012110, 09 2020. ( Citado na página 18. )

LibreTexts Team. *Bernoulli's Equation*. 2024. Acessado em: 14 de dezembro de 2024. Disponível em: <https://phys.libretexts.org/Bookshelves/University_Physics/University_Physics_(OpenStax)/Book%3A_University_Physics_I_-_Mechanics_Sound_Oscillations_and_Waves_(OpenStax)/14%3A_Fluid_Mechanics/14.08%3A_Bernoullis_Equation>. ( Citado na página 6. )

LIU, L. et al. Experimental study of surface damage of stainless steel subjected to cavitation collapse in aqueous environment. *Coatings*, v. 13, p. 1356, 08 2023. ( Citado na página 2. )

LU, W. et al. *An Efficient Gaussian Mixture Model and Its Application to Neural Network*. 2024. ( Citado na página 23. )

LUFF, P.; HEATH, C. Some 'technical challenges' of video analysis: social actions, objects, material realities and the problems of perspective. *Qualitative Research*, v. 12, n. 3, p. 255–279, 2012. ( Citado na página 12. )

MANJU, A.; VALARMATHIE, P. Análise de vídeo para extração de substância semântica usando OpenCV em python. *Journal of Ambient Intelligence and Humanized Computing*, v. 12, p. 4057–4066, 2021. ( Citado na página 13. )

MUNSON, B. R.; YOUNG, D. F.; OKIISHI, T. H. *Fundamentos da Mecânica dos Fluidos*. 1ª edição. ed. [S.l.]: Editora Edgard Blücher, 2004. 141-156, 223-245, 85-107 p. ISBN 85-212-0343-8. ( Citado 2 vezes nas páginas 4 and 5. )

OpenCV Team. *OpenCV*. 2024. Acesso em: 27 dez. 2024. Disponível em: <https://opencv.org>. ( Citado na página 13. )

PATIL, H.; MAHANDULE, V.; GUNJAL, A. Python in the evolution of ai: A comparative study of emerging technologies. SSRN, November 2024. Disponível em: <https://ssrn.com/abstract=5075929>. ( Citado na página 39. )

PRINCE, S. J. *Understanding Deep Learning*. [S.l.]: MIT Press, 2023. 147 – 147 p. ( Citado na página 30. )

RIBANI, R.; MARENGONI, M. A survey of transfer learning for convolutional neural networks. In: *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*. [S.l.: s.n.], 2019. p. 47–57. ( Citado 2 vezes nas páginas 26 and 27. )

ROSEBROCK, A. *Deep Learning for Computer Vision with Python: Starter Bundle*. PyImageSearch, 2017. 21 – 27, 29 – 34, 42-46, 76–86 p. Disponível em: <https://books.google.com.br/books?id=9Ul-tgEACAAJ>. ( Citado 2 vezes nas páginas 27 and 28. )

ROUSSEAU, F.; DRUMETZ, L.; FABLET, R. Residual networks as flows of diffeomorphisms. *Journal of Mathematical Imaging and Vision*, v. 62, 04 2020. ( Citado na página 18. )

SHADEED, G.; TAWFEEQ, M.; MAHMOUD, S. Automatic medical images segmentation based on deep learning networks. *IOP Conference Series: Materials Science and Engineering*, v. 870, p. 012117, 07 2020. ( Citado na página 17. )

SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. Disponível em: <httpsarxiv.orgabs1409.1556>. ( Citado na página 17. )

SOARES, P.; SILVA, J. da. Aplicação de redes neurais artificiais em conjunto com o método vetorial da propagação de feixes na análise de um acoplador direcional baseado em fibra Ótica. *Revista Brasileira de Computação Aplicada*, v. 3, 12 2011. ( Citado na página 13. )

STUDIO, S. *Water Pressure Sensor G1/4 1.2Mpa*. [S.l.], 2017. SKU 114991178. Disponível em: <https://www.seeedstudio.com/Water-Pressure-Sensor-G1%2F4-1.2MPa-p-2887.html>. ( Citado na página 35. )

SUPPONEN, O. et al. Detailed jet dynamics in a collapsing bubble. *Journal of Physics: Conference Series*, v. 656, n. 1, p. 012038, 2015. ( Citado na página 7. )

TEAM, K. *Transfer learning & fine-tuning - Keras Documentation*. 2015. <https://keras.io/guides/transfer_learning/>. Acessado em: 10/01/2025. ( Citado na página 28. )

TONG, Z. et al. Cavitation diagnosis for water distribution pumps: An early-stage approach combing vibration signal-based neural network with high-speed photography. *Sustainable Energy Technologies and Assessments*, v. 55, p. 102919, 2023. ISSN 2213-1388. ( Citado na página 9. )

VIERING, T. J.; LOOG, M. The shape of learning curves: a review. *CoRR*, abs/2103.10948, 2021. Disponível em: <https://arxiv.org/abs/2103.10948>. ( Citado na página 32. )

VISA, S. et al. Confusion matrix-based feature selection. In: . [S.l.: s.n.], 2011. v. 710, p. 120–127. ( Citado 2 vezes nas páginas 30 and 31. )

ZHANG, A. et al. Dive into deep learning. *CoRR*, abs/2106.11342, p. 194 – 198, 468 – 472, 493 – 499, 532 – 536, 2021. Disponível em: <https://arxiv.org/abs/2106.11342>. ( Citado 2 vezes nas páginas 29 and 30. )

# APÊNDICE A

# Códigos em *Python*

## A.1 Utilizando o capítulo de Apêndices

**Código A.1** – *Exemplo de Processamento Genérico de Filtro em Vídeos*

```
1
2 import cv2, numpy as np, os
3
4 # Escolha um filtro de borda para imagem
5 def aplicar_filtro(img, kernel):
6     return cv2.filter2D(img, -1, kernel)
7
8 # Extrai frames, corta, aplica filtro e salva
9 def processar_video(caminho, area_corte, pasta, id_video,
10 filtro, duracao=40):
11     cap = cv2.VideoCapture(caminho)
12     fps = cap.get(cv2.CAP_PROP_FPS)
13     total_frames = int(fps * duracao)
14
15     for i in range(1, total_frames + 1):
16         ret, frame = cap.read()
17         if not ret: break
18
19         cortado = frame[area_corte[0]:area_corte[1],
20         area_corte[2]:area_corte[3]]
21         imagem_filtrada = aplicar_filtro(cortado, filtro)
22         cv2.imwrite(os.path.join(pasta, f"{id_video}_{i:04d}
23         .png"), imagem_filtrada)
24     cap.release()
25
26 # Configurações
27 area_corte = (425, 520, 100, 1600)
```

```
pasta_saida = 'frames_processados'
os.makedirs(pasta_saida, exist_ok=True)

# Defina um filtro (ex: Prewitt, Sobel, etc.)
filtro_prewitt_x = np.array([[1, 0, -1], [1, 0, -1],
[1, 0, -1]])

# Lista de videos
videos = [os.path.join('videos', f) for f in ['video1.mp4',
'video2.mp4']]

# Processar cada video
for video in videos:
    id_video = int(os.path.basename(video).split('_')[0][1:])
    processar_video(video, area_corte, pasta_saida,
    id_video, filtro_prewitt_x)
```

## A.2 Clusterização de Imagens Utilizando Arquiteturas de Redes Neurais

**Código A.2** – *Código Python para Clusterização Simplificada*

```


# Configurações Iniciais
diretorio_saida = '0.25_ResNet50_Sobel_Simplificado'
diretorio_frames = 'C:/.../frames_0.25_Sobel' #Local frames
numero_imagens = 4800 # Número de imagens a serem usadas
numero_componentes_pca = 30  # Componentes PCA
numero_clusters = 4  # Define o número de clusters
os.environ['OMP_NUM_THREADS'] = '4' # Otimizar uso de CPU

# Criar diretório de saída
if not os.path.exists(diretorio_saida):
    os.makedirs(diretorio_saida)

# Carregar modelo a ser utilizado
modelo = ResNet50(weights='imagenet', include_top=False)

# Função para carregar e pré-processar imagens
def carregar_e_processar_imagem(local_imagem):
    img = image.load_img(local_imagem, target_size=(224, 224))
    img_array = image.img_to_array(img)
```

```
22      img_array = np.expand_dims(img_array, axis=0)
23      return preprocess_input(img_array)
24
25  # Selecionar frames aleatoriamente
26  nomes_frames = os.listdir(diretorio_frames)
27  frames_selecionados = random.sample(nomes_frames,
28  numero_imagens)
29
30  # Extrair features das imagens usando o ResNet50
31  lista_features = []
32  caminhos_imagens = [] # Guarda o caminho das imagens
33
34  inicio = time.time() # Marca o tempo de início do processo
35
36  for nome_frame in frames_selecionados:
37      caminho_frame = os.path.join(diretorio_frames,
38  nome_frame)
39      if os.path.isfile(caminho_frame):
40          img_processada = carregar_e_processar_imagem(caminho_
41  frame)
42          features = modelo.predict(img_processada)
43          lista_features.append(features.flatten())
44          caminhos_imagens.append(caminho_frame)
45
46  # Converter features para array numpy e normalizar
47  array_features = np.array(lista_features)
48  normalizador = StandardScaler()
49  array_features = normalizador.fit_transform(array_features)
50
51  # Aplicar PCA para reduzir a dimensionalidade
52  pca = PCA(n_components=numero_componentes_pca)
53  features_pca = pca.fit_transform(array_features)
54
55  # Aplicar Gaussian Mixture Model para clusterização
56  gmm = GaussianMixture(n_components=numero_clusters,
57  random_state=42)
58  clusters = gmm.fit_predict(features_pca)
59
60  tempo_total = time.time() - inicio # Calcula o tempo total
61  silhueta = silhouette_score(features_pca, clusters)#Silhueta
62
```

## A.3 Código Treinamento Supervisionado

**Código A.3** – *Código Python Simplificado para Treinamento para diferentes arquiteturas*

```
1
2 from google.colab import drive
3
4 # Montar Google Drive
5 drive.mount('/content/drive', force_remount=True)
6
7 # Diretórios
8 result_dir = Path('/content/drive/MyDrive/VGG19')
9 result_dir.mkdir(parents=True, exist_ok=True)
10 train_dir = Path('/content/treinamento')
11 val_dir   = Path('/content/Validacao')
12
13 # Data Augmentation para treinamento e validação
14 train_aug = ImageDataGenerator(rescale=1./255,
15 horizontal_flip=True)
16 val_aug   = ImageDataGenerator(rescale=1./255,
17             horizontal_flip=True,
18                                rotation_range=10,
19                                width_shift_range=0.1,
20                                height_shift_range=0.1)
21
22 # Geradores de dados
23 train_gen = train_aug.flow_from_directory(
24     train_dir, target_size=(224,224), batch_size=32,
25     class_mode='categorical', shuffle=True)
26 val_gen = val_aug.flow_from_directory(
27     val_dir, target_size=(224,224), batch_size=32,
28     class_mode='categorical', shuffle=True)
29
30 # Modelo base VGG19 (camadas congeladas)
31 base_model = VGG19(weights='imagenet', include_top=False,
32 input_shape=(224,224,3))
33 for layer in base_model.layers:
34     layer.trainable = False
35
36 # Construção do modelo final
37 modelo = models.Sequential([
38     base_model,
39     layers.GlobalAveragePooling2D(),
40     layers.Dropout(0.5),
41     layers.Dense(train_gen.num_classes, activation='softmax')
42 ])
```

```
43
44 # Compilação e treinamento
45 modelo.compile(optimizer=Adam(learning_rate=0.00055),
46                 loss='categorical_crossentropy',
47                 metrics=['accuracy'])
48 historico = modelo.fit(
49     train_gen, validation_data=val_gen, epochs=50,
50     steps_per_epoch=train_gen.samples
51 //train_gen.batch_size,
52     validation_steps=val_gen.samples
53 //val_gen.batch_size, verbose=1)
54
55 # Salvar o modelo treinado
56 try:
57     modelo.save(result_dir / 'modelo_finalVGG19.h5')
58     print("Modelo salvo com sucesso!")
59 except Exception as e:
60     print(f"Erro ao salvar o modelo: {e}")
```

## A.4 Código para Tempo Real e Interface

**Código A.4** – *Código Flask Simplificado para Tempo Real e Interface*

```
1
2 app = Flask(__name__)
3
4 # Carregar o modelo ja treinado do codigo anterior
5 model = keras.models.load_model('C:/Users/Admin/modelo.h5')
6 labels = ['nivel 1', 'nivel 2', 'nivel 3', 'nivel 4']
7 current_level, current_acc = "N/A", 0.0
8
9 #Processamento frames novos
10 def prep(frame):
11     return np.expand_dims(cv2.resize(frame, (224,224)).
12 astype('float32')/255.0, axis=0)
13
14 #Definição de filtros que vao ser ultilizados
15 def prewitt(frame):
16     kx = np.array([[1,0,-1]]*3, np.float32)
17     ky = np.array([[1,1,1],[0,0,0],[-1,-1,-1]], np.float32)
18     return cv2.addWeighted(cv2.filter2D(frame,-1,kx),0.5,
19                            cv2.filter2D(frame,-1,ky),0.5,0)
20
```

```
def gen_frames(path):
    global level, acc
    cap = cv2.VideoCapture(path)
    delay = 1/(cap.get(cv2.CAP_PROP_FPS) or 30)
    while True:
        start = time.time()
        ret, frame = cap.read()
        if not ret: break
        f = prewitt(frame)
        p = prep(f)
        pred = model.predict(p, verbose=0)[0]
        i = np.argmax(pred)
        level, acc = labels[i], pred[i]
        _, buf = cv2.imencode('.jpg', f)
        yield (b'--frame\r\nContent-Type: image/jpeg\r\n\r\n'
+ buf.tobytes() + b'\r\n')
        time.sleep(max(delay - (time.time()-start), 0))

@app.route('/')
def idx():
    return "Index"

@app.route('/video_feed')
def feed():
    path = 'C:/Users/Admin/Desktop/TCC/output_video.mp4'
    return Response(gen_frames(path),
                    mimetype='multipart/x-mixed-replace;
                    boundary=frame')

@app.route('/info')
def info():
    return jsonify({'level': level, 'accuracy': acc*100})

if __name__ == '__main__':
    app.run(debug=True, port=8080)
```

## A.5 Vídeo de Funcionamento do Projeto

Link para o vídeo: <https://drive.google.com/file/d/1Pn6DcvCeYbbl-0tK8Y8ILADe1pft1N
view?usp=sharing>

Caso o Vídeo não estiver disponível entre em contato com o email: <Jonassouza871@
hotmail.com>