



UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO PROGRAMA
DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO



JOSÉ GIDEL OLIVEIRA SARAIVA

GERAÇÃO DE IMAGENS SINTÉTICAS COM REDE ADVERSARIAL
GENERATIVA BASEADA EM SENSOR DE RESSONÂNCIA DE PLASMON DE
SUPERFÍCIE

MOSSORÓ

2025

JOSÉ GIDEL OLIVEIRA SARAIVA

**GERAÇÃO DE IMAGENS SINTÉTICA COM REDE ADVERSARIAL GENERATIVA
BASEADA EM SENSOR DE RESSONÂNCIA DE PLASMON DE SUPERFÍCIE**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação – associação ampla entre a Universidade Estadual do Rio Grande do Norte e a Universidade Federal Rural do Semiárido, como requisito parcial para obtenção do título de Mestre.

Linha de Pesquisa: Engenharia de Software e Sistemas Computacionais

Orientador: Dr. Silvio Roberto Fernandes de Araújo - UFERSA

Co-orientador: Dr. Leiva Casemiro Oliveira - UFERSA

MOSSORÓ

2025

© Todos os direitos estão reservados a Universidade Federal Rural do Semi-Árido. O conteúdo desta obra é de inteira responsabilidade do (a) autor (a), sendo o mesmo, passível de sanções administrativas ou penais, caso sejam infringidas as leis que regulamentam a Propriedade Intelectual, respectivamente, Patentes: Lei nº 9.279/1996 e Direitos Autorais: Lei nº 9.610/1998. O conteúdo desta obra tomar-se-á de domínio público após a data de defesa e homologação da sua respectiva ata. A mesma poderá servir de base literária para novas pesquisas, desde que a obra e seu (a) respectivo (a) autor (a) sejam devidamente citados e mencionados os seus créditos bibliográficos.

S48g Saraiva, José Gidel Oliveira.
Geração de imagens sintética com rede
adversarial generativa baseada em sensor de
ressonância de plasmon de superfície / José Gidel
Oliveira Saraiva. - 2025.
129 f. : il.

Orientador: Silvio Roberto Fernandes de Araújo.
Coorientadora: Leiva Casemiro Oliveira.
Dissertação (Mestrado) - Universidade Federal
Rural do Semi-árido, Programa de Pós-graduação em
Ciência da Computação, 2025.

1. Inteligência artificial. 2. Redes
adversariais generativas. 3. Espaço latente. 4.
Ressonância de plásmo de superfície. 5. Sensores
SPR e processamento de imagens. I. Araújo, Silvio
Roberto Fernandes de, orient. II. Oliveira, Leiva
Casemiro, co-orient. III. Título.

Ficha catalográfica elaborada por sistema gerador automático em conformidade
com AACR2 e os dados fornecidos pelo(a) autor(a).
Biblioteca Campus Mossoró / Setor de Informação e Referência
Bibliotecária: Keina Cristina Santos Sousa e Silva
CRB: 15/120

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC's) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecas da Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

JOSÉ GIDEL OLIVEIRA SARAIVA

**GERAÇÃO DE IMAGENS SINTÉTICA COM REDE ADVERSARIAL GENERATIVA
BASEADA EM SENSOR DE RESSONÂNCIA DE PLASMON DE SUPERFÍCIE**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação – associação ampla entre a Universidade Estadual do Rio Grande do Norte e a Universidade Federal Rural do Semiárido, como requisito parcial para obtenção do título de Mestre.

Linha de Pesquisa: Engenharia de Software e Sistemas Computacionais

Defendida em: 29/08/2025

BANCA EXAMINADORA

Silvio Roberto Fernandes de Araújo, Prof. Dr. (UFERSA)
Presidente

Leiva Casemiro Oliveira, Prof. Dr. (UFERSA)
Membro Examinador

Sebastião Emidio Alves Filho, Prof. Dr. (UERN)
Membro Examinador

Antonio Marcus Nogueira Lima, Prof. Dr. (UFCEG)
Membro Examinador

RESUMO

A inteligência artificial, por meio de modelos generativos como as redes adversárias generativas (em inglês, *Generative Adversarial Networks* – GANs), tem sido aplicada para superar a escassez de dados em técnicas sensíveis como a ressonância de plásmons de superfície (em inglês, *Surface Plasmon Resonance* – SPR) para aplicações científicas, nas quais custos elevados e limitações experimentais são obstáculos frequentes. Dentre os modelos generativos investigados na literatura, as GANs vêm sendo amplamente adotadas para geração de imagens sintéticas realistas devido à sua capacidade de equilibrar qualidade de saída e custo computacional. Este trabalho emprega uma rede generativa adversarial convolucional profunda (*Deep Convolutional Generative Adversarial Network* – DCGAN) para gerar imagens SPR sintéticas realistas. O modelo foi treinado com banco de dados consolidado da literatura, simulando sensores ópticos na configuração de Kretschmann com prisma polimérico para aplicações biológicas (PPBIO). A avaliação do modelo combinou métricas quantitativas e qualitativas para verificar a qualidade e a fidelidade das imagens sintéticas geradas. No aspecto quantitativo, foram consideradas as seguintes métricas: FWHM, largura total, energia normalizada, funções de perda do gerador e discriminador, maximum mean discrepancy (MMD), erro quadrático médio (MSE) e distância euclidiana. Para a avaliação qualitativa, foram utilizadas visualizações comparativas e projeções em espaço reduzido por meio do t-SNE e histogramas. A contribuição central deste trabalho tem dois aspectos: a demonstração da viabilidade de GANs para gerar imagens SPR com sensores ópticos na configuração PPBIO, e uma análise custo-benefício comparando dados sintéticos e reais na tarefa de identificação de substâncias via CNNs. A integração desses resultados representa uma inovação significativa para aplicações científicas dependentes de sistemas SPR, oferecendo uma solução que alia precisão técnica à viabilidade econômica.

Palavras-chave: inteligência artificial, redes adversariais generativas, espaço latente, ressonância de plásmon de superfície, sensores SPR e processamento de imagens.

ABSTRACT

Artificial intelligence, through generative models such as Generative Adversarial Networks (GANs), has been applied to overcome data scarcity in sensitive techniques like Surface Plasmon Resonance (SPR) for scientific applications, where high costs and experimental limitations are frequent obstacles. Among the generative models investigated in the literature, GANs have been widely adopted for generating realistic synthetic images due to their ability to balance output quality and computational cost. This work employs a Deep Convolutional Generative Adversarial Network (DCGAN) to generate realistic synthetic SPR images. The model was trained using a consolidated database from the literature, simulating optical sensors in the Kretschmann configuration with a Polymeric Prism for Biological Applications (PPBIO). Model evaluation combined quantitative and qualitative metrics to assess the quality and fidelity of the generated synthetic images. On the quantitative side, the following metrics were considered: FWHM, full width, normalized energy, asymmetry, minimum position, generator and discriminator loss functions, Maximum Mean Discrepancy (MMD), Mean Squared Error (MSE), and Euclidean Distance. For qualitative evaluation, comparative visualizations and reduced-space projections using t-SNE and histograms were used. The main contribution of this work has two aspects: demonstrating the feasibility of GANs to generate SPR images with optical sensors in the PPBIO configuration, and a cost-benefit analysis comparing synthetic and real data in the task of substance identification using CNNs. The integration of these results represents a significant innovation for scientific applications dependent on SPR systems, offering a solution that combines technical accuracy with economic feasibility.

Keywords: artificial intelligence, generative adversarial networks, latent space, surface plasmon resonance, spr sensors, image processing.

LISTA DE FIGURAS

Figura 1 – Visão geral da Inteligência Artificial.....	18
Figura 2 – Os principais componentes de um neurônio artificial.....	20
Figura 3 – Visão geral de uma rede neural artificial.....	21
Figura 4 – Ilustração do treinamento da GAN.....	23
Figura 5 – ilustra o treinamento da CGAN.....	28
Figura 6 – ilustra o treinamento da WGAN.....	30
Figura 7 – Camadas de rede do gerador DCGAN.....	33
Figura 8 – Camadas de rede do discriminador DCGAN.....	34
Figura 9 – Fluxograma do treinamento da DCGAN.....	34
Figura 10 - Funcionamento de um sensor SPR: esquema óptico, imagem experimental e extração de curva de ressonância.....	36
Figura 11 - Esquema óptico e montagem física do sistema do SPR PPBIO com interrogação angular (AIM).....	38
Figura 12 – Curva SPR com indicação de CR, CL, FWHM e ponto mínimo, acompanhada da imagem real.....	40
Figura 13 – Geração de imagens usando operações aritméticas em GAN.....	42
Figura 14 - Interpolação linear no espaço latente do StyleGAN.....	44
Figura 15 - Interpolação slerp no espaço latente do styleGAN.....	45
Figura 16 - Fluxograma da metodologia deste trabalho.....	49
Figura 17 – Imagens do sensor PPBIO	51
Figura 18 – Imagens da base de dados.....	52
Figura 19 - Resumo detalhado da arquitetura do discriminador.....	57
Figura 20 - Resumo detalhado da arquitetura do gerador.....	59
Figura 21 - Função de criação de pontos no espaço latente.....	59
Figura 22 - Geração de rótulos com ou sem suavização para o discriminador.....	60
Figura 23 - Inicialização de pesos para camadas convolucionais e de normalizações.....	60
Figura 24 - Visualização de imagens em escala de cinza.....	61
Figura 25 - Visualização em matriz de imagens escala de cinza.....	61
Figura 26 - Função para mostrar a perda do gerador e do discriminador durante o treinamento.....	62
Figura 27 - Procedimento de avaliação do modelo gerador com diferentes modos de amostragem latente.....	62
Figura 28 - Função para construir uma imagem sintética baseado na imagem alvo.....	64
Figura 29 - Soma de características de vetores latentes.....	65
Figura 30 - Interpolação linear e esférica entre vetores latentes.....	65
Figura 31 - transição de um vetor latente inicial para uma imagem alvo.....	66
Figura 32 - apresenta a evolução das perdas do discriminador e do gerador.....	78
Figura 33 - Comparação entre imagens reais e geradas da classe 1.33.....	79

Figura 34 – Comparação entre imagens reais e geradas da classe 1.335.....	80
Figura 35 – Comparação entre imagens reais e geradas da classe 1.34.....	80
Figura 36 – Comparação entre imagens reais e geradas da classe 1.345.....	80
Figura 37 – Comparação entre imagens reais e geradas da classe 1.35.....	81
Figura 38 - Comparação de histogramas médios 60 imagens reais e 60 geradas.....	82
Figura 39 – Perplexidade vs Divergência KL.....	84
Figura 40 - Exibição visual dos componentes do t-SNE para imagens reais e geradas.....	85
Figura 41 - Curvas de intensidade extraídas de 12 imagens geradas da classe 1.33.....	86
Figura 42 – Curvas de intensidade extraídas de 12 imagens geradas da classe 1.335.....	87
Figura 43 – Curvas de intensidade extraídas de 12 imagens geradas da classe 1.34.....	87
Figura 44 – Curvas de intensidade extraídas de 12 imagens geradas da classe 1.345.....	88
Figura 45 – Curvas de intensidade extraídas de 12 imagens geradas da classe 1.35.....	88
Figura 46 - Comparativo das curvas extraídas das imagens, comparação das imagens original e gerada e comparação dos parâmetros da classe 1.33.....	90
Figura 47 - Comparativo das curvas extraídas das imagens, comparação das imagens original e gerada e comparação dos parâmetros da classe 1.335.....	91
Figura 48 - Comparativo das curvas extraídas das imagens, comparação das imagens original e gerada e comparação dos parâmetros da classe 1.34.....	91
Figura 49 - Comparativo das curvas extraídas das imagens, comparação das imagens original e gerada e comparação dos parâmetros da classe 1.345.....	92
Figura 50 - Comparativo das curvas extraídas das imagens, comparação das imagens original e gerada e comparação dos parâmetros da classe 1.35.....	93
Figura 51 - Apresenta a adição de dois vetores latentes das classes 1.33 e 1.335.....	95
Figura 53 - apresenta a adição de dois vetores latentes das classes 1.335 e 1.34.....	96
Figura 55 - apresenta a subtração de dois vetores latentes das classes 1.33 e 1.335.....	97
Figura 56 - Análise das curvas de intensidade luminosa das três imagens.....	98
Figura 57 - apresenta a subtração de dois vetores latentes das classes 1.335 e 1.34.....	98
Figura 59 - interpolação linear entre dois vetores das classes 1.33 e 1.335.....	101
Figura 60 - interpolação linear entre dois vetores das classes 1.335 e 1.34.....	101
Figura 61 - Trajetórias de interpolação linear entre as classes 1.33 - 1.335 e 1.335 - 1.34, visualizadas via t-SNE.....	102
Figura 62 - interpolação esférica entre dois vetores das classes 1.33 e 1.335.....	103
Figura 63 - interpolação esférica entre dois vetores das classes 1.335 e 1.34.....	103
Figura 64 - Trajetórias de interpolação esférica entre Classes 1.33, 1.335 e 1.34 visualizadas via t-SNE.....	105
Figura 65 - Trajetória bidirecional no espaço latente (1.330 a 1.335, 1.335 a 1.330).....	107
Figura 66 - Imagens geradas da classe 1.330.....	107
Figura 67 - Imagens geradas da classe 1.335.....	108
Figura 68 - Trajetória das imagens transicional entre as classe (1.330 a 1.335).....	108
Figura 69 - Trajetória das imagens transicional entre as classe (1.335 a 1.330).....	108
Figura 70 - Trajetória bidirecional no espaço latente (1.335 a 1.34, 1.34 a 1.335).....	109
Figura 71 - Imagens geradas da classe 1.335.....	109
Figura 72 - imagens geradas da classe 1.34.....	110

Figura 73 - Trajetória das imagens transicional entre as classe (1.335 a 1.34).....	110
Figura 74 - Trajetória das imagens transicional entre as classe (1.34 a 1.335).....	110
Figura 75 - Trajetória bidirecional no espaço latente (1.330 a 1.335, 1.335 a 1.330).....	111
Figura 76 - Ilustra a transição entre classes.....	113
Figura 77 - Ilustra a plotagem de todas as classe sintética e a transição.....	114
Figura 78 - Espaço latente das imagens.....	115
Figura 79 – Imagens de transição entre as classes 1.330 e 1.335.....	116
Figura 80 - Classe estima no espaço latente (projeção t-SNE).....	117
Figura 81 - Curvas de intensidade da luz refletida nas imagens de transição (1.330 a 1.335)..	118

LISTA DE TABELAS

Tabela 1 - Hiperparâmetros usados na DCGAN.....	68
---	----

SUMÁRIO

1. INTRODUÇÃO.....	12
1.1 Motivação.....	15
1.2 Objetivo Geral.....	16
1.3 Objetivos Específicos.....	16
2. REFERENCIAL TEÓRICO.....	17
2.1 Inteligência Artificial.....	17
2.1.1 Aprendizado de Máquina.....	18
2.1.2 Redes Neurais Artificiais.....	19
2.2 Modelos Generativos de Imagens.....	21
2.2.1 Autoencoders Variacionais.....	22
2.2.2 Modelos de Difusão.....	22
2.2.3 Redes Generativas Adversariais.....	22
2.3 Variantes de GANs.....	26
2.3.1 Conditional GAN.....	27
2.3.2 Wasserstein GAN.....	29
2.3.3 Deep Convolutional GAN.....	32
2.4 Sensor SPR PPBIO.....	35
2.5 Dados Sintéticos.....	40
2.6 Técnicas de Geração de Novas Imagens.....	41
2.6.1 Operações com Vetores Latentes.....	41
2.6.2 Interpolação entre Vetores Latentes.....	43
2.6.3 Transição Entre Vetores Latente.....	46
2.7 Técnicas de Avaliação dos Resultados.....	47
3. METODOLOGIA.....	49
3.1 Ambiente de Desenvolvimento.....	49
3.2 Análise da Base de Dados.....	50
3.3 Exploração de Modelos Generativos de Imagens.....	54
3.4 Implementação da Arquitetura Escolhida.....	55
3.4.1 Preparação dos Dados.....	55
3.4.2 Arquitetura da DCGAN.....	56
3.4.3 Funções Auxiliares.....	59
3.4.4 Funções de Geração de Novas Imagens.....	63
3.4.5 Treinamento da DCGAN.....	67
3.4.6 Geração Direcionada de Imagem.....	69
3.5 Avaliação do Modelo Escolhido.....	71
3.5.1 Avaliação Quantitativa.....	71
3.5.2 Avaliação do Treinamento.....	74
3.5.3 Avaliação Física.....	75
3.5.4 Avaliação Visual.....	76
4. RESULTADOS.....	77

4.1 Avaliação do Modelo.....	77
4.1.1 Avaliação do Treinamento.....	77
4.1.2 Avaliação Visual das Imagens.....	79
4.1.3 Avaliação Geral dos Dados com MMD, Histogramas e o t-SNE.....	81
4.1.4 Avaliação por Classe.....	85
4.1.5 Avaliação Individual.....	89
4.2 Geração de Novas Classes.....	94
4.2.1 Operações com Vetores latente.....	94
4.2.2 Interpolações Entre Vetores Latentes.....	100
4.2.3 Transição Entre Vetores Latente.....	105
4.2.3.1 Casos com Parada Antecipada.....	106
4.2.3.2 Casos que Exigiram Todas as Épocas.....	111
4.2.4 Classes Estimadas.....	114
5. CONCLUSÃO.....	118
5.1 Trabalhos Futuros.....	120
REFERÊNCIA.....	121

1. INTRODUÇÃO

Nos últimos anos, a inteligência artificial, por meio de modelos generativos, ganhou destaque na geração de imagens sintéticas, que são imagens geradas artificialmente por algoritmos computacionais, dispensando o uso de sensores físicos, como câmeras ou sensores ópticos. Dentre os modelos generativos pesquisados, três abordagens se destacam: autoencoders variacionais (em inglês, *variational autoencoders* - VAEs), os modelos de difusão e as redes adversárias generativas (em inglês, *generative adversarial networks* - GANs). Cada modelo tem vantagens e limitações: os de difusão geram imagens de alta qualidade, mas exigem alto custo computacional (Ho, Jain e Abbeel, 2020); os VAEs tendem a produzir imagens borradas devido à sua natureza probabilística (Zhao, Song e Ermon, 2017). Já as GANs se destacam pela geração de imagens nítidas e realistas, baixo custo computacional (Goodfellow et al., 2014), embora alguns modelos sofram com instabilidade no treinamento (Arjovsky, Chintala e Bottou, 2017).

O marco inicial das GANs foi estabelecido por (Goodfellow et al., 2014), que propuseram uma arquitetura inovadora baseada na competição entre duas redes neurais: A primeira rede, chamada de geradora, tem como objetivo criar dados falsos (como imagens) a partir de vetores latentes. A segunda, é chamada de discriminadora, ela tenta distinguir se uma amostra recebida é real (proveniente dos dados de treinamento) ou falsa (gerada pela geradora). Durante o treinamento, a geradora melhora continuamente para enganar a discriminadora, enquanto o discriminador esta se torna mais eficiente em identificar dados falsos. Desde então, essa estrutura básica tem sido amplamente adaptada e expandida, dando origem a muitas variações de GANs.

Entre as GANs investigadas neste trabalho, as mais relevantes para processamento de imagens são a rede adversarial generativa condicional (em inglês, *conditional generative adversarial networks* - CGAN), que permite geração controlada por informações (Pingi, Nayak e Bashar, 2024); a rede adversarial generativa wasserstein (em inglês, *Wasserstein generative adversarial networks* - WGAN), que utiliza a distância de *wasserstein* para maior estabilidade no treinamento (Arjovsky, Chintala e Bottou, 2017) e a rede adversarial generativa convolucional profunda (em inglês, *deep convolutional generative adversarial*

Network - DCGAN), que emprega redes convolucionais para capturar características visuais hierárquicas (Radford, Metz e Chintala, 2016).

Paralelamente aos avanços em geração de imagens sintéticas, os sensores ópticos baseados na ressonância de plásmons de superfície (em inglês, *surface plasmon resonance* – SPR) têm se consolidado como ferramentas essenciais para análise molecular em tempo real, devido à sua alta sensibilidade, capacidade de detecção sem necessidade de marcadores e aplicação em diversas áreas, como biotecnologia, química e diagnóstico biomédico. Dentre as diferentes configurações, destaca-se a proposta por (Kretschmann e Raether, 1968).

Kretschmann, especialmente na variação que utiliza prismas polimérico para aplicações biológicas (PPBIO), como descrita por (MOREIRA, 2010). Essa estrutura, denominada sensor SPR PPBIO, se caracteriza por sua geometria simples, alta sensibilidade e viabilidade de produção nacional, além de dispensar elementos ópticos complexos, representando uma solução eficiente e de baixo custo para sensoriamento de substâncias aquosas.

Apesar do avanço dos sensores SPR e de seu potencial para aplicações científicas, a literatura aponta uma escassez de bases de dados públicas ou acessíveis que contenham imagens de sensores SPR obtidas em condições controladas (Tiwari et al., 2021). Essas restrições representam um gargalo crítico para o desenvolvimento de sistemas de análise baseados em aprendizado profundo, os quais exigem grandes volumes de dados para treinamento eficiente (LeCun, Bengio e Hinton, 2015).

Um dos esforços para mitigar essa limitação é o trabalho desenvolvido por (Sales, 2024), que propôs um processo estruturado para a geração de imagens sintéticas do sensor ópticos na configuração PPBIO. Embora o método tenha obtido êxito na criação de um conjunto de imagens, a quantidade gerada ainda é limitada, o que restringe sua aplicabilidade em cenários que demandam grandes volumes de dados, uma vez que, neste caso, as imagens são geradas a partir de cálculos com a estrutura de análise de Fresnel. Além disso, o processo de geração dessas imagens é demorado e suscetível a erros, uma vez que envolve diversas etapas manuais.

Esta pesquisa propõe uma abordagem inovadora para superar essas limitações por meio da geração de imagens sintéticas do sensor SPR, utilizando redes adversárias generativas. O método desenvolvido combina quatro elementos principais: uma arquitetura DCGAN adaptada para capturar as características do sensor SPR PPBIO; técnicas avançadas de exploração do espaço latente; algoritmos para transições suaves entre vetores latentes; e uma estratégia abrangente de avaliação.

A validação do modelo envolveu métricas quantitativas e qualitativas com o objetivo de verificar a qualidade e a fidelidade das imagens sintéticas geradas. No aspecto quantitativo, foram utilizadas as seguintes métricas: largura à meia altura (em inglês, *full width at half maximum* - FWHM), largura total, energia normalizada, funções de perda do gerador e do discriminador, discrepância máxima de médias (em inglês, *maximum mean discrepancy* - MMD), erro quadrático médio (em inglês, *mean squared error* - MSE) e distância euclidiana. Para a avaliação qualitativa, foram empregadas visualizações comparativas, projeções em espaço reduzido por meio da incorporação estocástica de vizinhança com distribuição t (em inglês, *t-distributed stochastic neighbor embedding* - t-SNE) e análise de histogramas.

Para demonstrar a eficácia deste estudo de geração de imagens sintéticas dos sensores SPR, foi utilizada uma DCGAN baseada na proposta original de (Radford, Metz e Chintala, 2016), que demonstrou seu potencial em tarefas de geração de imagens por meio de aprendizado não supervisionado. Como ponto de partida para a implementação, utilizou-se a base de código disponibilizada por (Barretto e Attux, 2019), posteriormente adaptada para atender às particularidades do conjunto de dados empregado neste trabalho. O modelo foi treinado com um conjunto de dados que está presente no trabalho de (Oliveira et al., 2025).

Esse dataset é composto por 600 imagens em tons de cinza, com resolução de 1280×1024 pixels, distribuídas igualmente entre cinco classes, em que cada classe representa um índice de refração distinto: 1,33; 1,335; 1,34; 1,345 e 1,35. A construção desse conjunto de dados seguiu uma metodologia estruturada, com o objetivo de representar fielmente diferentes condições de análise óptica, considerando os parâmetros experimentais do sensor SPR.

A contribuição central deste trabalho envolve dois aspectos: a demonstração da viabilidade do uso de GANs para gerar imagens realistas do sensor SPR na configuração

PPBIO, com alta diversidade para as classes presentes no conjunto de treinamento; e a geração de imagens correspondentes a outros índices de refração SPR, ou seja, de classes inexistentes no conjunto de treinamento. A integração desses resultados representa uma inovação significativa para aplicações científicas dependentes de sistemas SPR, oferecendo uma solução que alia precisão técnica à viabilidade econômica.

1.1 Motivação

A obtenção de imagens experimentais por meio de sensores de ressonância de plásmons de superfície, apresenta alto custo, exige infraestrutura laboratorial complexa e demanda elevado tempo experimental, o que limita a criação de bases de dados robustas para aplicações científicas em especial, no treinamento de redes neurais convolucionais para detecção de substâncias em tempo real.

Nesse cenário, a geração de imagens sintéticas surge como alternativa promissora para contornar tais restrições. Arquiteturas generativas, como VAEs, redes de mistura de densidade (em inglês, *mixture density networks* – MDN) e, em especial, GANs e suas variantes, têm se mostrado capazes de criar dados artificiais preservando características relevantes do domínio original. Na literatura, redes VAEs e MDN já foram utilizadas para simular estruturas ressonantes em sensores plasmônicos (Yaman et al., 2022; Batista, 2020). Modelos do tipo *Variational Autoencoder* também foram empregados para prever propriedades geométricas e ópticas de nanopartículas de ouro em sensores SPR (Yaman et al., 2022) e para o *design* de *metasurfaces* (Shi et al., 2020).

Zelaci et al. (2021) destacam-se por demonstrar como as GANs podem desempenhar um papel essencial no desenvolvimento de sensores de ressonância de plásmons de superfície baseados em fibras de cristal fotônico. A principal contribuição está no uso das GANs para gerar dados sintéticos realistas, ampliando significativamente o conjunto de dados disponível para treinar modelos de redes neurais artificiais (Zelaci et al., 2021).

Assim, a aplicação de GANs, em especial da arquitetura *deep convolutional* GAN, apresenta-se como estratégia viável para gerar imagens sintéticas de sensores SPR, viabilizando bases de dados mais extensas, diversificadas e controladas. A manipulação do espaço latente, que codifica as propriedades internas aprendidas pelo modelo, permite não

apenas ampliar conjuntos de treinamento, mas também criar variações controladas das imagens originais.

1.2 Objetivo Geral

Desenvolver modelos baseados na arquitetura de GANs para a geração de imagens sintéticas de SPR, com o intuito de construir uma base de dados artificial que viabilize o desenvolvimento de modelos inteligentes para sensores SPR.

1.3 Objetivos Específicos

i) Investigar, por meio da literatura especializada, diferentes modelos generativos como VAEs, modelos de difusão e GANs, identificando suas potencialidades e limitações para aplicação na geração de imagens do sensor SPR.

ii) Selecionar o modelo generativo mais adequado com base em critérios definidos, como fidelidade visual das imagens, estabilidade de treinamento, custo computacional, capacidade de manipulação no espaço latente, preservação de parâmetros físicos do sensor SPR e potencial de generalização para classes não presentes nos dados originais.

iii) Analisar a estrutura e a organização do espaço latente da rede geradora, além de medidas quantitativas e métricas morfométricas.

iv) Realizar experimentos de operações entre vetores, com o objetivo de obter variações controladas das imagens sintéticas do sensor SPR PPBIO.

v) Avaliar a relação entre a qualidade das imagens geradas, a complexidade do modelo e os custos computacionais envolvidos.

vi) Aplicar o modelo desenvolvido em um cenário prático colaborativo, onde a geração de imagens sintéticas complementa e amplia os dados obtidos por sensores físicos SPR PPBIO.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta os fundamentos teóricos essenciais para o desenvolvimento desta pesquisa, abordando conceitos de inteligência artificial e aprendizado de máquina, com ênfase em redes neurais artificiais e modelos generativos de imagens. Também são apresentados o sensor SPR PPBIO, o uso de dados sintéticos, as técnicas de geração de novas imagens e os métodos de avaliação dos resultados.

2.1 Inteligência Artificial

A inteligência artificial (IA) (em inglês, *artificial intelligence* - AI) busca simular comportamentos humanos na tomada de decisões e na execução de tarefas, desde as mais simples até as mais complexas (Monard; Baranauskas, 2000). A inteligência artificial está presente em diversas tecnologias do dia a dia, como assistentes virtuais (*Siri*, *Alexa*), recomendações de filmes e músicas (*Netflix*, *Spotify*), carros autônomos e até em diagnósticos médicos. Com o tempo, a IA consolidou-se como uma área central da ciência da computação, aplicável em diversos contextos, como geração de textos, áudios, vídeos e, especialmente, imagens.

A inteligência artificial é uma das inovações mais significativas dos últimos tempos, com suas origens na década de 1950, quando Alan Turing propôs o Teste de *Turing*, que tem como objetivo avaliar se uma máquina pode exibir um comportamento indistinguível do comportamento humano, de modo que um avaliador humano não consiga diferenciar se está interagindo com um humano ou com uma máquina (Turing, 1950). Essa proposta ganhou maior definição na *Dartmouth Conference*, realizada em Hanover, nos Estados Unidos (McCarthy *et al.*, 1955).

A Figura 1 ilustra a relação hierárquica entre a inteligência artificial e suas subáreas, como o aprendizado de máquina (em inglês, *machine learning* – ML) e as redes neurais artificiais (em inglês, *artificial neural networks* – ANNs), evidenciando como essas áreas se inter-relacionam. Em seguida, são descritos os conceitos centrais de cada uma dessas áreas e sua relevância no contexto das redes geradoras adversariais.

Figura 1 – Visão geral da Inteligência Artificial



No presente trabalho, a inteligência artificial foi explorada por meio do uso de redes neurais adversariais generativas, uma variação de GANs, mais especificamente em uma arquitetura conhecida como *deep convolutional* GAN para geração de imagens sintéticas semelhantes às imagens reais do sensor SPR.

2.1.1 Aprendizado de Máquina

O aprendizado de máquina é uma subárea da inteligência artificial, formalizada por Tom Michael Mitchell em seu livro *machine learning* (Tom M. Mitchell, 1997), no qual o autor apresenta uma das definições mais importantes da área. Nele, Mitchell explica que o ML foca no desenvolvimento de algoritmos capazes de aprender padrões diretamente dos dados, sem a necessidade de regras programadas manualmente.

Segundo Mitchell (1997), o aprendizado de máquina pode ser classificado em três categorias principais, sendo a primeira o aprendizado supervisionado, no qual os algoritmos são treinados com dados rotulados, permitindo a associação entre entradas e saídas. Um exemplo clássico é o sistema de filtragem de e-mails, capaz de detectar spam com base em exemplos previamente classificados, movendo-os automaticamente para pastas específicas.

O segundo tipo é o aprendizado não supervisionado, no qual os modelos são treinados com dados não rotulados e precisam identificar padrões e estruturas por conta própria (Bishop, 2006). Nesse contexto, os algoritmos de agrupamento (*clustering*) são amplamente utilizados para detectar similaridades entre dados. Por exemplo, é possível agrupar diferentes

tipos de cervejas com base em características como teor alcoólico, nível de amargor e cor, sem a necessidade de classificações prévias (Bishop, 2006).

O terceiro tipo é o aprendizado por reforço, no qual os modelos aprendem através de interações com um ambiente, recebendo recompensas (valores positivos ou negativos) com base em suas ações (Sutton e Barto, 2018). Um exemplo clássico é um agente treinado para jogar xadrez por meio de aprendizado por reforço e auto-jogo, sem receber conhecimento prévio. O agente experimenta movimentos e recebe recompensas conforme suas ações, ajustando sua estratégia ao longo do tempo para maximizar vitórias. Exemplos notáveis incluem o *Giraffe*, treinado por Lai (2015), e o *AlphaZero*, desenvolvido por Silver *et al.* (2017), que atingiram nível competitivo ao aprender somente por repetição e *feedback* positivo/negativo.

O Aprendizado de Máquina foi essencial nesta pesquisa para explorar a construção e o treinamento da DCGAN. Durante o processo, utilizou-se uma abordagem adversarial composta por dois modelos: um gerador treinado de forma não supervisionada e um discriminador treinado de forma supervisionada, para distinguir entre imagens reais e imagens sintéticas. Essa interação possibilitou que o modelo aprendesse padrões visuais a partir das imagens reais, gerando representações sintéticas de forma satisfatória de imagens sintéticas do sensor SPR.

2.1.2 Redes Neurais Artificiais

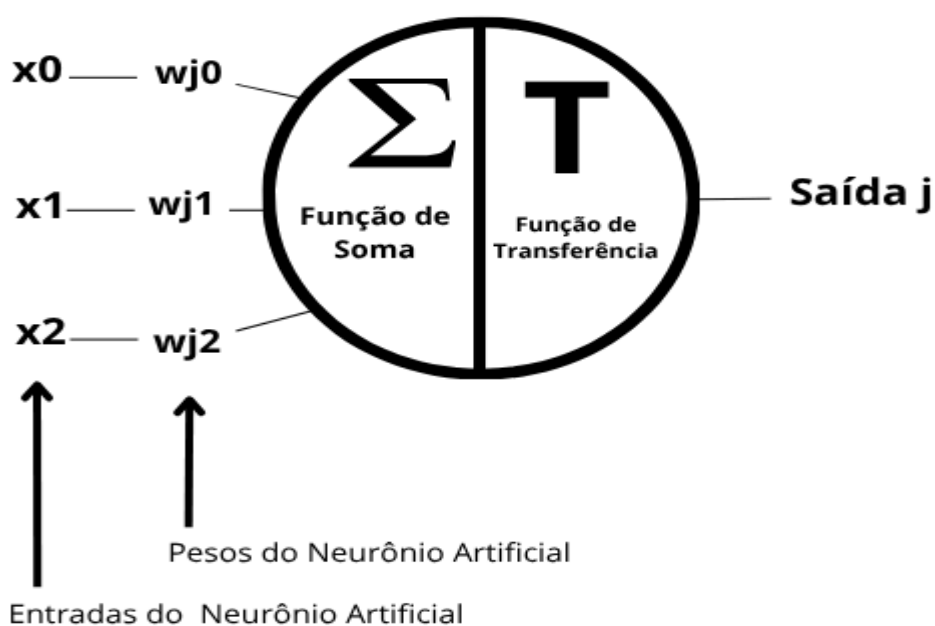
As ANNs são modelos computacionais inspirados no funcionamento do cérebro humano, capazes de aprender padrões complexos por meio de aprendizado supervisionado (Goodfellow et al., 2018). Sua arquitetura é construída em torno de neurônios artificiais (ou perceptrons), que simulam de forma simplificada os neurônios biológicos. Cada neurônio recebe múltiplas entradas, cada uma associada a um peso que determina sua importância relativa no processamento. Esses pesos são ajustados durante o treinamento, para minimizar erros nas predições (Rauber, 2005).

A história das redes neurais artificiais remonta aos anos 1940, com os trabalhos iniciais de McCulloch e Pitts em 1943, que propuseram um modelo matemático simplificado de um neurônio. No entanto, foi nas décadas seguintes que ocorreram avanços significativos,

com o desenvolvimento de algoritmos de treinamento mais eficientes e o aumento da capacidade computacional (Haykin, 1996).

A Figura 2 apresenta os principais componentes de um neurônio artificial, incluindo a entrada dos dados representada por (x), os pesos por (w_{j0}, w_{j1}, w_{j2}), a função de soma por (Σ), a função de transferência por (T) e a saída de informação por (j).

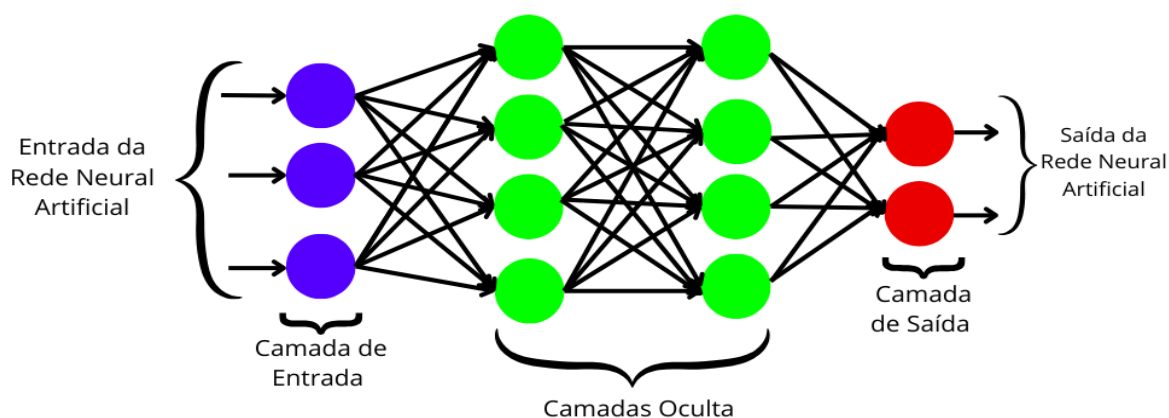
Figura 2 – Os principais componentes de um neurônio artificial



Fonte: Adaptada de (Tafner, 1998)

Na Figura 3 observam-se os componentes essenciais de uma rede neural simples, que incluem a camada de entrada, as camadas ocultas e a camada de saída. Esses elementos compõem a estrutura básica de uma rede neural. As camadas estão interconectadas por meio de nós ou neurônios. Cada camada recebe a saída da camada anterior como sua entrada, criando um fluxo sequencial de dados. As informações são processadas de maneira progressiva, desde a camada de entrada, passando pela camada oculta, até alcançar a camada de saída.

Figura 3 – Visão geral de uma rede neural artificial



Fonte: Adaptada de (Ferreira, 2020)

Com base nessa estrutura, compreende-se que as redes neurais processam dados de forma hierárquica, o que permite a extração progressiva de características em diferentes níveis de abstração.

No contexto desta dissertação, foram utilizadas duas redes neurais artificiais convolucionais profundas: o gerador e o discriminador. Essa estrutura permitiu a aprendizagem de representações complexas e a generalização a partir dos exemplos observados, tornando essas redes uma ferramenta fundamental no desenvolvimento de soluções baseadas em aprendizado de máquina e inteligência artificial (Goodfellow et al., 2014).

2.2 Modelos Generativos de Imagens

Os modelos generativos de imagens constituem uma classe de algoritmos de aprendizado de máquina capazes de aprender a distribuição de probabilidade subjacente a um conjunto de dados visuais, permitindo a geração de novas amostras realistas (Goodfellow et al., 2014). Esses modelos evoluíram ao longo das últimas décadas, passando de abordagens

probabilísticas clássicas para arquiteturas profundas modernas, com impacto significativo em áreas como síntese de imagens, aumento de dados e arte digital.

Neste trabalho, foram investigados três paradigmas distintos de modelos generativos: os *autoencoders variacionais* (VAEs), os modelos de difusão e as redes gerativas adversariais (GANs).

2.2.1 Autoencoders Variacionais

Propostos por Kingma e Welling, os Autoencoders Variacionais são modelos generativos baseados em inferência variacional, que aprendem uma representação latente compacta dos dados de entrada. Essa representação é então utilizada para gerar novas amostras por meio da decodificação de pontos amostrados do espaço latente, resultando em imagens semelhantes às originais (Kingma e Welling, 2022).

Embora sejam eficientes na modelagem de distribuições de probabilidade e na estruturação de espaços latentes contínuos, os VAEs enfrentam limitações. O processo de otimização do *Evidence Lower Bound* (ELBO) tende a promover uma compressão excessiva das informações, o que leva, frequentemente, à geração de imagens com aparência borrada e baixa definição, comprometendo a qualidade visual das amostras sintéticas.

2.2.2 Modelos de Difusão

Os modelos de difusão, introduzidos por (Sohl-Dickstein *et al.*, 2015), baseiam-se em processos estocásticos inspirados na termodinâmica. O processo de geração ocorre em duas etapas: (i) processo direto, no qual ruído é gradualmente adicionado às imagens reais por meio de um processo de difusão; e (ii) processo reverso, no qual um modelo neural é treinado para remover esse ruído de forma progressiva, reconstruindo as imagens a partir de ruído puro.

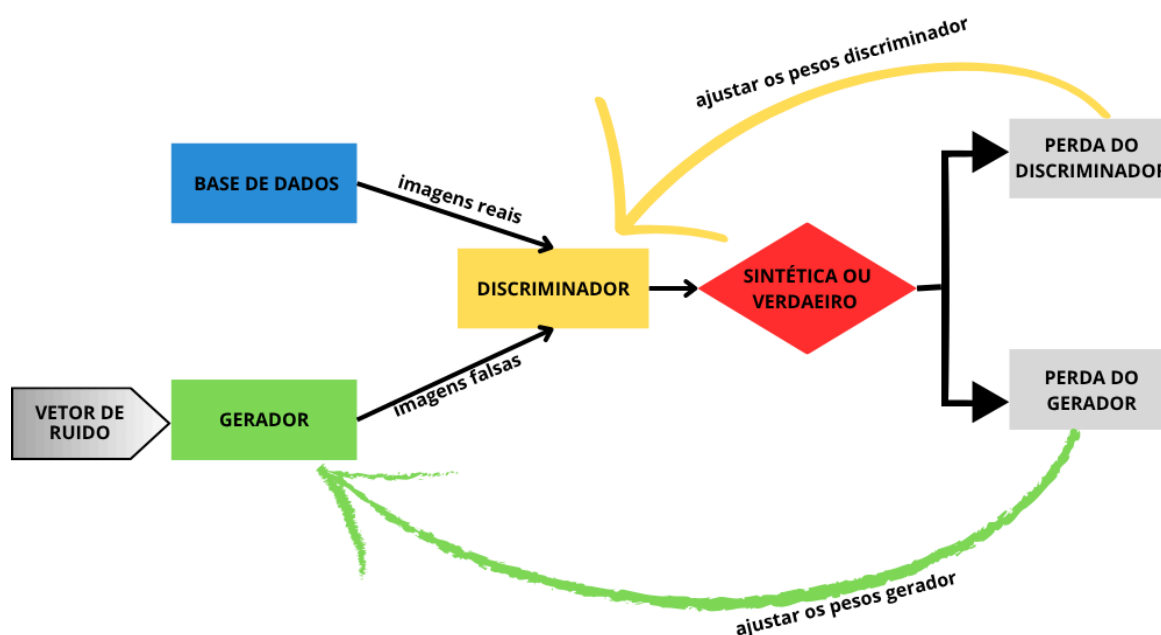
Esses modelos se destacam na literatura recente pela capacidade de gerar imagens de altíssima fidelidade, especialmente em tarefas condicionadas (Dhariwal e Nichol, 2021). No entanto, a natureza iterativa dos modelos de difusão impõe um custo computacional elevado, tanto no treinamento quanto na inferência, o que limita sua aplicabilidade em ambientes com restrição de recursos computacionais.

2.2.3 Redes Generativas Adversariais

A arquitetura *Generative Adversarial Network* foi inicialmente descrita no artigo *Generative Adversarial Networks*, propostas por Goodfellow, o qual revolucionou o campo da previsão ao utilizar o potencial das redes neurais geradora e discriminadora treinadas em conjunto (Goodfellow *et al.*, 2014). Essa abordagem inovadora é fundamentalmente composta por dois componentes principais: o gerador e o discriminador, que estabelecem uma dinâmica adversarial para produzir dados sintéticos que se assemelham aos dados reais.

A Figura 4 ilustra o treinamento da GAN: tudo começa com um vetor de ruído, que serve como entrada para o gerador, o componente responsável por criar imagens sintéticas. Em seguida, essas imagens geradas são enviadas ao discriminador junto com as imagens reais (*dataset*), que as avalia uma de cada vez e decide se são autênticas ou não. Esse processo se repete várias vezes, até que o gerador aprenda a criar imagens cada vez mais parecidas com as reais, enquanto o discriminador tenta se aprimorar em distinguir o que é real do que é falso.

Figura 4 – Ilustração do treinamento da GAN



A função objetiva da GAN é formulada como um problema de minimax, representado pela função (1):

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (1)$$

Neste contexto, temos dois modelos que competem entre si: o gerador (G) e o discriminador (D). O objetivo do gerador, representado por $\min_G$ da função (1), é minimizar sua função de perda para enganar o discriminador, ou seja, produzir amostras $G(z)$ que sejam tão realistas que D as classifique como verdadeiras. Por outro lado, o discriminador, representado por $\max_D$ da função (1), busca maximizar sua capacidade de distinguir corretamente os dados reais x das amostras falsas geradas pelo $G(z)$.

Aqui temos as funções de perda utilizada no gerador e no discriminador: O termo $\mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)]$ da função (1), representa o valor médio do logaritmo da probabilidade que o discriminador D atribui às amostras reais x . Esse valor varia de menos infinito ($-\infty$) até 0, e quanto mais próximo de 0, melhor o discriminador está reconhecendo essas amostras como reais, ou seja, $D(x)$ está próximo de 1. E o outro termo, $\mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$ da função (1), representa o valor médio do logaritmo da probabilidade que o discriminador D atribui às amostras geradas $G(z)$. Esse valor também varia de menos infinito ($-\infty$) até 0, e quanto mais negativo ele for, melhor o gerador G está atuando, pois está produzindo amostras falsas que se parecem com as reais, enganando o discriminador, fazendo $D(G(z))$ se aproximar de 1.

A função geradora, denotada por $G(z)$, onde z é uma amostra de ruído aleatório de uma distribuição pré-definida (como uma distribuição *gaussiana*), é responsável por mapear pontos do espaço latente para amostras de dados sintéticos. Seu objetivo é aprender a mapear eficazmente amostras de ruído para amostras que são indistinguíveis dos dados reais.

O espaço latente (EL) refere-se a um espaço multidimensional contínuo, geralmente de dimensionalidade reduzida, onde os dados são representados de forma compacta e estruturada. Essa representação latente captura características essenciais dos dados, permitindo operações como geração, interpolação e transformação.

A importância do espaço latente reside na sua capacidade de condensar as variabilidades dos dados reais em uma forma compacta, facilitando a síntese de novos exemplos que preservam propriedades semânticas e estruturais. Além disso, um espaço latente bem definido possibilita a manipulação controlada das características geradas, permitindo a

exploração de interpolação e transições suaves entre amostras e a modificação de atributos específicos sem necessidade de intervenção direta nos dados originais (Radford, Metz e Chintala, 2016).

Vetores latentes (VL) são representações de amostras individuais dentro do espaço latente, isto é, em espaços de baixa dimensionalidade que codificam características essenciais dos dados, como imagens (Bengio, Courville e Vincent, 2014). Cada dimensão do vetor contribui para características latentes, que podem corresponder a atributos semânticos ou padrões estruturais. Em GANs, os vetores latentes são mostrados a partir de uma distribuição simples, como a *gaussiana*, para servir como entrada ao gerador, que os transforma em dados sintéticos no espaço original.

As dimensões dos vetores latentes em GANs podem variar significativamente conforme a arquitetura e os objetivos do modelo. Por exemplo, a arquitetura DCGAN, proposta por (Radford, Metz e Chintala, 2016), utiliza vetores latentes de 100 dimensões, enquanto arquiteturas mais avançadas, como StyleGAN de (Karras, Laine e Aila, 2019), empregam vetores de 512 dimensões para capturar uma maior complexidade nos dados gerados.

Por outro lado, a função discriminadora, denotada por $D(x)$, onde x é uma amostra de dados, é responsável por avaliar a probabilidade da amostra ser real ou sintética. Normalmente, o valor fornecido pelo discriminador, representado por $D(x)$, é um número entre 0 e 1. Quando essa pontuação está próxima de 0, indica que a imagem é falsa, quando está próxima de 1, indica que é uma imagem verdadeira.

Conforme ilustrado na Figura 4, essas perdas são utilizadas para corrigir os respectivos pesos das redes, permitindo que ambas evoluam continuamente durante o treinamento. A função de perda consiste em duas partes principais: a perda do gerador e a perda do discriminador. A perda do gerador avalia sua capacidade de produzir amostras sintéticas convincentes, enquanto a perda do discriminador mede sua eficácia em distinguir corretamente entre amostras reais e geradas. O gerador busca reduzir sua própria perda ao criar amostras mais realistas, enquanto o discriminador também minimiza sua perda ao aprimorar sua capacidade de classificação. Essa competição adversarial resulta em um equilíbrio dinâmico entre as duas redes.

Apesar de terem sido utilizados amplamente em uma variedade de aplicações, as GANs enfrentam desafios significativos. De acordo com (Pradhyumna e Mohana, 2022), existem vários obstáculos significativos no treinamento de GAN, como:

i) Instabilidade: Refere-se a flutuações ou variações abruptas nos valores dos parâmetros da rede neural durante o treinamento (Arjovsky, Chintala e Bottou, 2017).

ii) Colapso de Modo (*Mode Collapse*): Ocorre quando o gerador de uma GAN produz amostras com baixa diversidade, limitando-se a poucas características do conjunto de dados (Radford, Metz e Chintala, 2016).

iii) Não convergência: Refere-se à incapacidade do modelo de estabilizar-se em uma solução satisfatória durante o treinamento (Arjovsky, Chintal e Bottou, 2017).

Embora *Autoencoders Variacionais* e *modelos de difusão* representem avanços significativos na geração de imagens, este trabalho optou por não utilizá-los devido a duas limitações, como a baixa qualidade visual das imagens geradas por VAEs e o elevado custo computacional inerente aos modelos de difusão.

Nesse contexto, as redes adversárias generativas emergiram como a escolha mais adequada, pois oferecem um equilíbrio eficaz entre qualidade na geração e eficiência computacional, especialmente quando se considera as características das imagens SPR, com pouca complexidade visual. No entanto, reconhece-se que o uso de GANs exige atenção a desafios específicos, como a instabilidade durante o treinamento e possíveis comprometimentos na qualidade das saídas. Além disso, a literatura já apresenta diversas soluções que podem ser aplicadas para a manipulação do espaço latente, possibilitando a síntese de imagens de classes não apresentadas ao gerador.

Assim, destaca-se a importância de investigar arquiteturas e técnicas que mitiguem esses problemas, garantindo tanto a estabilidade do processo de treinamento quanto a fidelidade das imagens geradas.

2.3 Variantes de GANs

Desde sua introdução por Goodfellow Et Al., (2014), GANs passaram por diversas modificações arquiteturas com o objetivo de superar limitações da versão original, como a

instabilidade no treinamento, o colapso de modo e a baixa qualidade de algumas amostras geradas. Essas modificações deram origem a um conjunto de variantes conhecidas como CGAN, WGAN e DCGAN, entre outras.

2.3.1 Conditional GAN

A CGAN é uma variação das GANs que permite a geração condicionada de amostras sintéticas. Isso significa que o modelo é capaz de gerar imagens associadas a uma determinada classe ou informação específica. Proposta por (Mirza e Osindero, 2014), essa abordagem tem sido amplamente adotada em diversas aplicações, especialmente na geração de imagens condicionadas.

A principal diferença entre a CGAN e a GAN convencional reside na incorporação de uma informação condicional, como rótulos de classe ou atributos específicos, tanto no gerador quanto no discriminador. Dessa forma, além do ruído aleatório, essas redes também utilizam a condição fornecida para realizar a tarefa proposta com maior controle sobre a saída gerada.

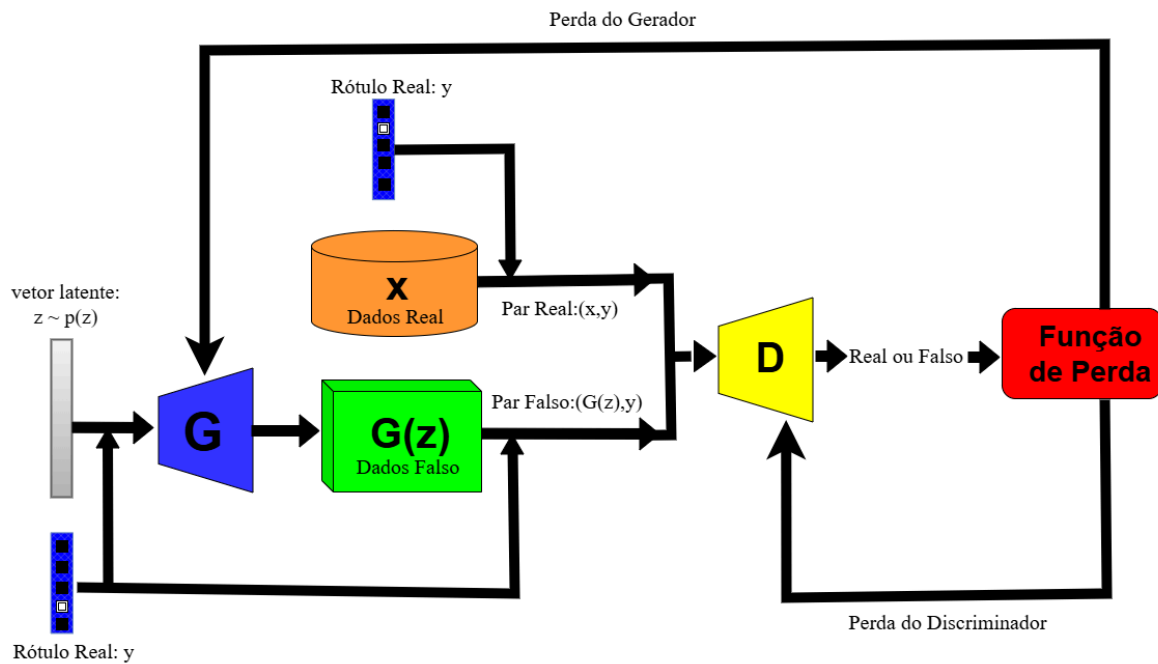
A Figura 5 demonstra o processo de treinamento de uma CGAN. Esse processo inicia-se com a amostragem de um vetor latente z , extraído de uma distribuição probabilística. Esse vetor é então combinado com um rótulo de classe real y , que representa a condição imposta ao modelo. O vetor latente condicionado é passado ao gerador G , que produz uma imagem sintética $G(z)$ que busca imitar os dados reais pertencentes à classe y .

Paralelamente, o sistema possui um conjunto de dados reais X , cada um associado ao seu respectivo rótulo y . Esses pares reais (x, y) e os pares gerados $(G(z), y)$ são enviados ao discriminador D , cuja função é distinguir entre os dados reais e os dados falsos. O discriminador recebe, não apenas a imagem (real ou gerada), mas também o rótulo correspondente, o que caracteriza a natureza condicional da arquitetura.

O discriminador então realiza sua tarefa de classificação, emitindo uma probabilidade de que o par recebido seja real. Com base nessa saída, a função de perda calcula dois tipos de erro: o erro do discriminador, que busca melhorar sua capacidade de distinguir entre dados reais e falsos, e o erro do gerador, que visa melhorar sua capacidade de enganar o discriminador, produzindo imagens cada vez mais realistas e coerentes com o rótulo y .

Essas perdas são então retropropagação da perda do discriminador atualizando seus próprios parâmetros, enquanto a perda do gerador é usada para atualizar os parâmetros de G , visando maximizar a chance de o discriminador classificar os dados gerados como reais. Esse ciclo se repete continuamente, com o gerador aprendendo a criar amostras mais convincentes e o discriminador se tornando mais exigente, até que um equilíbrio seja alcançado entre os dois.

Figura 5 – ilustra o treinamento da CGAN



Fonte: Adaptada de (Tafner, 1998)

A função Objetiva da CGAN é formulada como um problema de minimax, representado pela equação (2):

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z|y)))] \quad (2)$$

Considerando esse cenário, temos dois modelos que competem entre si: o gerador (G) e o discriminador (D). O objetivo do gerador, representado por $\min_G$ na equação (2), é minimizar sua função de perda para enganar o discriminador, ou seja, produzir amostras $G(z|y)$ que sejam tão realistas que D as classifique como verdadeiras, mesmo considerando a

condição y (como um rótulo ou informação adicional). Por outro lado, o discriminador, representado por $\max_D$ na equação (2), busca maximizar sua capacidade de distinguir corretamente os dados reais x , condicionados por y , das amostras falsas geradas pelo gerador $G(z|y)$, utilizando também o rótulo y como informação auxiliar no processo de decisão.

Aqui temos as funções de perda utilizadas pelo gerador e pelo discriminador. O termo $E_{x \sim p_{data}(x)}[\log D(x|y)]$, da equação (2), representa o valor médio do logaritmo da probabilidade que o discriminador D atribui às amostras reais x , dado o rótulo y . Esse valor varia de $-\infty$ até 0 e, quanto mais próximo de 0, melhor o discriminador está reconhecendo essas amostras como reais (ou seja, $D(x|y)$ está próximo de 1). O outro termo, $E_{z \sim p_z(z)}[\log(1 - D(G(z|y)))]$, também da equação (2), representa o valor médio do logaritmo da probabilidade que o discriminador atribui às amostras geradas $G(z|y)$. Esse valor também varia de $-\infty$ até 0 e, quanto mais negativo ele for, melhor o gerador G está atuando, pois está produzindo amostras sintéticas que se parecem com as reais, enganando o discriminador fazendo $D(G(z|y))$ se aproximar de 1.

2.3.2 Wasserstein GAN

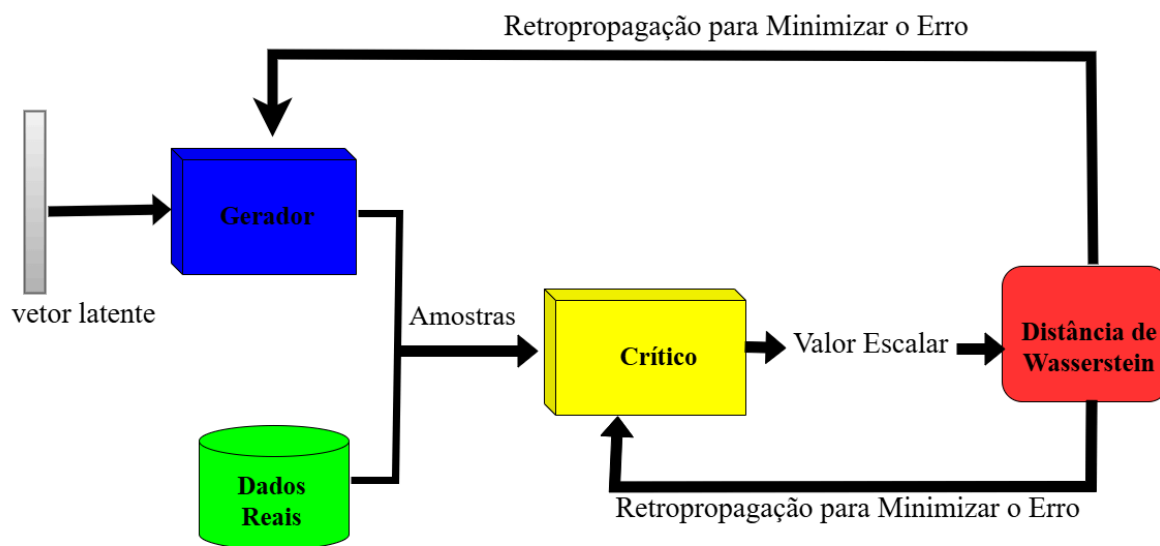
A WGAN é uma variante das GANs introduzida por (Arjovsky, Chintala e Bottou, 2017). Essa abordagem visa resolver os problemas de treinamento encontrados em GANs convencionais por meio da substituição da função de perda tradicional pela distância de Wasserstein, também conhecida como distância *Earth Mover* (Arjovsky, Chintala e Bottou, 2017).

As primeiras versões de GANs enfrentam desafios no treinamento devido à natureza não diferenciável de sua função de perda, o que pode resultar em problemas de convergência e dificuldades na atualização dos pesos do gerador e do discriminador. A WGAN aborda essas limitações ao substituir a função de perda original, baseada na entropia cruzada, pela distância de *Wasserstein*, que oferece uma medida mais contínua e sensível à diferença entre as distribuições de probabilidade das imagens reais e sintéticas.

A Figura 6 ilustra o treinamento da WGAN. O processo inicia-se com a entrada de um vetor latente aleatório no gerador, que transforma esse vetor em uma amostra sintética. Essa amostra é então enviada ao Crítico, juntamente com os dados reais. Ao contrário do

discriminador tradicional que classifica amostras como “reais” ou “falsas”, o crítico da WGAN atribui um valor escalar contínuo, refletindo a autenticidade das amostras. Com base nesse valor escalar, é calculada a distância de *Wasserstein*, que serve como função de perda para o treinamento. Essa métrica é então utilizada para realizar a *retropropagação para minimizar o erro*: tanto o gerador quanto o crítico ajustam seus parâmetros para minimizar essa distância. Geralmente, o crítico é atualizado múltiplas vezes para cada atualização do gerador, assegurando uma estimativa precisa da função de perda.

Figura 6 – ilustra o treinamento da WGAN



Fonte: Adaptada de (Radford; Metz; Chintala, 2016)

A função objetiva da WGAN, apresentada no trabalho de Arjovsky, Chintala e Bottou (2017), é formulada como um problema de minimax, representado pela equação (3).

$$W(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} (\mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_\theta}[f(x)]) \quad (3)$$

Dessa forma, temos dois modelos que atuam de forma adversária: o gerador (G), que tenta produzir amostras sintéticas realistas, e o crítico (f), que busca distinguir entre dados reais e gerados. Diferentemente das GANs tradicionais, a WGAN não utiliza um

discriminador com uma saída binária (real/falso), mas sim um crítico que atribui valores reais (*escores*), representando o quão “real” uma amostra parece (Gulrajani *et al.*, 2017).

A função objetivo da WGAN é expressa pela distância de *Wasserstein* entre duas distribuições de probabilidade: P_r , que representa os dados reais, e P_θ , que representa os dados gerados pelo modelo. Essa distância, denotada por $W(P_r; P_\theta)$, mede o custo mínimo para transformar a distribuição gerada na distribuição real, sendo uma métrica mais estável e contínua do que as usadas em GANs tradicionais.

Para calcular essa distância, considera-se o supremo (*sup*) da diferença entre duas expectativas, onde o supremo é tomado sobre todas as funções f que são *1-Lipschitz*, ou seja, funções cuja variação é limitada para evitar mudanças abruptas. Essa restrição é fundamental para garantir a validade e a estabilidade da métrica durante o treinamento.

A função f , chamada de crítico, atua como um avaliador que busca maximizar a diferença entre os escores médios atribuídos às amostras reais e às amostras geradas. A primeira expectativa, $\mathbb{E}_{x \sim P_r}[f(x)]$, representa a média dos valores dados pelo crítico às amostras reais; valores maiores indicam que o crítico reconhece essas amostras como autênticas.

Já a segunda expectativa, $\mathbb{E}_{x \sim P_\theta}[f(x)]$ calcula a média dos escores dados às amostras sintéticas produzidas pelo gerador. O objetivo do gerador é minimizar a diferença entre essas duas expectativas, produzindo amostras tão convincentes que o crítico lhes atribui escores próximos aos das amostras reais.

Assim, o treinamento envolve a minimização da distância de *Wasserstein* entre as distribuições de probabilidade das imagens reais e geradas, sendo realizado em duas etapas: atualizando os pesos do gerador para “enganar” o crítico e, geralmente, atualizando os pesos do crítico várias vezes (por exemplo, cinco) para cada atualização do gerador, a fim de manter uma estimativa confiável da métrica de *wasserstein* (Arjovsky, Chintala e Bottou, 2017).

Embora a WGAN apresente vantagens teóricas em termos de estabilidade de treinamento, sua implementação prática exige validação empírica cuidadosa, como testes com diferentes números de atualizações do crítico e ajustes sensíveis da penalização de gradiente. Esses procedimentos demandam tempo e esforço significativos para alcançar um equilíbrio

adequado entre as redes, o que tornaria o processo de desenvolvimento mais custoso e demorado. Considerando o objetivo deste trabalho e a simplicidade do conjunto de dados, optou-se por utilizar a DCGAN, cuja configuração mais direta permite alcançar resultados satisfatórios com menor esforço de ajuste e validação.

2.3.3 Deep Convolutional GAN

A DCGAN é uma variante das redes generativas adversariais, introduzida por Alec Radford, Luke Metz e Soumith Chintala em 2016. Essa abordagem combina a estrutura original das GANs, proposta por (Goodfellow Et Al., 2014), com redes neurais convolucionais profundas (Deep Convolutional Neural Networks – DCGAN), tornando-se especialmente adequada para tarefas de geração de imagens (Radford, Metz e Chintala, 2016).

Na DCGAN, o gerador é responsável por transformar um vetor latente de baixa dimensionalidade em uma imagem de maior resolução, utilizando camadas convolucionais transpostas. Por outro lado, o discriminador é composto por camadas convolucionais convencionais, encarregado de distinguir entre imagens reais e imagens geradas. A função objetiva da DCGAN é a mesma utilizada na GAN tradicional, baseada em um jogo do tipo minimax entre dois modelos: o gerador (G) e o discriminador (D). O objetivo do gerador é criar amostras que enganem o discriminador, enquanto este busca distinguir corretamente entre imagens reais e sintéticas. Para uma discussão mais aprofundada sobre a formulação matemática da função objetivo, consulte a Seção 2.2.3.

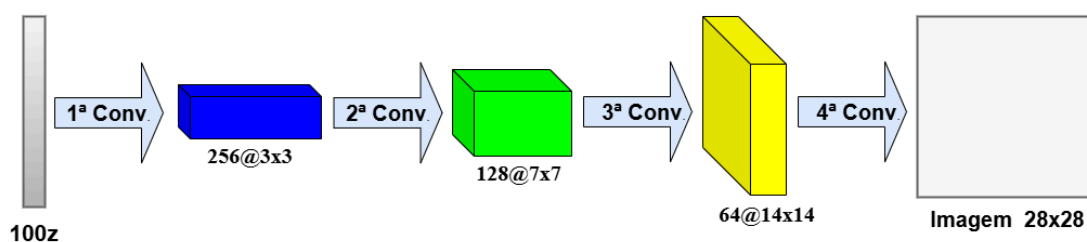
Para compreender o funcionamento da DCGAN, é necessário entender os fundamentos das redes neurais convolucionais profundas. Propostas por (LeCun, Bengio e Hinton, 2015) é amplamente adotadas após o sucesso da AlexNet (Krizhevsky, Sutskever e Hinton, 2012), essas redes são compostas por camadas convolucionais intercaladas com funções de ativação não lineares (como ReLU), camadas de normalização e, eventualmente, camadas de *pooling*. Essas estruturas permitem a extração hierárquica de características, o que é essencial para o processamento de imagens (Simonyan e Zisserman, 2015).

As camadas convolucionais aplicam filtros (*kernels*) sobre as entradas, gerando mapas de ativação que destacam padrões como bordas e texturas. Durante o treinamento, os pesos desses filtros são ajustados para capturar as características mais relevantes. As funções de

ativação não lineares, como ReLU ou *LeakyReLU*, introduzem complexidade no aprendizado, permitindo que a rede modele relações não lineares. Técnicas como camadas de normalização contribuem para a estabilidade do treinamento e aceleram a convergência. Em algumas arquiteturas, camadas de *pooling* são empregadas para reduzir a dimensionalidade, mas em redes geradoras, a manipulação do tamanho é geralmente feita pelas camadas convolucionais transpostas ou pelo uso de *stride*.

O gerador da DCGAN, ilustrado na Figura 7, transforma um vetor latente aleatório (z de 100 dimensões) em uma imagem sintética com características visuais realistas. No exemplo apresentado, a primeira camada de convolução transposta gera um tensor com a notação ' $256@3\times3$ '—ou seja, 256 mapas de características de dimensão 3×3 pixels (256 filtros, cada um com 3 pixels de altura e largura). Em seguida, uma segunda camada amplia essa representação para $128@7\times7$ (128 mapas de 7×7 pixels), e a terceira camada a converte em $64@14\times14$ (64 mapas de 14×14 pixels). Por fim, a última camada produz a imagem final em escala de cinza, com resolução de 28×28 pixels (Li, Song e Zhang, 2020).

Figura 7 – Camadas Convolucionais Transpostas da rede geradora da DCGAN

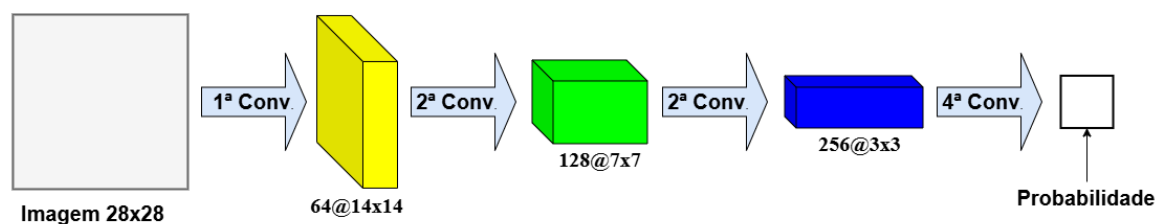


Fonte: Adaptado de (Li, Song e Zhang, 2020)

O discriminador, por sua vez, recebe uma imagem em tons de cinza de 28×28 pixels e a processa por meio de uma sequência de camadas convolucionais. A primeira camada gera uma saída de $64@14\times14$, ou seja, 64 mapas de características com dimensões 14×14 pixels, seguida por uma camada de $128@7\times7$ (128 mapas de 7×7 pixels) e outra de $256@3\times3$ (256 mapas de 3×3 pixels). Por fim, a última camada condensa as informações em um único valor

escalar, que indica a probabilidade de a imagem ser real. Essa estrutura progressiva permite ao discriminador refinar as características extraídas para uma classificação eficiente (Li, Song e Zhang, 2020).

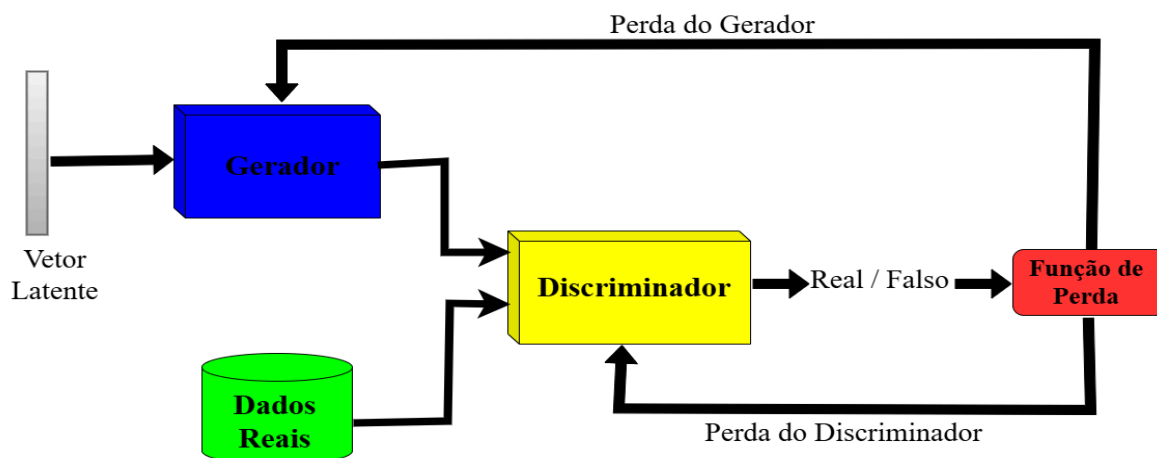
Figura 8 – Camadas Convolucionais da rede discriminadora da DCGAN



Fonte: Adaptado de (Li, Song e Zhang, 2020)

O treinamento da DCGAN é ilustrado na Figura 9, seu ciclo é baseado em um ciclo adversarial entre o gerador Figura 7 e o discriminador Figura 8. O processo se inicia com a entrada de um vetor latente no gerador, que produz uma imagem sintética. Essa imagem, juntamente com uma imagem real, é passada ao discriminador, que tenta distinguir entre ambas. Em seguida, calcula-se a função de perda, que ajusta os pesos de ambos os modelos. A perda do discriminador visa aprimorar sua capacidade de classificação, enquanto a perda do gerador busca maximizar sua habilidade de enganar o discriminador.

Figura 9 – Fluxograma do treinamento da DCGAN



Fonte: Adaptado de (Radford, Metz e Chintala, 2016)

Dentre as variantes de GANs analisadas, a CGAN, WGAN e DCGAN, este trabalho optou pela utilização exclusiva da DCGAN, por representar um equilíbrio adequado entre simplicidade arquitetural, custo computacional e qualidade das amostras geradas. A CGAN, embora útil para tarefas supervisionadas, depende de informações adicionais, como rótulos ou atributos, o que torna seu uso incompatível com o foco deste estudo, que é a geração não supervisionada de imagens do sensor SPR.

Já WGAN, por sua vez, oferece estabilidade teórica superior durante o treinamento, mas impõe um custo computacional elevado, exigindo múltiplas atualizações do crítico e o uso de penalização de gradiente. Tais exigências se mostram desproporcionais frente ao conjunto de dados utilizado, composto por imagens de baixa resolução e complexidade. Dessa forma, a DCGAN foi selecionada por representar um equilíbrio entre simplicidade, custo computacional e capacidade de gerar amostras visualmente satisfatórias. Sua natureza não supervisionada, aliada à eficiência observada na literatura para tarefas semelhantes, justificou sua adoção como a única abordagem implementada neste estudo.

2.4 Sensor SPR PPBIO

A ressonância de plásmons de superfície (em inglês, *surface plasmon resonance* – SPR) ocorre quando a luz, sob condições específicas de ângulo de incidência, polarização e comprimento de onda, incide sobre uma camada metálica, excitando elétrons na interface metal-dielétrica e possibilitando o acoplamento ressonante. Esse fenômeno foi descrito por Ritchie, sendo Otto foi um dos pioneiro a propor uma configuração experimental para excitar plásmons utilizando um prisma separado por uma fina camada de ar. Posteriormente, Kretschmann e Raether otimizaram esse arranjo ao acoplar diretamente o metal à superfície do prisma, estabelecendo a base da configuração mais utilizada atualmente em sensores SPR (RITCHIE, 1957; OTTO, 1968; KRETSCHMANN; RAETHER, 1968).

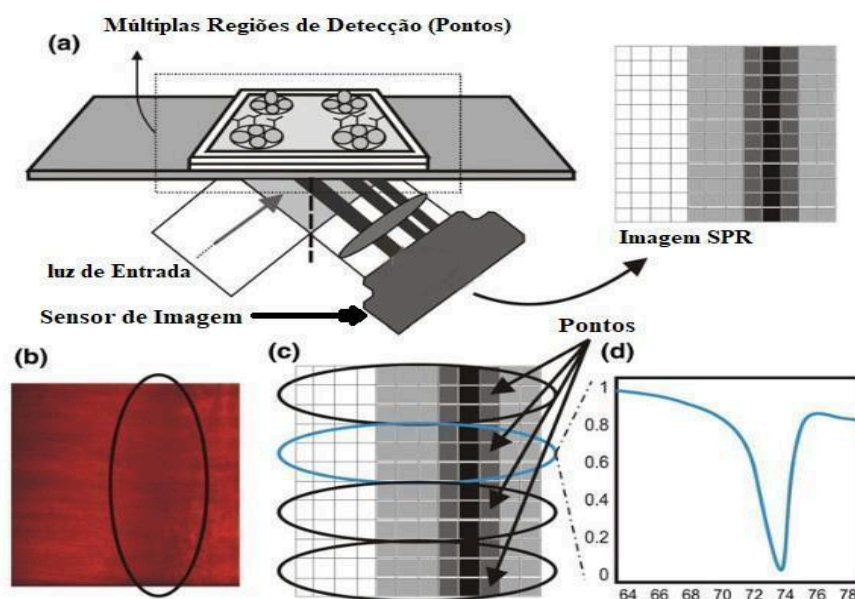
A Figura 10 ilustra de forma geral, o funcionamento de sensores baseados na técnica de ressonância de *plasmons* de superfície. Na Figura 10a), é apresentado um esquema conceitual de um sensor SPR, no qual um feixe de luz monocromática incide sobre um prisma óptico, que direciona a luz para uma estrutura sensora composta por múltiplas camadas: um substrato dielétrico, uma camada metálica (geralmente ouro ou prata) e o meio de análise. Essa configuração permite a excitação dos plasmons de superfície quando a luz atinge o ângulo de ressonância. A luz refletida é captada por um sensor de imagem, resultando em uma imagem SPR, que exibe padrões ópticos relacionados às propriedades do material analisado.

Na Figura 10b), é exibida uma imagem experimental colorida com uma faixa vertical mais escura, que indica a variação ao longo da amostra. Essa imagem pode ser processada e convertida em uma matriz de pixels, conforme ilustrado na Figura 10c), facilitando a análise de regiões específicas, conhecidas como pontos de detecção.

A partir desses pontos, é possível extrair curvas de refletividade, como a apresentada Figura 10d). Essa curva SPR exibe um vale de ressonância, que representa o ponto de mínima refletividade e está diretamente relacionado ao índice de refração da substância sob análise.

Essa abordagem permite monitorar com alta sensibilidade alterações nas propriedades ópticas da amostra, o que torna os sensores SPR uma ferramenta poderosa em diversas áreas, como diagnóstico biomédico, monitoramento ambiental e controle de qualidade.

Figura 10 - Funcionamento de um sensor SPR: esquema óptico, imagem experimental e extração de curva de ressonância



Fonte: Adaptada de (Oliveira *et al.*, 2019)

Um modelo de sensor SPR desenvolvido por (MOREIRA, 2010) baseado na configuração de Kretschmann, denominado sensor SPR PPBIO, utiliza um Prisma Polimérico para Aplicações Biológicas (PPBIO). Inicialmente projetado para aplicações biológicas, esse sensor emprega um prisma polimérico de geometria trapezoidal compacta, com altura inferior a 3 mm, que dispensa o uso de géis ou óleos de acoplamento óptico. Essa característica representa uma vantagem significativa em relação a configurações SPR convencionais, ao simplificar a montagem experimental.

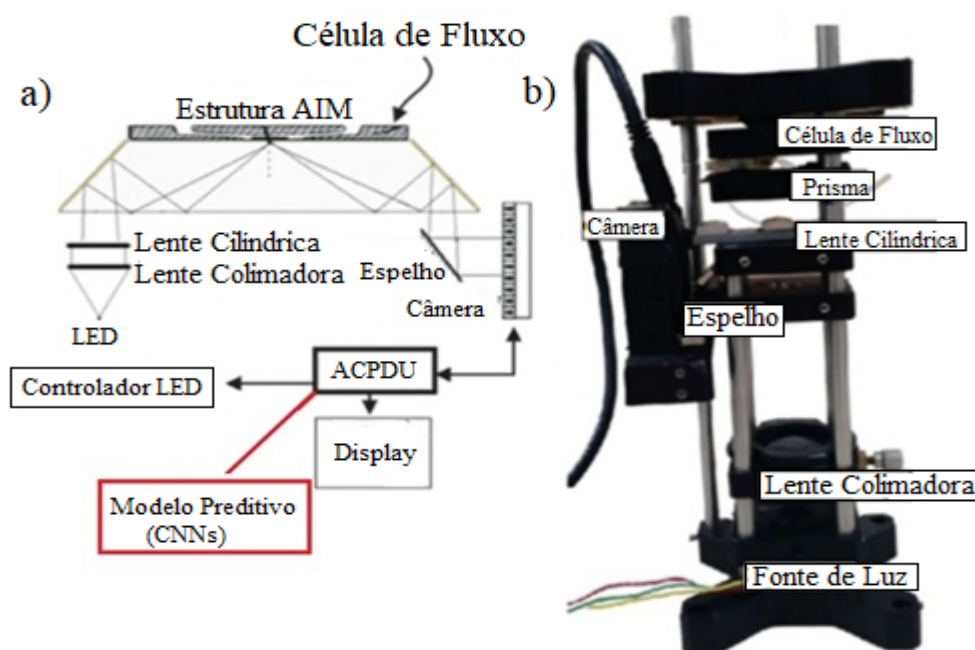
Posteriormente, o modelo foi caracterizado e aplicado experimentalmente por Oliveira em 2019, que detalhou sua implementação em um sistema óptico compacto e automatizado, operando no modo de interrogação angular (AIM).

A Figura 11 apresenta o sistema SPR PPBIO. Na parte da Figura 11a), observa-se o esquema óptico da estrutura AIM, no qual a luz emitida por uma fonte LED é colimada e focalizada por lentes, incide sobre a célula de fluxo e atravessa o prisma polimérico, onde ocorre a excitação dos plasmons de superfície. A luz refletida é direcionada por um espelho

até uma câmera CMOS, que realiza a aquisição da imagem SPR. O sistema é controlado eletronicamente por um conjunto que inclui um controlador de LED, uma unidade de processamento (ACPDPU) e um display.

A parte da Figura 11b), mostra a fotografia da montagem física do sistema, destacando os principais componentes: fonte de luz, lentes colimadoras e cilíndrica, espelho, prisma, célula de fluxo e câmera. A estrutura é sustentada verticalmente, garantindo o alinhamento preciso dos elementos ópticos para a obtenção das imagens SPR.

Figura 11 - Esquema óptico e montagem física do sistema do SPR PPBIO com interrogação angular (AIM)



Fonte: Adaptada de (OLIVEIRA et al., 2019)

O princípio de funcionamento baseia-se na incidência de um diodo laser de 670 nm sobre uma estrutura multicamada composta por um substrato de vidro BK7, uma fina camada de ouro (50 nm), e a amostra líquida em análise. A interação entre a luz incidente e os plasmons de superfície na interface metal-dielétrico resulta em uma atenuação no feixe refletido, capturada pela câmera. A imagem formada apresenta uma região escura associada à condição de ressonância, cuja posição espacial varia conforme o índice de refração da

substância analisada, possibilitando a detecção de alterações mínimas na ordem de 10^{-5} RIU (Refractive Index Unit). O RIU é a unidade utilizada para expressar pequenas variações no índice de refração, que por sua vez é definido como a razão entre a velocidade da luz no vácuo e no material analisado.

A faixa angular útil do sistema, entre 62° e 73° , foi otimizada para análise de substâncias aquosas com índices de refração próximos ao da água (1,33 a 1,385), tornando-o especialmente adequado para aplicações em monitoramento químico e biológico.

Trabalhos mais recentes têm explorado a integração deste sensor com técnicas de inteligência artificial, especialmente com redes neurais artificiais, para automatizar a classificação de substâncias diretamente a partir das imagens SPR, superando limitações de métodos convencionais baseados no ajuste manual de curvas (Gomes; Souza; Oliveira, 2021), (Malinick *et al.*, 2022).

Essa abordagem inovadora abre caminho para o desenvolvimento de sistemas de sensoriamento óptico inteligentes, com aplicações potenciais em diagnóstico médico, monitoramento ambiental e controle industrial, onde a detecção rápida e precisa de variações em interfaces líquidas é essencial. Ainda assim, a calibração precisa da relação pixel-ângulo e a otimização dos parâmetros de aquisição de imagem permanecem como desafios cruciais para maximizar o desempenho desse tipo de sensor em ambientes práticos.

A imagem obtida pelo sensor, portanto, não apenas revela visualmente a condição de ressonância, mas também permite uma análise quantitativa por meio da curva SPR. Essa curva representa a variação da refletividade da luz em função do ângulo de incidência, e serve como base para a obtenção de diversas métricas morfológicas. Tais métricas são essenciais para avaliar o desempenho do sensor, caracterizar as amostras e alimentar algoritmos de classificação automática. Entre elas, destaca-se a Largura Total na Metade da Altura (FWHM), cuja determinação permite uma análise mais detalhada do ponto de ressonância e da resposta óptica do sistema.

Dentre essas métricas, a FWHM se destaca por ser diretamente extraída da curva SPR. Essa curva permite identificar a posição de ressonância por meio do ponto de menor intensidade (valor mínimo), que pode ser determinado por diferentes métodos, como o cálculo

do centróide ou o ajuste de um polinômio aos dados experimentais para localizar o mínimo de forma precisa.

A partir desse ponto mínimo, calcula-se a FWHM como a distância entre dois pontos da curva localizados à metade da diferença entre os valores máximo e mínimo de refletividade. Conforme mostrado nas Funções 4 e 5, os pontos de referência utilizados neste cálculo são denominados CR (direita do mínimo) e CL (esquerda do mínimo), e também são empregados na estimativa da simetria (A) e da largura da curva (L_C). Essas medidas morfológicas são sensíveis a alterações no índice de refração do analito e na espessura da camada sensível, sendo, portanto, indicadores úteis para a análise comparativa entre diferentes amostras (Sales, 2024).

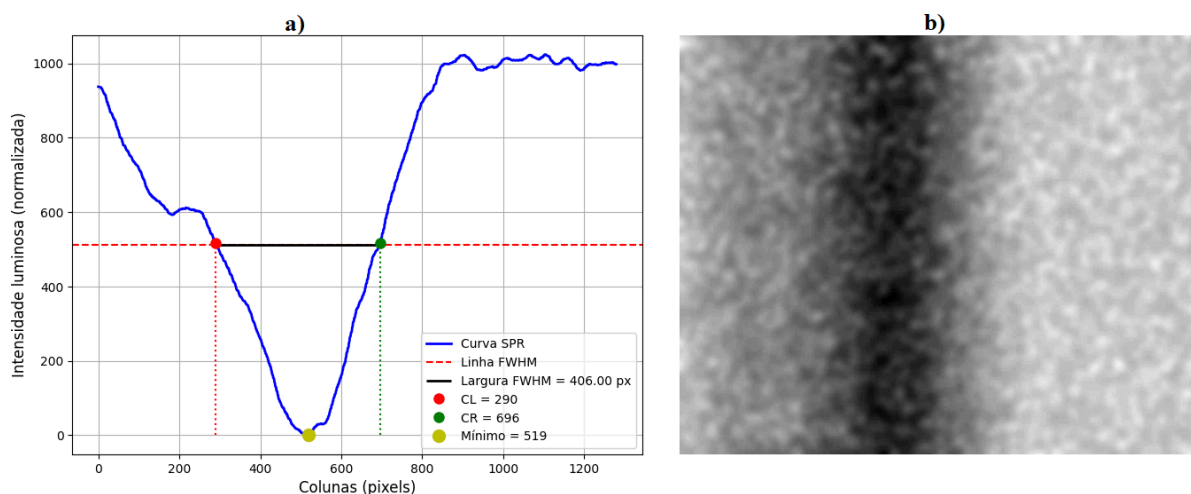
$$L_C = CR + CL \quad (4)$$

$$A = CR/CL \quad (5)$$

A Figura 12 apresenta um gráfico e uma imagem. Na Figura 12a), apresenta-se a curva SPR obtida a partir da intensidade luminosa normalizada (eixo y) em função da posição dos pixels na imagem (eixo x). A linha azul corresponde à curva experimental, onde se observa um vale acentuado característico da ressonância de plasmons de superfície. A linha tracejada vermelha indica o nível de meia altura utilizado para calcular a largura à meia altura (FWHM). Os pontos em vermelho (CL = 290 px) e verde (CR = 696 px) marcam, respectivamente, as posições laterais esquerda e direita na interseção com a linha FWHM, enquanto o ponto amarelo (Mínimo = 519 px) indica a posição do valor mínimo da curva. A largura FWHM calculada é de 406,00 pixels, representada pela linha preta horizontal.

Na Figura 12b), apresenta-se a imagem real correspondente ao gráfico, onde se identifica uma faixa central escura associada ao mínimo de intensidade da curva SPR, ladeada por regiões mais claras que indicam maior refletividade.

Figura 12 – Curva SPR com indicação de CR, CL, FWHM e ponto mínimo, acompanhada da imagem real.



2.5 Dados Sintéticos

Dados sintéticos são informações geradas artificialmente por meio de simulações computacionais ou modelos matemáticos, e não capturadas diretamente por sensores físicos. No contexto da aprendizagem profunda, esses dados são amplamente utilizados para treinar modelos em cenários com escassez de dados reais. Os dados sintéticos podem acelerar o desenvolvimento de sistemas inteligentes ao fornecerem conjuntos diversificados, balanceados e com anotações precisas.

Neste trabalho, foram empregados dados sintéticos gerados a partir das curvas de ressonância de plasmons de superfície, conforme o trabalho desenvolvido por Oliveira, (2025) banco de imagens simuladas com base em curvas SPR teóricas, utilizando as equações de Fresnel para representar diferentes índices de refração de substâncias aquosas, aliado a técnicas de processamento digital de imagem. Esse conjunto foi originalmente construído para o treinamento de redes neurais convolucionais (CNNs) e, aqui, é reaproveitado para o treinamento e validação de uma DCGAN, com o objetivo de gerar novas imagens realistas do domínio SPR.

2.6 Técnicas de Geração de Novas Imagens

Modelos generativos, como as GANs, operam em um espaço latente contínuo e estruturado, a partir do qual é possível gerar amostras realistas. Esse espaço latente, geralmente de alta dimensionalidade, permite manipulações vetoriais que resultam em alterações controladas nas imagens geradas. Entre as principais técnicas de geração de novas imagens, destacam-se as operações vetoriais (soma e subtração), interpolação entre vetores latentes e transição entre vetores latentes.

2.6.1 Operações com Vetores Latentes

O artigo (Karras, Laine e Aila, 2019) apresenta uma arquitetura inovadora, chamada *StyleGAN*, que utiliza operações aritméticas no espaço latente para melhorar a flexibilidade e qualidade das imagens geradas por GAN. Esta arquitetura propõe uma abordagem alternativa para a síntese de imagens, permitindo um controle intuitivo e específico em relação aos atributos de alto nível e à variação estocástica nas imagens geradas. Ela introduz uma separação automática e não supervisionada de atributos de alto nível e variação estocástica, bem como métodos automatizados para quantificar a qualidade de interpolação e a desentrelaçamento dos fatores latentes de variação.

Nessa arquitetura, os os vetores latentes (VL) são usados para controlar diversos aspectos da geração de imagens, como pose, expressão facial e estilo visual. As operações aritméticas com VL permitem criar características a partir de dois VL de classes diferentes, sendo crucial para a geração de imagens que transitam entre diferentes características. Na Figura 13, é ilustrado dois conjuntos de imagens que foram gerados a partir de seus respectivos espaços latentes das fontes A e B. Então esses estilos estão relacionados às características principais, como pose idade, sexo, formato do rosto, estilo geral do cabelo e óculos, que estão sendo transferidos das duas fontes para a nova imagem.

Figura 13 – Geração de imagens usando operações aritméticas em GAN



Fonte: Adaptada de (Karras, Laine e Aila, 2019)

Portanto, as imagens geradas a partir dos cruzamentos entre a primeira linha e a primeira coluna representam composições únicas que não fazem parte da base de dados original. Esta arquitetura é capaz de combinar vetores latentes de diferentes classes e produzir uma imagem distinta, mesmo sem ter sido treinada com exemplos específicos de pessoas com tais características.

A força do espaço latente está em sua capacidade de capturar padrões complexos, permitindo uma ampla variedade de operações, desde a geração de novas imagens até a manipulação de atributos específicos em conteúdo já existente. Uma técnica particularmente interessante nesse contexto é a inversão de GAN, que permite, a partir de uma imagem real, e descobrir como ela seria representada no espaço latente dessa GAN, como ilustrado a Figura 13.

Também na Figura 13, é observado o processo inverso de uma GAN, ou seja, diferentemente do processo convencional de amostragem e geração usando um gerador

treinado G , a inversão de GAN mapeia uma imagem real x para o espaço latente e obtém o código latente z^* . A imagem reconstruída x^* é então obtida por $G(z^*)$. Variando o código latente z^* em diferentes direções interpretáveis, como $z^* + n_1$ e $z^* + n_2$, onde n_1 e n_2 representam a idade e o sorriso no espaço latente, respectivamente. Assim, é possível editar o atributo correspondente da imagem real. Os resultados de reconstrução da imagem da Figura 9 são apresentados no artigo (Xia *et al.*, 2022).

2.6.2 Interpolação entre Vetores Latentes

A interpolação entre vetores latentes é uma das técnicas mais reveladoras para entender o comportamento do modelo generativo das GANs. Essa abordagem permite mapear trajetórias no espaço latente, gerando sequências de imagens que transitam continuamente entre dois pontos codificados (Goodfellow Et Al., 2014). Na prática, quando aplicada a modelos como as DCGAN (Radford, Metz e Chintala, 2016), essa técnica não apenas demonstra a coerência topológica do espaço aprendido, mas também revela relações semânticas entre características distintas.

A equação objetiva (6) representa interpolação linear, ele opera através de uma média ponderada entre dois vetores latentes z_1 e z_2 usando um parâmetro $\alpha \in [0,1]$. Embora computacionalmente eficiente, esse método apresenta uma limitação crítica em espaços latentes gaussianos: ao seguir um caminho retilíneo, a interpolação frequentemente atravessa regiões de baixa densidade probabilística, gerando amostras distorcidas ou semanticamente inconsistentes White (2016). Um exemplo concreto desse fenômeno pode ser observado em experimentos com GANs treinadas em datasets de rostos, onde a interpolação linear ocasionalmente produz faces borradas ou com artefatos anatômicos nas posições intermediárias.

$$z_{\text{interp}} = (1 - \alpha)z_1 + \alpha z_2 \quad (6)$$

A Figura 14, é uma Interpolação linear entre pares de imagens embutidas no espaço latente da *StyleGAN*. Cada linha representa uma transição entre uma imagem real (à esquerda) e outra (à direita). As imagens intermediárias são geradas por interpolação linear entre os respectivos vetores latentes (Abdal, Qin e Wonka, 2019a).

Figura 14 - Interpolação linear no espaço latente do *StyleGAN*



Adaptado de (Abdal, Qin e Wonka, 2019a)

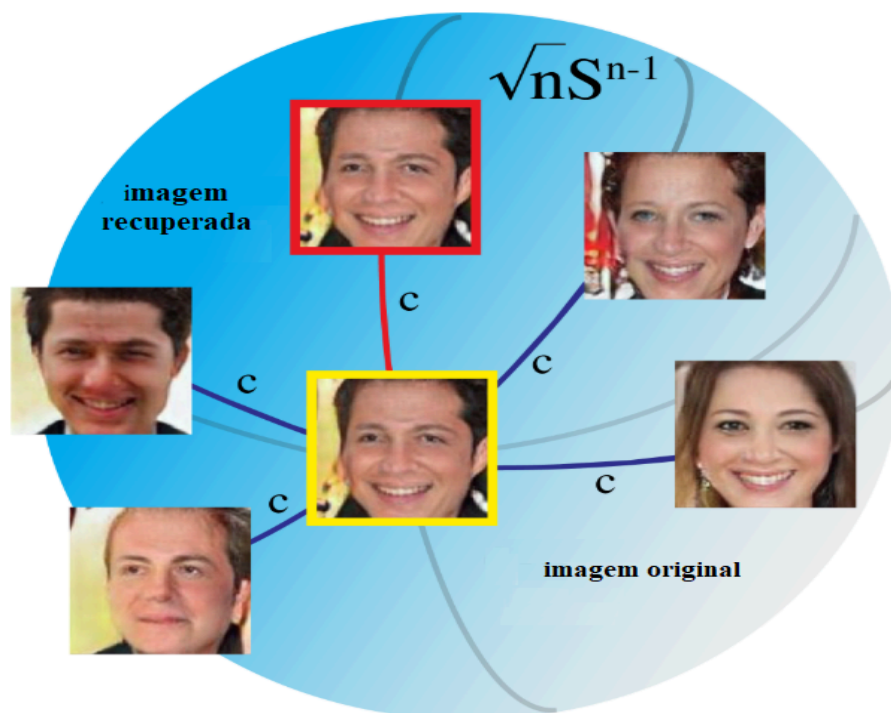
A interpolação esférica, mantém os vetores interpolados na variedade de alta densidade do espaço latente. Desenvolvida originalmente para animação 3D (Shoemake, 1985), a interpolação esférica calcula trajetórias ao longo do arco de um hipersfera, respeitando a distribuição estatística subjacente. Visualmente, isso se traduz em transições mais suaves e semanticamente consistentes - um efeito particularmente evidente em modelos como *StyleGAN* (Karras, Laine e Aila, 2019), onde a interpolação esférica entre vetores de estilo produz variações graduais em características como idade, expressão facial ou iluminação

$$z_{\text{interp}} = \left[\frac{\sin((1-\alpha)\theta)}{\sin(\theta)} \right] \cdot z_1 + \left[\frac{\sin(\alpha\theta)}{\sin(\theta)} \right] \cdot z_2 \quad (7)$$

A Figura 15, representa visualmente o espaço latente esférico de um modelo generativo como uma esfera (formalmente descrita por $\sqrt{n}S^{n-1}$), onde a maioria dos vetores latentes se concentra devido às propriedades das distribuições *Gaussiana* em alta dimensão. A imagem original (contorno amarelo) é gerada a partir de um vetor z_0 , enquanto a imagem recuperada (contorno vermelho) corresponde a um vetor z_1 obtido por otimização inversa.

Apesar da distância entre z_0 e z_1 , ambas produzem imagens semanticamente similares, revelando que direções específicas no espaço latente preservam o conteúdo da imagem. A curva C que as conecta representa uma trajetória geodésica privilegiada, ao longo da qual pequenos ajustes melhoram progressivamente o realismo sem alterar atributos fundamentais (Wen *et al.*, 2021).

Figura 15 - Interpolação slerp no espaço latente do styleGAN



Adaptado de Wen et al., (2021)

Essa estrutura esférica ilustra dois conceitos-chave: primeiro, que o espaço latente possui regiões semanticamente organizadas, segundo, que é possível navegar nessas direções para otimizar resultados sem modificar a rede. Essa abordagem permite extrair maior desempenho de redes existentes, tornando modelos generativos mais acessíveis sem sacrificar qualidade.

2.6.3 Transição Entre Vetores Latente

A transição entre vetores latentes por meio de otimização direta constitui uma técnica fundamental para a exploração controlada do espaço latente em modelos generativos. Diferentemente da interpolação tradicional, que opera por meio de combinações lineares ou esféricas entre vetores, essa abordagem implementa um processo de transformação não linear, guiado por critérios de otimização (Zhu *et al.*, 2018).

O método parte de um vetor latente inicial (geralmente extraído de uma imagem de referência por técnicas de inversão de GAN) que é ajustado iterativamente para aproximar-se de uma imagem-alvo específica. Esse ajuste é realizado com o uso de algoritmos de otimização, sendo o otimizador *Adam* (Kingma e Ba, 2017) amplamente empregado pela sua eficiência em cenários de otimização com gradientes ruidosos, como é o caso da reconstrução de imagens a partir do espaço latente. Durante esse processo, busca-se preservar a validade topológica do espaço latente, a coerência semântica das características transformadas e a diversidade estrutural aprendida pelo modelo generativo (Abdal, Qin e Wonka, 2019b).

Particularmente em aplicações de classificação de padrões SPR, essa técnica possibilita a síntese de imagens intermediárias entre classes que preservam validade física, contribuindo para a ampliação de conjuntos de treinamento, o refinamento de fronteiras de decisão entre diferentes classes e a análise quantitativa de transições morfológicas. A geração de uma imagem com características híbridas, porém fisicamente plausíveis, mostra-se especialmente útil para estudos de sensibilidade e calibração em sistemas SPR, conforme evidenciado nos experimentos da Seção 4.2.

2.7 Técnicas de Avaliação dos Resultados

A avaliação dos resultados gerados por modelos generativos exige a aplicação de métricas quantitativas e qualitativas que mensuram a fidelidade, coerência e utilidade prática das imagens sintetizadas. Essa etapa é fundamental para validar a qualidade das amostras, sobretudo em aplicações científicas, como na análise de respostas ópticas em sensores SPR, onde a preservação de características físicas específicas é crucial.

Foram utilizadas as métricas, largura total na metade da altura, largura da curva, energia do sinal, assimetria e ponto de mínimo. A curva SPR, obtida da imagem refletida pelo

sensor, permite identificar o ponto de ressonância por métodos como centróide ou ajuste polinomial. A FWHM, a largura da curva (L_C) e a assimetria (A) são sensíveis a variações no índice de refração e na espessura da camada sensível, sendo úteis na comparação entre amostras. Detalhes dessas métricas encontram-se na Seção 2.4.

Além disso, métricas internas de desempenho em arquiteturas GAN, como as funções de perda do gerador e do discriminador, indicam a estabilidade e o equilíbrio durante o treinamento, conforme evidenciado nos experimentos da Seção 4.1.1.

Por fim, A métrica discrepância máxima de médias (em inglês, *Maximum Mean Discrepancy* – MMD), fundamentada em kernels no espaço de hilbert com núcleo reproduzível (RKHS), é eficaz na detecção de diferenças não lineares entre distribuições (Gretton et al., 2012). conforme evidenciado na Seção 3.4.1, o erro quadrático médio (*Mean Squared Error* – MSE) é amplamente utilizado para avaliar a qualidade de reconstruções, embora seja sensível a outliers (Hastie, Friedman; e Tibshirani, 2001).

A distância euclidiana é uma métrica clássica utilizada para medir a similaridade entre dois pontos em um espaço vetorial. Trata-se da forma mais comum de distância geométrica, baseada na norma L2, e é amplamente empregada em tarefas de agrupamento (*clustering*), redução de dimensionalidade e análise de similaridade em espaços latentes. Em contextos envolvendo modelos generativos, como GANs, essa métrica permite comparar vetores latentes ou representações intermediárias, avaliando o quão próximos estão dois pontos no espaço gerado pelo modelo (Tan, Steinbach e Kumar, 2005).

Em termos matemáticos, a distância euclidiana entre dois vetores $x = (x_1, x_2, \dots, x_n)$ e $y = (y_1, y_2, \dots, y_n)$, ambos pertencentes ao espaço R^n , é dada por:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (8)$$

Essa métrica assume que todas as dimensões do espaço possuem a mesma importância e escala, sendo portanto sensível à magnitude e à presença de *outliers*. Por esse motivo, seu uso geralmente é precedido por uma etapa de normalização dos dados, especialmente em aplicações de alto custo computacional ou em espaços de alta dimensionalidade. Ainda assim, sua simplicidade e interpretabilidade a tornam uma das principais ferramentas para análise

quantitativa de proximidade entre amostras, especialmente na avaliação de variações em representações geradas por redes neurais (Aggarwal, 2015).

Para a avaliação qualitativa, ferramentas visuais são essenciais para análise interpretativa dos resultados. Histogramas permitem examinar a distribuição de atributos morfológicos e identificar possíveis desvios ou assimetrias entre conjuntos reais e gerados. A técnica t-SNE (*t-Distributed Stochastic Neighbor Embedding*) é particularmente útil para visualizar dados de alta dimensão em duas ou três dimensões, preservando as estruturas locais e revelando agrupamentos e sobreposições que não seriam detectados por métricas numéricas tradicionais (Van der Maaten e Hinton, 2008). O método funciona convertendo as distâncias entre pontos no espaço original em probabilidades de similaridade, tanto no espaço de alta quanto de baixa dimensão. Em seguida, o t-SNE busca uma representação em baixa dimensão que preserve essas probabilidades, minimizando a divergência de Kullback–Leibler entre as distribuições. Dessa forma, pontos próximos no espaço original tendem a permanecer próximos na visualização, revelando agrupamentos e estruturas locais que seriam difíceis de identificar por métodos lineares tradicionais.

Parâmetros como a perplexidade, que controla o número efetivo de vizinhos considerados na vizinhança local, e a definição de uma semente aleatória (*random seed*), garantem o equilíbrio entre estruturas locais e globais e a reprodutibilidade dos resultados.

A seleção do valor de perplexidade para o t-SNE foi realizada combinando critérios visuais e quantitativos. A Figura 39 ilustra a relação entre diferentes valores de perplexidade e a divergência de Kullback–Leibler, utilizada como métrica de qualidade da projeção.

$$D_{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} \quad (9)$$

A divergência de Kullback–Leibler (KL) mede o quanto uma distribuição de probabilidade Q se afasta de uma distribuição de referência P . Baseia-se na soma ponderada por $P(x)$ do termo $\log\left(\frac{P(x)}{Q(x)}\right)$, que indica a diferença entre as probabilidades atribuídas por cada distribuição a um mesmo evento. Esse valor representa a quantidade de informação perdida ao se usar Q no lugar de P . Quanto menor a divergência KL, mais próxima está Q da distribuição real P . Essa medida é amplamente utilizada em aprendizado de máquina,

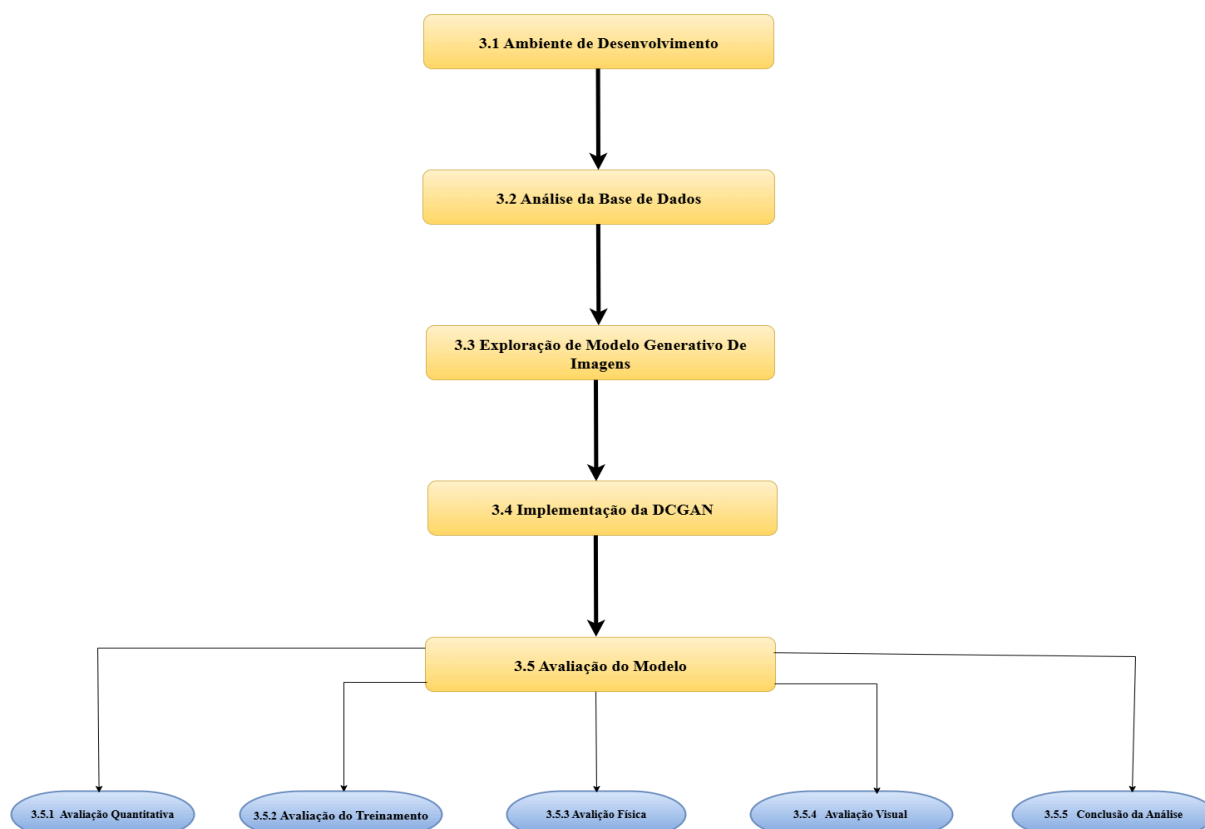
modelos probabilísticos e métodos de redução de dimensionalidade (Kullback & Leibler, 1951).

3. METODOLOGIA

Este capítulo apresenta a metodologia da pesquisa, estruturada em cinco etapas: (1) Ambiente de Desenvolvimento Seção 3.1, com a preparação de recursos e ferramentas; (2) Análise da Base de Dados Seção 3.2, envolvendo caracterização e pré-processamento; (3) Exploração de Modelos Generativos de Imagens Seção 3.3; (4) Implementação da DCGAN

Seção 3.4, com configuração e adaptação do modelo; e (5) Avaliação do Modelo Desenvolvido Seção 3.5, incluindo análise quantitativa, acompanhamento do treinamento, avaliação física da curva de intensidade, avaliação visual e síntese dos resultados. Essa abordagem garantiu uma análise abrangente do desempenho do modelo em todas as fases.

Figura 16 - Fluxograma da metodologia deste trabalho



3.1 Ambiente de Desenvolvimento

Como parte essencial da implementação da metodologia proposta, foram utilizados dois ambientes computacionais complementares para o desenvolvimento, treinamento e testes da DCGAN. A escolha desses ambientes visou atender às diferentes demandas computacionais das etapas do projeto, garantindo eficiência e reprodutibilidade dos experimentos.

As implementações principais e os treinamentos que exigiram maior capacidade de processamento foram realizados em uma estação de trabalho localizada no laboratório de pesquisa da Universidade Federal Rural do Semiárido (UFERSA). Esta estação é equipada

com sistema operacional *Windows*, processador *quad-core*, GPU (em inglês, *Graphics Processing Unit*) NVIDIA *GeForce GTX 1060*, 16 *gigabytes* de memória RAM (em inglês, *Random Access Memory*) e SSD (*Solid State Drive*) de 500 *gigabytes*.

O desenvolvimento do modelo foi conduzido no Visual Studio, um ambiente de desenvolvimento integrado (em inglês, *Integrated Development Environment* - IDE) da *Microsoft*. A implementação utilizou a linguagem *Python 3.9* e bibliotecas como *PyTorch*, *Torchvision*, *NumPy* e *Matplotlib*, com suporte ao dispositivo CUDA (em inglês, *Compute Unified Device Architecture*), viabilizando o uso eficiente da GPU por meio do *PyTorch*.

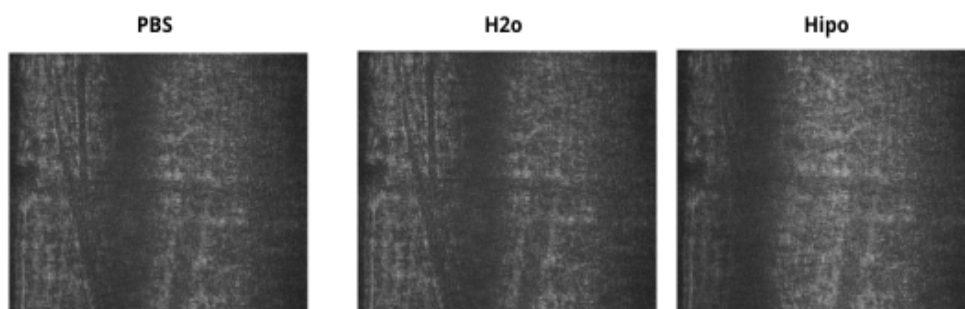
Para tarefas de menor carga, como testes preliminares, ajustes de código e visualização de resultados, foi utilizado o ambiente *Google Colab (Collaboratory)*, que oferece acesso gratuito a GPUs e uma interface interativa baseada em *Jupyter Notebook*. A utilização combinada desses dois ambientes permitiu uma execução otimizada, ajustando os recursos computacionais às necessidades específicas de cada etapa do desenvolvimento (Paszke *et al.*, 2019).

3.2 Análise da Base de Dados

O conjunto de dados utilizado nesta pesquisa foi apresentado por (Sales, 2024), que consiste em 600 imagens simuladas de sensores SPR em tons de cinza (monocromáticas), com resolução de 1280×1024 pixels, igualmente distribuídas em cinco classes correspondentes a índices de refração específicos: 1,33; 1,335; 1,34; 1,345 e 1,35. Cada classe representa uma substância diferente analisada pelo sensor.

O desenvolvimento desta base de dados iniciou-se com a aquisição de imagens experimentais por meio do sensor SPR, configurado com o Prisma Polimérico para Aplicações Biológicas (PPBIO), conforme ilustrado na Figura 17. Foram capturadas imagens de três substâncias distintas: tampão de fosfato (PBS), água destilada (H_2O) e hipoclorito de sódio (HIPO), cada uma apresentando características ópticas específicas em função de seus diferentes índices de refração. Essas imagens da Figura 17 serviram como referência fundamental para todo o processo subsequente de modelagem e simulação, Oliveira *et al.*, 2019.

Figura 17 – Imagens do sensor PPBIO .

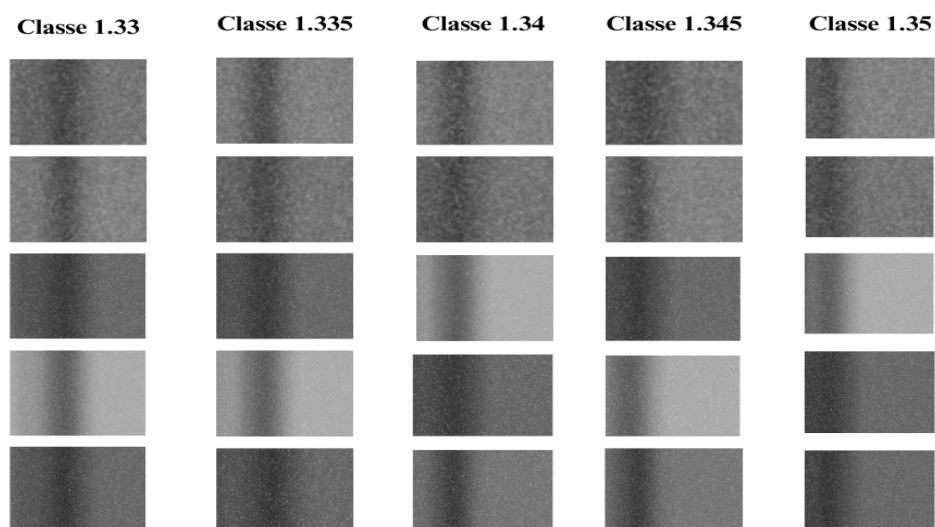


Fonte: Adaptada de (Oliveira et al., 2019)

Nas imagens experimentais da Figura 17, observa-se uma textura em tons de cinza com faixas claras e escuras distribuídas verticalmente. A região mais escura, chamada vale de ressonância, apresenta um deslocamento angular característico conforme o índice de refração (RIU) da substância analisada. Essa variação na posição do vale - ilustrada na comparação entre H₂O (RIU=1,330), PBS (RIU=1,331) e HIPO (RIU=1,345) - constitui um parâmetro óptico sensível que permite a identificação direta de diferentes substâncias pela análise das imagens SPR.

Na Figura 18, são apresentados exemplos das imagens da base de dados utilizada por Oliveira et al. (2025) em seu projeto. Essa base foi gerada por meio das equações de Fresnel e técnicas de PDI (Processamento Digital de Imagens). Cada coluna corresponde a uma classe específica, evidenciando as diferenças sutis nas características visuais associadas a cada valor de índice.

Figura 18 – Imagens da base de dados



Fonte: Adaptada de (Sales, 2024)

A partir das imagens reais, obtidas por meio de instrumentação SPR Figura 17, (Sales, 2024), realizou uma análise detalhada para extração das curvas SPR correspondentes a cada substância. Esse processo envolveu a identificação de características-chave como: a posição angular do mínimo de refletividade (ponto de ressonância), Largura Total na Metade da Altura (em inglês, Full Width at Half Maximum – FWHM), e a morfologia de cada curva, incluindo possíveis assimetrias. Essas informações constituíram a base para a criação de um modelo matemático fiel à realidade experimental.

Na primeira etapa da modelagem, as propriedades essenciais das curvas SPR foram extraídas das imagens reais, com destaque para: a localização precisa do vale de ressonância (ponto de mínima refletividade), a profundidade característica deste vale, a largura da curva e suas particularidades morfológicas, como assimetrias. Essas características foram quantificadas através do modelo matemático de Fresnel em sua forma matricial (Born e Wolf, 1999), que descreve com precisão a relação entre a refletividade da luz, o ângulo de incidência e os índices de refração dos materiais analisados.

Depois, foi o estabelecimento da relação pixel-ângulo, obtida através de calibração experimental. Essa correlação permitiu traduzir as posições angulares teóricas em coordenadas de pixel específicas, criando um mapeamento preciso onde cada coluna de pixels

na imagem correspondia a um ângulo determinado de incidência luminosa. A faixa mais escura em cada imagem experimental, correspondente ao ponto de ressonância, serviu como referência fundamental para essa calibração.

A integração dessas características físicas das curvas SPR com a relação pixel-ângulo possibilitou a conversão dos dados teóricos em representações visuais precisas. Através desse processo, as curvas teóricas de refletividade em função do ângulo foram transformadas em perfis de refletividade versus pixel, dando origem a imagens de Fresnel monocromáticas com resolução de 1280×1024 pixels. Cada imagem gerada representava fielmente o padrão óptico característico de uma substância específica, preservando tanto os princípios físicos fundamentais quanto as particularidades observadas experimentalmente.

Foi utilizado um fatiamento por densidade da curva SPR. Essa técnica envolveu a atribuição de valores de pixel (tons de cinza) de acordo com a densidade de refletividade em diferentes regiões da curva, gerando quatro padrões distintos de mapeamento: C1, com concentração máxima de tons escuros (0-50) no vale de ressonância e transição abrupta para tons claros (200-255) nas bordas; C2, caracterizado por distribuição uniforme de tons médios (100-150) ao longo da curva e leve escurecimento na região de ressonância; C3, que apresenta gradiente suave desde o vale de ressonância (0-100) até as bordas (150-255); e C4, com alto contraste entre vale estreito (0-30) e bordas muito claras (220-255), simulando condições experimentais ideais.

Para aprimorar o realismo das imagens simuladas, aplicou-se um sofisticado processamento digital de imagens segundo Oliveira et al. (2025), implementando sequencialmente: suavização por Gaussian Blur, adição controlada de ruído Sal e Pimenta, aplicação de filtro Máximo e novo estágio de Gaussian Blur. Essas operações foram cuidadosamente ajustadas em diferentes configurações e regiões da imagem (incluindo tratamento global, faixas horizontais completas ou parciais), gerando variações realistas que reproduziam as imperfeições naturais dos sistemas ópticos reais.

A partir destas técnicas, (Sales, 2024) construiu o dataset composto por 600 imagens com resolução de 1280×1024 pixels, sendo 120 em cada uma das 5 classes, as quais são relacionadas com o índice de refração. As imagens utilizadas neste trabalho foram obtidas a partir de um repositório público do (GESyCA) no GitHub. Os dados estão organizados em

dois conjuntos de dados, o primeiro é o conjunto de treino, disponível em https://github.com/GESyCA/SPR-GAN/tree/main/dataset_treino, e o segundo conjunto é o de teste, disponível em https://github.com/GESyCA/SPR-GAN/tree/main/dataset_teste. Ambos os conjuntos foram acessados em 7 de agosto de 2025. Vamos nos referir às imagens obtidas deste *dataset* como “imagens reais” para distinguir das imagens “geradas” pela arquitetura implementada neste trabalho.

3.3 Exploração de Modelos Generativos de Imagens

Neste trabalho, foi realizada uma análise exploratória inicial com o objetivo de identificar o modelo generativo mais adequado à tarefa de geração de imagens sintéticas do sensor SPR PPBIO. Foram consideradas três abordagens principais com base na literatura: Autoencoders Variacionais (VAEs), modelos de difusão e redes adversariais generativas (GANs), incluindo variantes como CGAN, WGAN e DCGAN.

Os principais critérios para a escolha do modelo foram: a capacidade de gerar imagens sintéticas com qualidade visual satisfatória, sem a necessidade de rótulos supervisionados, e com baixo custo computacional. Inicialmente, os modelos VAE e os modelos de difusão foram descartados por limitações conhecidas. Os VAEs tendem a produzir imagens com baixa nitidez, devido à compressão excessiva do espaço latente (Kingma; Welling, 2014). Já os modelos de difusão, embora gerem imagens de excelente qualidade, apresentam um custo computacional extremamente elevado (Dhariwal e Nichol, 2021), incompatível com os recursos disponíveis neste estudo.

Entre as variantes de GANs, a CGAN foi descartada por exigir rótulos supervisionados durante o treinamento, o que contraria o objetivo de geração não supervisionada. A WGAN, embora mais estável por utilizar a distância de *Wasserstein* como função de perda (Arjovsky, Chintala e Bottou, 2017b), demanda uma implementação mais complexa e custosa, incluindo múltiplas atualizações do discriminador para cada atualização do gerador e penalizações de gradiente para garantir a suavidade da função. Diante disso, optou-se por não utilizá-la, considerando a simplicidade e o tamanho das imagens empregadas em 224×224 pixels em tons de cinza.

Com base na literatura, a DCGAN demonstrou ser uma arquitetura adequada aos objetivos deste trabalho. Para validar essa escolha, foram realizados testes práticos, como geração de imagens, operações com vetores, interpolação e transição com o conjunto de dados específico da pesquisa. Essa etapa demandou adaptações, incluindo ajustes de hiperparâmetros, modificações nas arquiteturas do gerador e do discriminador, e alterações no pré-processamento dos dados. Após 50 épocas de treinamento, o modelo atingiu convergência em aproximadamente 1 a 2 horas, assegurando estabilidade no processo, viabilidade computacional e a geração de resultados satisfatórios, conforme apresentado no capítulo 4.

3.4 Implementação da Arquitetura Escolhida

Este projeto apresenta a implementação de uma rede generativa adversarial profunda, com foco na geração de imagens simuladas de sensores SPR. A escolha da arquitetura DCGAN baseia-se na proposta original de Radford, Metz e Chintala, (2016), que demonstraram seu potencial em tarefas de geração de imagens por meio de aprendizado não supervisionado. Como ponto de partida para a implementação, utilizou-se a base de código disponibilizada por (Barretto e Attux, 2019).

A partir dessa código, foram realizadas diversas modificações, incluindo ajustes nas arquiteturas das redes geradora e discriminadora, aplicação da técnica de t-SNE para visualização das posições das imagens no espaço latente, implementação de transições entre imagens e alguns ajustes nos parâmetros. O código ficou estruturado de forma modular, clara e didática, com cada etapa cuidadosamente desenvolvida para garantir o correto funcionamento e a adaptabilidade do modelo aos objetivos deste trabalho.

3.4.1 Preparação dos Dados

O processo inicia-se com a importação das bibliotecas essenciais para a construção da rede neural, utilizando o *framework PyTorch* como base principal. São empregados módulos auxiliares como *torchvision*, *matplotlib*, *numpy* e *torch.nn*, que viabilizam desde o carregamento e a transformação das imagens até a definição das camadas e o treinamento da rede. Também são utilizados recursos para divisão dos dados e carregamento eficiente em mini-batches. No ambiente do *Google Colab*, o *Google Drive* é montado para possibilitar o acesso a arquivos como modelos treinados e resultados salvos. O dispositivo de execução é

definido como a CPU, garantindo que o código possa ser executado mesmo na ausência de GPU.

Nesta etapa, foi definida a preparação dos dados, como o caminho para os diretórios que contêm as imagens simuladas de SPR, separadas em conjuntos de treinamento e teste. As imagens passam por uma sequência de transformações: conversão para escala de cinza e redimensionamento de 1280×1024 pixels para 224×224 pixels, com o objetivo de reduzir a complexidade computacional do treinamento. Embora esse redimensionamento, feito com alteração da proporção original, possa causar uma leve perda de qualidade visual e distorção em algumas regiões periféricas, os padrões visuais centrais relacionados ao fenômeno de ressonância de plásmons foram preservados de forma satisfatória, garantindo a validade da representação física no processo de geração.

Também houve uma conversão das imagens para tensor e normalização para o intervalo $[-1, 1]$. Embora normalizações em intervalos como $[0, 1]$ ou $[0, 255]$ sejam comuns em outras aplicações, no contexto específico da DCGAN, a normalização para $[-1, 1]$ é considerada uma boa prática recomendada na literatura, conforme apresentado no trabalho de (Radford, Metz e Chintala, 2016).

Para facilitar o acesso ao nome dos arquivos durante o treinamento ou a análise de resultados, foi implementada a classe personalizada *CustomImageFolder*, que estende a *ImageFolder* do *torchvision* e retorna, além da imagem, o nome do arquivo. A partir dessa classe, são construídos os conjuntos de dados (*datasets*) de treinamento e teste, que são carregados nos *DataLoaders* do *PyTorch*: *trainloader* e *testloader*. O conjunto de treinamento é composto por 540 imagens (108 por classe), enquanto o conjunto de teste contém 60 imagens (12 por classe).

3.4.2 Arquitetura da DCGAN

Sobre a arquitetura temos a rede discriminadora (*class Discriminator*) composta por seis camadas convolucionais sequenciais. Ela recebe como entrada imagens em escala de cinza com dimensão $1 \times 224 \times 224$ e reduz progressivamente essa dimensionalidade até produzir uma saída escalar. A primeira camada reduz para 112×112 com d_n filtros; a segunda, para 56×56 com $2 \times d_n$ filtros; a terceira, para 28×28 com $4 \times d_n$ filtros; a quarta,

para 14×14 com $8 \times d_n$ filtros; e a quinta, para 7×7 com $16 \times d_n$ filtros. Todas essas camadas utilizam filtros 4×4 , *stride* 2 e *padding* 1, e não utilizam viés (*bias=False*). Após cada convolução, é aplicada a função de ativação *Leaky ReLU*.

A Figura 19 apresenta a arquitetura da rede discriminadora, detalhando o tipo de cada camada (*Layer type*), as dimensões das saídas em cada etapa (*Output Shape*) e o número de parâmetros treináveis por camada (*Param*). A rede é composta por camadas convolucionais (*Conv2d*), totalizando 11.192.320 parâmetros treináveis e nenhum parâmetro não treinável. A estimativa de uso de memória inclui 0,19 megabytes (MB) para a entrada, 11,87 megabytes (MB) para o processo de propagação direta e reversa, e 42,70 megabytes (MB) para os parâmetros, resultando em um total aproximado de 54,75 *megabytes* (MB).

Figura 19 - Resumo detalhado da arquitetura do discriminador

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 112, 112]	1,024
Conv2d-2	[-1, 128, 56, 56]	131,072
Conv2d-3	[-1, 256, 28, 28]	524,288
Conv2d-4	[-1, 512, 14, 14]	2,097,152
Conv2d-5	[-1, 1024, 7, 7]	8,388,608
Conv2d-6	[-1, 1, 1, 1]	50,176
Total params: 11,192,320		
Trainable params: 11,192,320		
Non-trainable params: 0		
Input size (MB): 0.19		
Forward/backward pass size (MB): 11.87		
Params size (MB): 42.70		
Estimated Total Size (MB): 54.75		

A sexta camada aplica uma convolução com *kernel* 7×7 e *stride* 1, reduzindo a saída para 1×1 . Em seguida, é aplicada a função de ativação *sigmoid*, que converte a saída para um valor entre 0 e 1, representando a probabilidade da imagem ser real. Essa arquitetura permite que o discriminador aprenda a diferenciar entre imagens reais e sintéticas.

Em seguida, tem-se a rede geradora (classe generator) composta por seis camadas de convolução transposta, cuja finalidade é transformar um vetor latente de dimensão $z_i \times 1 \times 1$

em uma imagem final com dimensão $1 \times 224 \times 224$. A primeira camada projeta o vetor para uma *feature map* com dimensão $g_n \times 16 \times 7 \times 7$, seguida por normalização em lote (*BatchNorm*) e ativação *Leaky ReLU*.

As camadas seguintes aumentam progressivamente a resolução da imagem: segunda camada $g_n \times 8 \times 14 \times 14$, terceira ($g_n \times 4 \times 28 \times 28$), quarta $g_n \times 2 \times 56 \times 56$, quinta ($g_n \times 112 \times 112$) e sexta ($1 \times 224 \times 224$). Todas as camadas, exceto a última, são seguidas por *BatchNorm* e *Leaky ReLU*. Essa arquitetura permite que a rede aprenda a gerar imagens sintéticas visualmente similares às reais.

A Figura 20 apresenta a arquitetura da rede geradora, detalhando o tipo de cada camada (*Layer type*), as dimensões das saídas em cada etapa (*Output Shape*) e o número de parâmetros treináveis por camada (*Param*). A rede é composta por camadas de transposição convolucional (*ConvTranspose2d*) intercaladas com camadas de normalização (*BatchNorm2d*), totalizando 16.163.712 parâmetros treináveis e nenhum parâmetro não treinável. A estimativa de uso de memória inclui 24,12 megabytes (MB) para o processo de propagação direta e reversa, e 61,66 *megabytes* (MB) para os parâmetros, resultando em um total aproximado de 85,78 *megabytes* (MB).

Figura 20 - Resumo detalhado da arquitetura do gerador

Layer (type)	Output Shape	Param #
ConvTranspose2d-1	[-1, 1024, 7, 7]	5,017,600
BatchNorm2d-2	[-1, 1024, 7, 7]	2,048
ConvTranspose2d-3	[-1, 512, 14, 14]	8,388,608
BatchNorm2d-4	[-1, 512, 14, 14]	1,024
ConvTranspose2d-5	[-1, 256, 28, 28]	2,097,152
BatchNorm2d-6	[-1, 256, 28, 28]	512
ConvTranspose2d-7	[-1, 128, 56, 56]	524,288
BatchNorm2d-8	[-1, 128, 56, 56]	256
ConvTranspose2d-9	[-1, 64, 112, 112]	131,072
BatchNorm2d-10	[-1, 64, 112, 112]	128
ConvTranspose2d-11	[-1, 1, 224, 224]	1,024
Total params: 16,163,712		
Trainable params: 16,163,712		
Non-trainable params: 0		
Input size (MB): 0.00		
Forward/backward pass size (MB): 24.12		
Params size (MB): 61.66		
Estimated Total Size (MB): 85.78		

3.4.3 Funções Auxiliares

Além da arquitetura principal do modelo, existem funções auxiliares que desempenham papéis essenciais na geração de imagens Figuras 21 a 27.

A função *generateLatentPoints* gera pontos no espaço latente para treinar um modelo, como um DCGAN. Ela cria um tensor aleatório *z* com valores amostrados de uma distribuição normal padrão. O tensor tem o formato (*batch_size*, *z_i*, 1, 1) e é alocado no dispositivo (CPU ou GPU) definido por *device*. Esse vetor latente é usado como entrada para o gerador.

Figura 21 - Função de criação de pontos no espaço latente

```
[ ] def generateLatentPoints(batch_size, z_i):
    z = torch.randn(batch_size, z_i, 1, 1, device=device)
    return z
```

A função *generateLabels* é usada no treinamento do discriminador para fornecer os rótulos verdadeiros ou falsos, com ou sem suavização, dependendo do caso. Ela cria tensores de rótulos verdadeiros (1) ou falsos (0), com opção de suavização (*label smoothing*). Quando a suavização está ativada (*smooth=True*), os rótulos verdadeiros são valores aleatórios entre

0.8 e 1.0, e os falsos entre 0.0 e 0.2, o que ajuda a estabilizar o treinamento. Os rótulos são gerados no formato $(batch_size, 1, 1, 1)$ e no dispositivo especificado por device.

Figura 22 - Geração de rótulos com ou sem suavização para o discriminador

```
def generateLabels(arg, batch_size, smooth=False):
    random = torch.rand(batch_size, 1, 1, 1, device=device)
    ones = torch.ones(batch_size, 1, 1, 1, device=device)
    zeros = torch.zeros(batch_size, 1, 1, 1, device=device)

    # label smoothing is on
    if smooth:
        # true labels are in range [0.8, 1.2]
        if arg == 'True':
            labels = 0.2 * random + 0.8

        # false labels are in range [0.0, 0.2]
        if arg == 'False':
            labels = 0.2 * random

    # label smoothing is off
    else:
        if arg == 'True':
            # true labels are 1s
            labels = ones
        if arg == 'False':
            # false labels are 0s
            labels = zeros

    return labels
```

A função *initWeights* inicializa os pesos de camadas em uma rede neural, geralmente usada em modelos como DCGAN. Ela aplica uma inicialização normal com média 0 e desvio padrão 0.02 para camadas convolucionais (*Conv*). Para camadas de normalização em lote (*BatchNorm*), os pesos são inicializados com média 1 e desvio padrão 0.02, enquanto os vieses (bias) são definidos como 0. Essa inicialização ajuda a estabilizar e acelerar o treinamento.

Figura 23 - Inicialização de pesos para camadas convolucionais e de normatizações

```
[ ] def initweights(m):
    classname = m.__class__.__name__
    if classname.find('Conv') != -1:
        nn.init.normal_(m.weight.data, 0.0, 0.02)
    elif classname.find('BatchNorm') != -1:
        nn.init.normal_(m.weight.data, 1.0, 0.02)
        nn.init.constant_(m.bias.data, 0)
```

No que diz respeito à visualização dos resultados, a função *printImages* exibe uma sequência linear de imagens geradas, removendo os eixos para facilitar a análise visual dos padrões aprendidos pela rede.

Figura 24 - Visualização de imagens em escala de cinza

```

# function to print sample images
def printImages(images):
    for i in range(len(images)):
        ax = plt.subplot(1,len(images),i+1)
        ax.set_axis_off()
        ax = plt.imshow(images[i].view(224,224).cpu().numpy(), cmap='gray')
    plt.show()

```

Já a função *printImageMatrix* organiza as imagens em uma matriz bidimensional, sendo ideal para observar variações geradas ou transições ao longo da interpolação no espaço latente o recurso fundamental para compreender a continuidade e a semântica aprendida pelo modelo.

Figura 25 - Visualização em matriz de imagens escala de cinza

```

def printImageMatrix(imageData, title):
    numRows = len(imageData)

    n=0
    fig = plt.figure(1)
    fig.suptitle(title)
    for line in imageData:
        for i in range(len(line)):
            ax = fig.add_subplot(numRows,len(line),n+1)
            ax.set_axis_off()
            ax = plt.imshow(line[i].view(224, 224).cpu().numpy(), cmap='gray')
            n+=1
    plt.savefig('image:' + str(title) + '.png')
    plt.show()

```

A função *printLosses* plotar as curvas de perda tanto da rede geradora quanto da discriminadora ao longo das épocas, permitindo o monitoramento contínuo da estabilidade do treinamento e identificação de possíveis colapsos de modo ou oscilações.

Figura 26 - Função para mostrar a perda do gerador e do discriminador durante o treinamento

```
# function to print model losses
def printLosses(d_loss, g_loss, save=False):
    plt.plot(d_loss, label='Discriminator Loss', c='blue')
    plt.plot(g_loss, label='Generator Loss', c='orange')
    plt.title('Training Losses')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    if save:
        plt.savefig('image:losses.png')
    plt.show()
```

A função *evaluateModel* avalia a qualidade do modelo gerador por meio da interpolação de vetores latentes (p1 e p2). A interpolação pode ser realizada de forma linear ou esférica, o que permite observar a transição suave entre diferentes pontos no espaço latente e, conseqüentemente, acompanhar a evolução da capacidade gerativa da rede em produzir imagens realistas e coerentes com a distribuição dos dados de entrada.

Figura 27 - Procedimento de avaliação do modelo gerador com diferentes modos de amostragem latente

```
# function to evaluate model
def evaluateModel(G, z_i, mode='random', p1=[], p2=[]):
    num_images = 10

    if mode == 'random':
        # generates random latent points
        z = generateLatentPoints(num_images, z_i)

    elif mode == 'linear':
        p1 = p1.reshape(z_i)
        p2 = p2.reshape(z_i)
        # linearly interpolates between the two points
        z = interpolateLatentPoints(G, p1, p2, num_images=num_images, mode=mode)

    elif mode == 'spherical':
        p1 = p1.reshape(z_i)
        p2 = p2.reshape(z_i)
        # spherically interpolates between the two points
        z = interpolateLatentPoints(G, p1, p2, num_images=num_images, mode=mode)

    z = z.reshape(num_images, z_i, 1, 1)
    # generate sample images
    G.eval()
    with torch.no_grad():
        sampleImages = G(z)

    return sampleImages
```

3.4.4 Funções de Geração de Novas Imagens

Funções criadas para gerar novas imagens a partir de imagens existentes, utilizando diferentes estratégias Figuras 28 a 31. Entre elas, destacam-se: a geração de imagens com base nas características combinadas de duas imagens; a interpolação linear e esférica entre vetores latentes; e, por fim, a criação de uma transição contínua entre duas imagens, resultando em várias imagens intermediárias.

Para gerar novas imagens semelhantes a uma imagem alvo específica, foi implementada a função *findLatentVector*. Essa função tem como objetivo encontrar um vetor latente que, ao ser passado pela rede geradora treinada, produza uma imagem visualmente próxima da imagem real fornecida. O processo inicia com a criação de um vetor latente aleatório, com dimensão compatível com a rede, e o uso do otimizador Adam para ajustá-lo gradualmente. A função de perda utilizada é o erro médio quadrático (MSE), que calcula a diferença entre a imagem gerada e a imagem alvo. Durante o processo, os gradientes são computados e usados para atualizar iterativamente o vetor latente.

O treinamento ocorre ao longo de várias épocas, com um critério de parada baseado na ausência de melhoria significativa na perda durante um número consecutivo de iterações, definido pelo parâmetro *patience*. Se não houver melhora após esse limite, o processo é interrompido para evitar ajustes desnecessários. Ao final, o vetor latente otimizado é retornado, representando a codificação mais próxima da imagem alvo dentro do espaço latente aprendido pela rede geradora. Esse vetor pode, então, ser utilizado para outras operações, como interpolações ou transformações, contribuindo para a análise e síntese de novas imagens.

Figura 28 - Função para construir uma imagem sintética baseado na imagem alvo

```
def findLatentVector(G, z_i, image, maxEpochs=5000, patience=5):
    z = torch.randn(1, z_i, 1, 1, device=device, requires_grad=True) # Cria um vetor latente inicial aleatório
    z_optimizer = optim.Adam([z], lr=0.01) # otimizador Adam para ajustar o vetor latente `z`
    criterion = nn.MSELoss(reduction='sum') # função de perda (loss function) do tipo MSE (Mean Squared Error).
    z_optimizer.zero_grad() # Zera os gradientes acumulados do otimizador.

    #Inicializa variáveis de controle:
    last_loss = 10000
    no_change = 0

    #Loop de treinamento que itera até `maxEpochs`
    for epoch in range(maxEpochs + 1):
        generated_image = G(z) # Gera uma imagem a partir do vetor latente `z`
        loss = criterion(generated_image, image) # Calcula a perda comparando a imagem gerada com a imagem alvo
        loss.backward() # Calcula os gradientes da perda em relação ao vetor latente `z`.
        z_optimizer.step() # Atualiza o vetor latente `z`
        z_optimizer.zero_grad() # Zera os gradientes acumulados no otimizador

        if last_loss - loss.item() < 0: # Verifica se a perda atual não melhorou significativamente
            no_change += 1
        else:
            no_change = 0

        if no_change >= patience: # Interrompe o treinamento se não houver melhoria por `patience`
            break

        last_loss = loss.item() # Atualiza `last_loss` com a perda atual para a próxima iteração

    return z.detach() # Retorna o vetor latente `z` ajustado, com os gradientes desativados
```

A função *gerar_imagem_soma_latentes* recebe um gerador pré-treinado, o caminho para um arquivo contendo dados de vetores latentes, os nomes de duas imagens específicas e uma função para exibição de imagens. Ela carrega os dados do arquivo, extrai os vetores latentes correspondentes às imagens indicadas, realiza a soma desses vetores e gera uma nova imagem a partir do vetor resultante usando o gerador. Além disso, a estrutura da função permite fácil adaptação para outras operações entre vetores latentes. Por exemplo, para realizar a subtração entre dois vetores latentes, basta alterar o operador de soma pelo de subtração.

Figura 29 - Soma de características de vetores latentes

```
def gerar_imagem_soma_latentes(G, caminho_arquivo, nome_img1, nome_img2, print_func):

    # Carrega os dados do arquivo
    dados_carregados = torch.load(caminho_arquivo, weights_only=False)

    # Seleciona os vetores latentes das imagens indicadas
    imagem_selecionada_1 = next((item for item in dados_carregados if item['image_index'] == nome_img1), None)
    vetor_latente_1 = imagem_selecionada_1['vetor_latente']

    imagem_selecionada_2 = next((item for item in dados_carregados if item['image_index'] == nome_img2), None)
    vetor_latente_2 = imagem_selecionada_2['vetor_latente']

    # Soma dos vetores latentes
    vetor_latente_soma = vetor_latente_1 + vetor_latente_2

    # Gera imagem a partir do vetor latente somado
    with torch.no_grad():
        G.eval()
        image_soma = G(vetor_latente_soma).cpu().numpy()

    # Exibe a imagem gerada
    print_func([image_soma])

    return image_soma
```

As funções linear (*lerp*) e esférica (*slerp*) (Figura 28) são utilizadas para realizar interpolações entre dois vetores latentes no espaço vetorial, com o objetivo de gerar transições graduais entre representações no espaço latente.

A função *lerp* recebe três parâmetros: *val*, um escalar entre 0 e 1 que define a proporção da interpolação; *low*, o vetor inicial; e *high*, o vetor final. O resultado é um vetor posicionado linearmente entre *low* e *high*, conforme o valor de *val*.

Já a função *slerp* também recebe *val*, *low* e *high*, mas considera a curvatura da esfera unitária em torno da origem. Ela calcula o ângulo entre os vetores normalizados e utiliza essa informação para realizar a interpolação sobre a superfície da esfera, garantindo uma transição suave. Nos casos em que os vetores são colineares (ângulo próximo de zero), a função recorre à interpolação linear como limite.

Figura 30 - Interpolação linear e esférica entre vetores latentes

```
[ ] def lerp(val, low, high):
    return (1.0 - val) * low + val * high

def slerp(val, low, high):
    omega = arccos(clip(dot(low/norm(low), high/norm(high)), -1, 1))
    so = sin(omega)
    if so == 0:
        # Regra de L'Hôpital/LEP
        return (1.0-val) * low + val * high
    return sin((1.0-val)*omega) / so * low + sin(val*omega) / so * high
```

A função *trancisao* Figura 31 tem como objetivo realizar uma transição progressiva entre uma imagem inicial, gerada a partir de um vetor latente, e uma imagem alvo fornecida. Para isso, o código utiliza um processo de otimização no qual o vetor latente carregado (*busca_vetor1_carregado*) é ajustado iterativamente por meio do otimizador Adam. A cada época (*epoch*), o vetor latente atualizado é passado para o gerador (*G*) que produz uma nova imagem, e essa imagem gerada é comparada com a imagem alvo usando a função de perda *MSELoss*, que mede a diferença entre as duas.

Durante o processo, as imagens geradas e os respectivos vetores latentes são armazenados em uma lista chamada de “resultados”. O processo de otimização é interrompido automaticamente quando não há melhora significativa na perda por um número consecutivo de épocas definido por *patience*, evitando *overfitting* ou execuções desnecessárias. Ao final, a função retorna todas as imagens geradas ao longo do processo de transição, permitindo visualizar como a imagem inicial vai se transformando gradualmente até se aproximar da imagem alvo.

Figura 31 - transição de um vetor latente inicial para uma imagem alvo

```
def trancisao(G, z_i, busca_vetor1_carregado, img_alvo1, maxEpochs= maxEpochs, patience=5):
    z = busca_vetor1_carregado.clone().detach().requires_grad_(True)
    z_optimizer1 = optim.Adam([z], lr=0.01)
    criterion1 = nn.MSELoss(reduction='sum')# metrica de avaliação na comparação entres as imagens gerada e alvo
    z_optimizer1.zero_grad()
    last_loss1 = 10000
    no_change1 = 0
    resultados = [] # Lista para armazenar dados
    for epoch in range(maxEpochs + 1):
        generated_image1 = G(z)
        loss = criterion1(generated_image1, img_alvo1)
        loss.backward()
        z_optimizer1.step()
        z_optimizer1.zero_grad()

        if epoch % 1 == 0:
            img = generated_image1.squeeze().detach().cpu().numpy()
            resultados.append({'id': epoch+1, 'imagem_gerada': img, 'vetor_latente': z.clone().detach()})
            plt.figure(figsize=(2, 2))
            plt.imshow(img, cmap='gray')
            plt.title(f'Epoch {epoch}')
            plt.axis('off')
            plt.show()

        if last_loss1 - loss.item() < 0:
            no_change1 += 1
        else:
            no_change1 = 0
        if no_change1 >= patience:
            break
        last_loss1 = loss.item()

    return resultados # Retorna todas as imagens e vetores latentes salvos
```

3.4.5 Treinamento da DCGAN

O treinamento da DCGAN é realizado pela função *train()*, que coordena a interação entre o gerador (*G*) e o discriminador (*D*). Em cada época, o modelo percorre *mini-batches*, pequenos subconjuntos do conjunto de dados de treinamento fornecidos pelo *trainloader*, utilizando imagens reais para atualizar os parâmetros do discriminador.

Optou-se por utilizar *batch_size = 1*, considerando o tamanho reduzido do conjunto de dados (540 imagens). Com essa configuração, cada mini-batch representa aproximadamente 0,185% do total de dados. Essa escolha permite atualizações mais frequentes dos parâmetros da rede, o que é vantajoso em cenários com poucos dados.

Primeiro, o discriminador (*D*) é treinado com imagens reais, rotuladas como verdadeiras, e em seguida com imagens sintéticas geradas pelo gerador (*G*) a partir de vetores aleatórios criados pela função *generateLatentPoints()*. Essas imagens sintéticas são rotuladas como falsas. As perdas para imagens reais, armazenadas na variável *d_loss_RS*, e para imagens sintéticas, na variável *d_loss_FS*, são calculadas pela função *BCELoss()* (entropia cruzada binária) e somadas para atualizar os pesos do discriminador.

$$L_{BCE} = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})] \quad (10)$$

Na equação 10, temos a entropia cruzada binária definida como, *y* é o rótulo verdadeiro (1 para imagens reais e 0 para falsas) e $\hat{y}$ é a saída do discriminador (probabilidade entre 0 e 1). A soma dessas perdas é usada para atualizar os pesos do discriminador (*D*) (Ian Goodfellow et al., 2017).

Depois, o gerador é treinado. Novos vetores aleatórios são gerados, transformados em imagens por *G* e avaliados por *D*, mas agora com rótulos "verdadeiros". Isso força *G* a criar imagens cada vez mais realistas para enganar *D*.

Durante o treinamento, as perdas de *G* e *D* são armazenadas e exibidas a cada época por meio da função *printLosses()*. Amostras de imagens geradas são mostradas com a função *printImages()* e, a cada quatro épocas, também armazenadas na lista *generatedImages* para análise futura.

Os modelos são inicializados com pesos aleatórios *initWeights()* e otimizados com o algoritmo Adam (taxa de aprendizado de 0.00001 e parâmetros *betas* iguais a 0.5 e 0.999), configurado para promover estabilidade. Uma das técnicas utilizadas para melhorar a estabilidade foi a suavização de rótulos (*label smoothing*). Essa técnica consiste em substituir o valor fixo dos rótulos reais (geralmente 1.0) por valores aleatórios dentro de um intervalo, como entre 0.8 e 1.0, durante o treinamento do discriminador. Da mesma forma, os rótulos falsos (gerados pelo gerador) podem assumir valores entre 0.0 e 0.2.

Ao final, a função retorna as perdas médias de G e D ao longo das épocas, além das imagens sintéticas geradas. Esse ciclo adversarial resulta na criação de um gerador capaz de produzir imagens realistas, enquanto o discriminador se torna um avaliador cada vez mais refinado. Para ajudar na compreensão do modelo, foi selecionado os principais parâmetros utilizados na DCGAN, como mostrado na Tabela 1.

Tabela 1 - Hiperparâmetros usados na DCGAN

Parâmetro	Valor	Descrição
z_i	100	Dimensão do vetor latente de entrada do gerador
d_n	64	Nº de filtros base da rede discriminadora
g_n	64	Nº de filtros base da rede geradora
$batch_size$	1	Nº de imagens por lote no DataLoader
$maxEpochs$	50	Nº máximo de épocas de treinamento
$lr(D)$	0.00001	Taxa de aprendizado do discriminador
$lr(G)$	0.00001	Taxa de aprendizado do gerador
$betas(D)$	(0.5, 0.999)	Parâmetros β do Adam para o discriminador
$betas(G)$	(0.5, 0.999)	Parâmetros β do Adam para o gerador
$image_size$	224×224	Tamanho das imagens usadas no treino
$image_channels$	1	Nº de canais (1 = escala de cinza)
$normalization$	(-1, 1)	Intervalo de normalização das imagens

3.4.6 Geração Direcionada de Imagem

Após o treinamento, a DCGAN está pronta para gerar imagens sintéticas do sensor SPR correspondentes às classes 1.330, 1.335, 1.340, 1.345 e 1.350 para as quais foi treinada. O processo de geração direcionada de imagem dessas, começa com a função *findLatentVector*, que cria um vetor latente aleatório de 100 dimensões, que serve como ponto de partida. Esse vetor é então ajustado gradualmente por meio de um processo de otimização.

Em cada etapa, a imagem gerada pelo modelo é comparada com uma imagem-alvo, e a diferença entre elas é calculada usando a função de erro quadrático médio (em inglês, *Mean Squared Error Loss – MSELoss*), que mede o erro médio entre os pixels das duas imagens. Essa métrica é essencial para orientar os ajustes no vetor latente (Haykin, 1996).

Com base nesse erro, o vetor latente é atualizado com otimizador Adam, que ajusta os parâmetros da rede neural com o objetivo de minimizar a diferença entre a imagem gerada e a imagem-alvo (Kingma e Ba, 2017). O processo de otimização é interrompido automaticamente quando a perda deixa de melhorar após algumas épocas consecutivas, conforme definido pelo critério de paciência.

Além disso, é possível gerar imagens correspondentes a classes que não foram observadas durante o treinamento, como 1.331, 1.332, 1.333 e 1.334. Isso é feito pela função de *transicao()*. Esse processo permite que uma imagem inicial evolua gradualmente, perdendo suas características originais enquanto adquire as da imagem-alvo. O método opera em duas etapas principais: primeiro, extrai-se o vetor latente da imagem de partida que se deseja transformar; em seguida, realiza-se uma transição suave em direção ao vetor latente da imagem alvo. Essa abordagem permite criar uma sequência de transição onde, a cada passo, a imagem vai se tornando menos parecida com a original e mais semelhante ao alvo, essa função é evidenciada na Figura 31.

Um exemplo prático desta técnica é a geração de imagens com índices de refração intermediários entre duas classes conhecidas. Por exemplo, entre uma imagem A com índice 1,330 e uma imagem B com índice 1,335, é possível gerar uma nova imagem correspondente ao índice 1,331, representando uma transição contínua e realista entre os dois casos.

Esse processo é viabilizado pelo deslocamento gradual da mancha escura característica entre diferentes posições. Em determinado ponto da transição, a imagem gerada deixa de corresponder exatamente às classes 1.330 ou 1.335, passando a apresentar características visuais intermediárias. Dessa forma, obtém-se uma imagem que representa uma nova classe intermediária entre as duas originais, tanto em termos de índice de refração quanto de morfologia visual.

Para avaliar a capacidade de generalização e a continuidade do espaço latente aprendido pelo modelo gerador da DCGAN, utilizamos a função de *interpolateLatentPoints* entre vetores latentes. Essa técnica consiste em combinar gradualmente as características de dois vetores distintos ($p1$ e $p2$), resultando em imagens intermediárias nas quais os atributos visuais de ambas as entradas são suavemente mesclados (Radford; Metz; Chintala, 2016); (Karras, Laine e Aila, 2019). O objetivo principal é gerar uma sequência contínua de imagens sintéticas que revele como o modelo mapeia regiões do espaço latente para o domínio das imagens, permitindo analisar tanto a coerência semântica quanto a suavidade das transições.

Inicialmente, selecionamos pares de imagens e extraímos seus vetores latentes correspondentes ($p1$ e $p2$). A interpolação é então realizada através de duas abordagens complementares: (1) interpolação linear, que calcula combinações ponderadas entre os vetores ao longo de uma trajetória retilínea no espaço latente; e (2) interpolação esférica (slerp), que preserva a distância angular entre os vetores, sendo particularmente adequada para espaços latentes normalizados, uma propriedade comum em modelos generativos profundos (White, 2016). A escolha entre esses métodos é controlada por um parâmetro específico, permitindo comparar seus efeitos na qualidade e semântica das imagens geradas.

Cada vetor interpolado é subsequentemente processado pelo gerador da DCGAN, produzindo imagens que revelam transições graduais entre os atributos das imagens de origem. Essa metodologia oferece três vantagens principais: (a) análise qualitativa da organização do espaço latente, (b) verificação da capacidade de generalização do modelo, e (c) identificação de possíveis descontinuidades ou regiões degeneradas no espaço de características. Por esses motivos, a interpolação latente tornou-se um protocolo padrão para avaliação de modelos generativos, conforme demonstrado em trabalhos (Radford et al., 2016).

A função *gerar_imagem_soma_latentes* tem como objetivo investigar como operações algébricas entre vetores latentes influenciam a geração de imagens em modelos generativos, com ênfase na análise das propriedades geométricas e semânticas do espaço latente. Para isso, é conduzido um experimento a partir de dois vetores latentes extraídos de imagens do *dataset* gerada por este modelo. Em seguida, é realizada uma subtração vetorial ($v1 - v2$) entre esses vetores, depois é aplicado um soma vetorial ($v1 + v2$). Essas operações são apresentadas com mais detalhes na Seção 4.2.1.

O vetor resultante dessas operações é então passado pelo gerador do modelo, produzindo uma nova imagem sintética que, em tese, incorpora as características diferenciais entre as duas condições originais. Esse procedimento permite avaliar se o espaço latente aprendido pelo modelo é capaz de capturar relações semânticas entre os atributos das imagens de forma estruturada.

3.5 Avaliação do Modelo Escolhido

Neste trabalho, foram desenvolvidos dois conjuntos de dados compostos por imagens geradas por uma DCGAN. O primeiro conjunto, denominado *dataset_60*, contém 60 imagens em tons de cinza, com resolução de 224×224 pixels, distribuídas igualmente entre 5 classes, ou seja, 12 imagens por classe. O segundo conjunto, chamado *dataset_600*, é composto por 600 imagens com as mesmas características (tons de cinza e resolução de 224×224 pixels), também distribuídas igualmente entre 5 classes, com 120 imagens por classe.

O *dataset_600* será utilizado como entrada em uma rede neural, com o objetivo de demonstrar a aplicabilidade prática das imagens geradas. Já o *dataset_60* será empregado na visualização dos resultados, permitindo a avaliação da eficiência do modelo por meio de análises quantitativas, qualitativas e específicas.

3.5.1 Avaliação Quantitativa

Para avaliar a similaridade entre as distribuições dos dados reais e gerados, foi utilizada a métrica (em inglês, *Maximum Mean Discrepancy* - MMD), uma ferramenta estatística eficaz para mensurar a semelhança entre distribuições de probabilidade com base em amostras. Introduzida por (Gretton *et al.*, 2012), a MMD se baseia na projeção dos dados em um espaço de Hilbert com núcleo reproduzível (em inglês, *Reproducing Kernel Hilbert*

Space - RKHS), no qual a distância entre duas distribuições é estimada pelas médias empíricas das amostras projetadas.

Nos experimentos realizados, a MMD foi aplicada para mensurar a similaridade entre as distribuições das características extraídas do conjunto de dados gerado (*dataset_60*) e do conjunto de teste real (*testloader*). Para a extração dessas características, utilizou-se a rede InceptionV3, pré-treinada no ImageNet, conforme práticas amplamente adotadas na literatura. As imagens foram convertidas para o formato RGB, redimensionadas para 224×224 pixels, e processadas pela *InceptionV3* com a camada final removida, a fim de obter os vetores de características.

O cálculo da MMD foi realizado utilizando um *kernel Gaussiano* com $\sigma = 1.0$. Foram analisadas 60 imagens reais e 60 imagens geradas, cada grupo tem 5 classes, cada classe 12 imagens, no intervalo das classes 1.33 a 1.35.

O histograma é uma das ferramentas estatísticas mais tradicionais para análise de dados visuais, introduzido por Karl Pearson no final do século XIX como método para representar graficamente a distribuição de frequências de variáveis contínuas (Pearson, 1894). No contexto do processamento de imagens, o histograma tornou-se amplamente utilizado por sua capacidade de descrever a distribuição das intensidades de pixels em uma imagem, fornecendo informações essenciais sobre contraste, brilho e outras características visuais (Gonzalez e Woods, 2018).

Nos experimentos conduzidos, foi utilizada uma base de dados composta pelo conjunto de dados gerado (*dataset_60*) e pelo conjunto de dados de teste (*testloader*). A análise foi realizada por meio da comparação entre os histogramas das distribuições de características extraídas de ambos os conjuntos. O objetivo dessa análise é verificar se as imagens geradas apresentam uma distribuição estatística de intensidades de pixel semelhante à das imagens reais, utilizando como ferramenta principal a análise de histogramas médios.

Para isso, foi implementado um código em *Python* utilizando as bibliotecas *PyTorch*, *NumPy* e *Matplotlib*. O código realiza a leitura de um arquivo .pt contendo as imagens reais e geradas, normaliza os valores de intensidade de *pixel* para o intervalo $[0, 255]$, e calcula os histogramas individuais de cada imagem com 256 *bins*. Em seguida, os histogramas das

imagens reais e geradas são somados separadamente e divididos pelo número total de imagens (60), resultando nos histogramas médios para cada grupo. Por fim, os dois histogramas médios são plotados em um mesmo gráfico, permitindo uma comparação visual direta entre as distribuições.

Foi utilizada uma base composta pelo conjunto de dados gerado (*dataset_60*) e pelo conjunto de dados de teste (*testloader*), com o objetivo de projetar as distribuições em espaços de baixa dimensão por meio das técnicas de Incorporação Estocástica de Vizinhos com Distribuição-t (em inglês, *t-Distributed Stochastic Neighbor Embedding* - t-SNE) (Van der Maaten e Hinton, 2008). Essas técnicas são fundamentais para facilitar a visualização e a interpretação dos dados, permitindo avaliar a capacidade do modelo em reproduzir a distribuição estatística e as características visuais dos dados reais em baixa dimensão.

Neste trabalho, utilizamos o t-SNE como ferramenta de análise exploratória, com o objetivo de identificar padrões e agrupamentos nos dados. Segundo os autores Kobak e Berens (2019), embora valores padrão de perplexidade (30–50) funcionem bem para conjuntos grandes, conjuntos muito pequenos exigem valores menores para evitar a inclusão de vizinhos artificiais que não refletem a estrutura real dos dados.

Para esta análise, foram testados diversos valores de perplexidade (8 a 17) a fim de avaliar o comportamento dos dados na projeção bidimensional. O valor 12 foi selecionado com base em critérios empíricos de qualidade visual, pois proporcionou um equilíbrio satisfatório entre a separação dos grupos originais e a continuidade das trajetórias interpoladas. Observou-se que, ao se afastar desse valor, seja aumentando ou diminuindo a perplexidade, a estrutura do conjunto apresentava deformações, comprometendo a organização e a interpretação da projeção.

Complementando essa análise, implementou-se um procedimento automático para auxiliar na seleção da perplexidade, baseado no cálculo da divergência de *Kullback–Leibler* obtida pelo t-SNE para diferentes valores do parâmetro (Kullback e Leibler, 1951). O método indicou os valores 12 e 13 de perplexidade como ideais, em concordância com a avaliação qualitativa inicial, reforçando a robustez da escolha e reduzindo a subjetividade do processo.

As técnicas foram aplicadas em diferentes contextos, como geração de imagens, interpolação e transição entre classes. Os experimentos foram realizados com um conjunto de 60 imagens geradas, com resolução de 224×224 *pixels*, em tons de cinza, distribuídas igualmente entre 5 classes, totalizando 12 imagens por classe. A alta dimensionalidade tanto das imagens (em pixels) quanto o vetor latente deles torna inviável a visualização direta, o que reforça a necessidade da redução dimensional para uma representação visual em duas dimensões.

Para mapear a transição entre as classes 1.330 e 1.335 no espaço latente, utilizou-se a distância euclidiana como métrica fundamental, uma prática comum na análise de trajetórias em espaços latentes gerados por redes generativas adversariais, conforme discutido por White (2016). As coordenadas X e Y foram extraídas de um conjunto de dados gerados pela DCGAN, por meio do t-SNE, com perplexidade 12 e semente aleatória fixada em 0. A partir dessas informações, foram definidos dois pontos de referência: $P1$ (classe 1.330) e $P2$ (classe 1.335). A distância total entre os dois extremos serviu como base para mapear os vetores latentes intermediários ao longo da trajetória entre as classes.

Cada vetor teve sua classe estimada proporcionalmente à sua distância relativa à P_1 , resultando em uma gradação contínua de valores entre as classes 1.330 e 1.335. Essa abordagem revelou a estrutura interna do espaço latente, indicando uma transição suave e sem rupturas.

3.5.2 Avaliação do Treinamento

Durante o treinamento da rede DCGAN, foi realizado o monitoramento das perdas médias do discriminador (d_loss_avg) e do gerador (g_loss_avg) ao longo das épocas. A estabilização simultânea dessas duas curvas de perda sugere que ambos os modelos convergem para um ponto de equilíbrio, no qual o gerador passou a produzir amostras de alta qualidade e o discriminador manteve uma capacidade discriminatória limitada, porém não nula (Goodfellow et al., 2014).

Essa dinâmica confirma a eficácia do processo de treinamento e a adequação dos hiperparâmetros escolhidos, além de evidenciar a ausência de problemas comuns, como *mode collapse* ou divergência das perdas (Radford; Metz; Chintala, 2016).

3.5.3 Avaliação Física

A avaliação física da qualidade das imagens geradas pelo modelo foi conduzida com base em quatro métricas principais: full width at half maximum (FWHM), largura, energia e assimetria. Essas métricas foram escolhidas por sua relevância na caracterização óptica de imagens obtidas por sensores SPR, permitindo uma análise quantitativa das propriedades essenciais das curvas de intensidade geradas.

A métrica FWHM (largura total na metade da altura) é amplamente utilizada na literatura por ser um indicador direto da resolução e da sensibilidade das curvas SPR. Ela permite mensurar a nitidez da resposta óptica e é fundamental na avaliação da performance do sensor em diferentes configurações.

A largura da curva, entendida como a extensão da resposta de intensidade ao longo do eixo de interesse, oferece uma medida complementar ao FWHM, ajudando a caracterizar a distribuição espacial da informação óptica. Já a Energia, definida como a soma dos quadrados das intensidades dos pixels ao longo do perfil da curva, fornece uma medida global da intensidade acumulada, refletindo o conteúdo energético da imagem.

Por fim, a assimetria da curva em torno do ponto de mínimo, frequentemente associada ao ponto de ressonância do sensor SPR, permite avaliar desvios na forma da curva, o que pode indicar variações estruturais ou comportamentais no sinal gerado. Essa métrica é obtida a partir da comparação entre os pontos de cruzamento à meia altura localizados à esquerda e à direita do mínimo da curva.

A utilização combinada dessas métricas proporciona uma análise física robusta, permitindo comparar a qualidade das imagens geradas com os padrões esperados em sensores SPR e avaliar a fidelidade do modelo quanto à reprodução de propriedades ópticas essenciais, conforme Oliveira (2016).

Outra avaliação foi conduzida por meio de operações aritméticas no espaço latente, com foco em somas e subtrações de vetores latentes extraídos de imagens reais pertencentes a diferentes classes. Inicialmente, foram selecionadas imagens de distintas classes (por exemplo, 1.33, 1.335 e 1.34) e, a partir dessas, realizou-se a extração dos vetores latentes por meio do modelo DCGAN previamente treinado. Esses vetores, que representam a codificação

semântica das imagens no espaço latente, foram então combinados por meio de operações de soma e subtração.

As somas tem como objetivo verificar a possibilidade de geração de imagens , morfologicamente plausíveis, refletindo uma fusão gradual entre as classes originais intermediárias. Já as subtrações buscaram explorar a distinção visual entre os padrões, revelando características exclusivas de cada classe.

3.5.4 Avaliação Visual

A avaliação visual direta das imagens geradas pelo modelo teve como objetivo principal verificar sua fidelidade em relação às imagens simuladas do sensor SPR. Para cada uma das cinco classes analisadas (1.33, 1.335, 1.34, 1.345 e 1.35), foram selecionadas amostras reais e suas respectivas versões sintéticas produzidas pela DCGAN, organizadas em pares para permitir uma comparação visual detalhada. Essa abordagem possibilita a identificação tanto de similaridades estruturais quanto de eventuais distorções nas imagens geradas.

Três aspectos fundamentais guiaram esta análise qualitativa. Primeiramente, avaliou-se a fidelidade morfológica das imagens sintéticas, com especial atenção à forma, posição e contraste da mancha escura característica das curvas SPR, que representa o ponto de ressonância do sensor. Em segundo lugar, examinou-se a diversidade intra-classe, garantindo que as imagens geradas apresentassem variações naturais sem repetições excessivas ou falhas perceptíveis que pudessem comprometer sua qualidade e utilidade prática.

4. RESULTADOS

Nesta seção, são apresentados os principais resultados obtidos no desenvolvimento do trabalho, abrangendo a avaliação do modelo proposto e a geração de novas classes. As análises aqui realizadas visam demonstrar a eficiência do modelo tanto do ponto de vista qualitativo quanto quantitativo, explorando aspectos visuais, estatísticos e físicos das imagens geradas.

4.1 Avaliação do Modelo

A avaliação do modelo foi realizada em múltiplas etapas, garantindo uma análise abrangente de seu comportamento. Inicialmente, foi examinado o processo de treinamento, verificando a convergência e a estabilidade do aprendizado Seção 4.1.1. Em seguida, uma avaliação visual das imagens geradas ou processadas pelo modelo permitiu identificar possíveis artefatos ou inconsistências Seção 4.1.2.

Para uma análise mais aprofundada, foi utilizado técnicas de estatísticas como histogramas e redução de dimensionalidade (t-SNE) para revelar a distribuição e a estrutura dos dados Seção 4.1.3. Adicionalmente, o desempenho foi avaliado separadamente por classe, destacando eventuais desequilíbrios ou dificuldades específicas Seção 4.1.4. Por fim, casos individuais foram analisados para entender melhor os erros e acertos do modelo Seção 4.1.5.

4.1.1 Avaliação do Treinamento

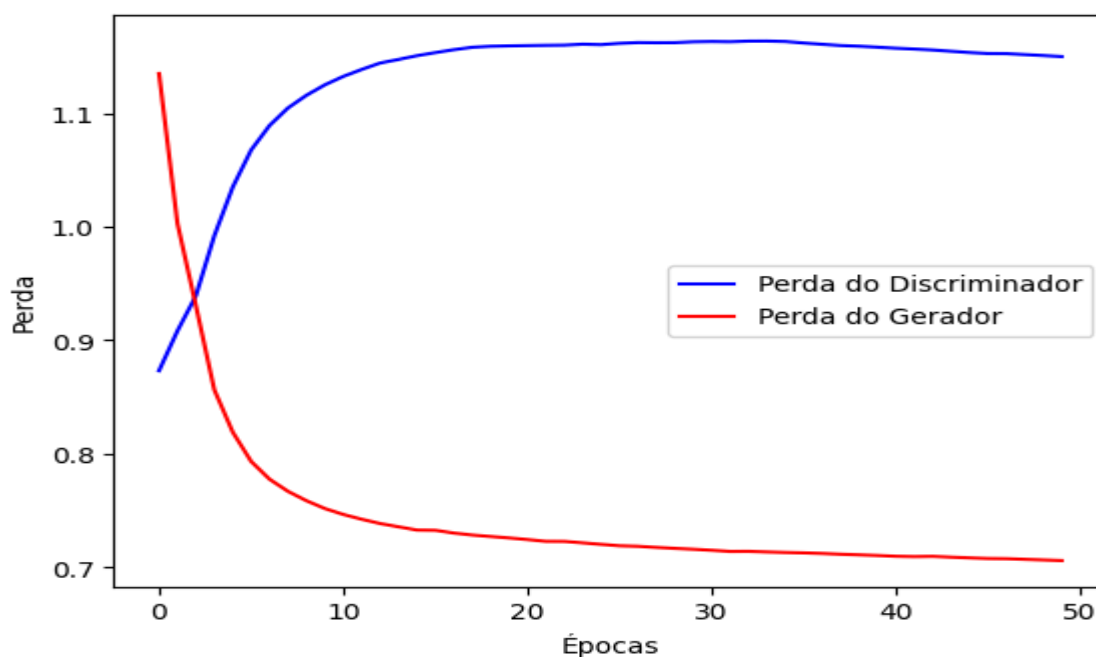
De início, analisou-se o processo de treinamento da DCGAN por meio das curvas de perda do gerador (linha vermelha) e do discriminador (linha azul), conforme ilustrado na Figura 32. Nas primeiras épocas, observou-se que a perda do gerador começou elevada, refletindo sua dificuldade inicial em produzir imagens realistas. À medida que o treinamento progrediu, essa perda foi diminuindo e se estabilizou ao longo das 50 épocas, indicando que o gerador havia aprendido a criar imagens mais convincentes.

Paralelamente, a perda do discriminador apresentou um comportamento característico: após um aumento inicial nas primeiras épocas, estabilizou-se ao longo do treinamento. Esse padrão é típico em GANs bem-sucedidas, demonstrando que o discriminador estava encontrando maior dificuldade em distinguir entre imagens reais e falsas.

Embora o gráfico apresente uma instabilidade em torno da 30ª época, as imagens geradas nesse ponto ainda exibiam borrões e não apresentavam qualidade satisfatória, conforme verificado nos testes detalhados na Seção 4.1.5, o que indicava a necessidade de mais épocas de treinamento. Foram realizados testes até a 55ª época; no entanto, após a 50ª, as imagens não mostraram melhorias significativas, apenas se mantendo estáveis. Dessa forma, o treinamento do modelo foi finalizado com 50 épocas.

Embora a função do BCE (Binary Cross-Entropy), utilizada para calcular a perda do gerador e do discriminador, gere valores entre 0 e 1 para cada amostra individual, o valor médio apresentado nos gráficos pode ultrapassar esse intervalo. Isso ocorre porque os gráficos representam a média da perda calculada sobre todos os exemplos de um *batch*, que é baseada nas probabilidades previstas pelo modelo e não nos valores individuais das amostras. Esse comportamento é observado no trabalho de Saad et al. (2024), ao investigar os valores de perda do gerador e do discriminador na DCGAN.

Figura 32 - Evolução das perdas do discriminador e do gerador



Além disso, o comportamento suave e estável das curvas de perda ao longo do treinamento indica que não houve ocorrência de *mode collapse*, isso é comum em

treinamentos instáveis de GANs, pois o gerador aprende a produzir apenas um número limitado de amostras, comprometendo a diversidade das imagens geradas.

Esses resultados iniciais sugerem que o modelo atingiu um equilíbrio estável entre o gerador e o discriminador, operando de forma eficaz dentro dos parâmetros definidos. Nas próximas seções, serão apresentadas análises visuais e quantitativas que reforçam e comprovam o desempenho positivo alcançado pela DCGAN na geração de imagens sintéticas.

4.1.2 Avaliação Visual das Imagens

Para cada classe, foram geradas 12 imagens, totalizando 60 amostras. No entanto, visando melhor organização visual e clareza na apresentação dos resultados, optou-se por exibir apenas 5 imagens por classe.

As Figuras 33 a 37 apresentam uma comparação visual entre amostras reais e imagens geradas pela DCGAN, abrangendo as classes analisadas: 1,33, 1,335, 1,34, 1,345 e 1,35. Cada figura é composta por duas linhas de imagens: a primeira linha exibe as imagens reais no domínio espacial e a segunda linha mostra as imagens geradas no mesmo domínio. As análises a seguir referem-se às características gerais observadas em todas essas figuras, sem distinção individual por classe.

Na comparação entre as imagens reais (primeira linha) e as imagens geradas (segunda linha), observa-se que o modelo conseguiu replicar com razoável fidelidade as características morfológicas gerais presentes nas imagens reais, como a forma e a posição das regiões de maior densidade. No entanto, nota-se uma diferença na distribuição da intensidade luminosa: as imagens reais apresentam variações naturais de brilho, com áreas claramente mais escuras ou mais claras, enquanto as imagens geradas tendem a exibir uma tonalidade mais homogênea e suavizada, indicando uma limitação do modelo em capturar a complexidade da textura original.

Figura 33 - Comparação entre imagens reais e geradas da classe 1.33

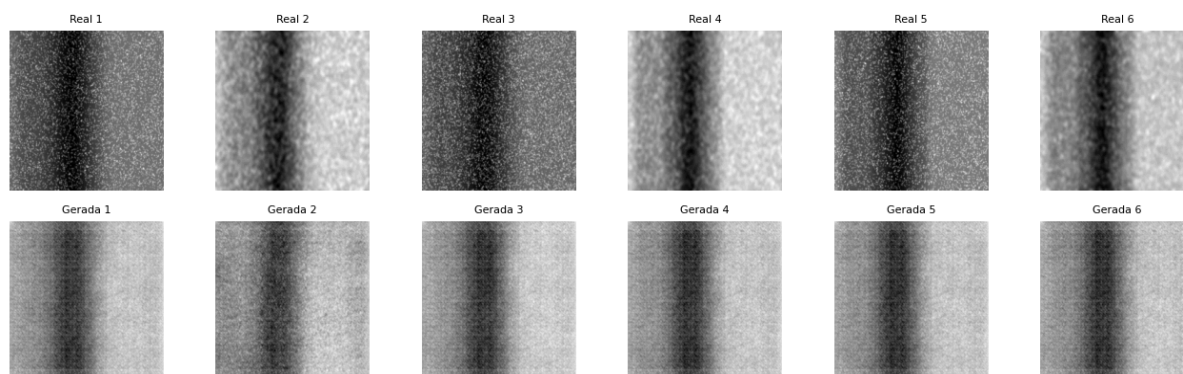


Figura 34 – Comparação entre imagens reais e geradas da classe 1.335

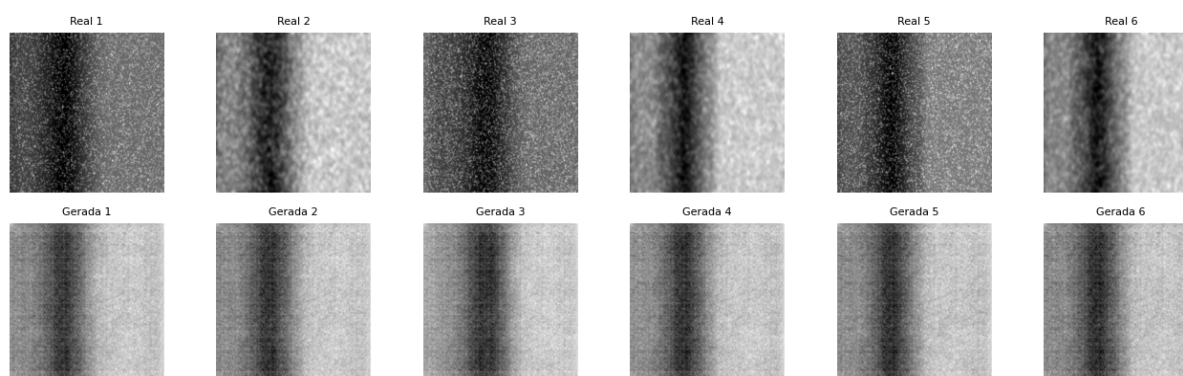


Figura 35 – Comparação entre imagens reais e geradas da classe 1.34

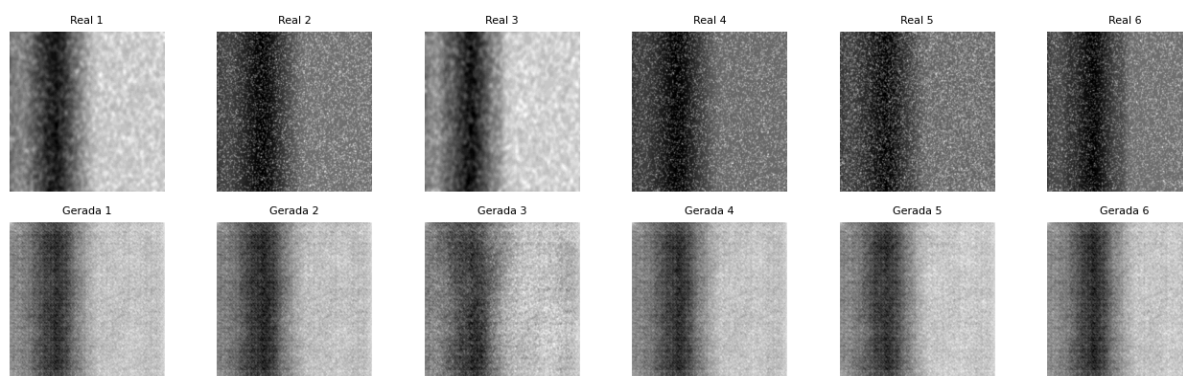


Figura 36 – Comparação entre imagens reais e geradas da classe 1.345

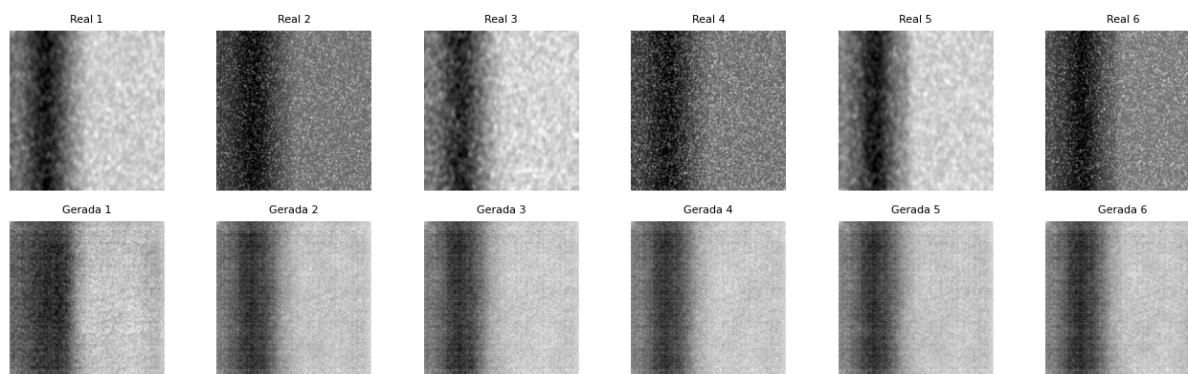
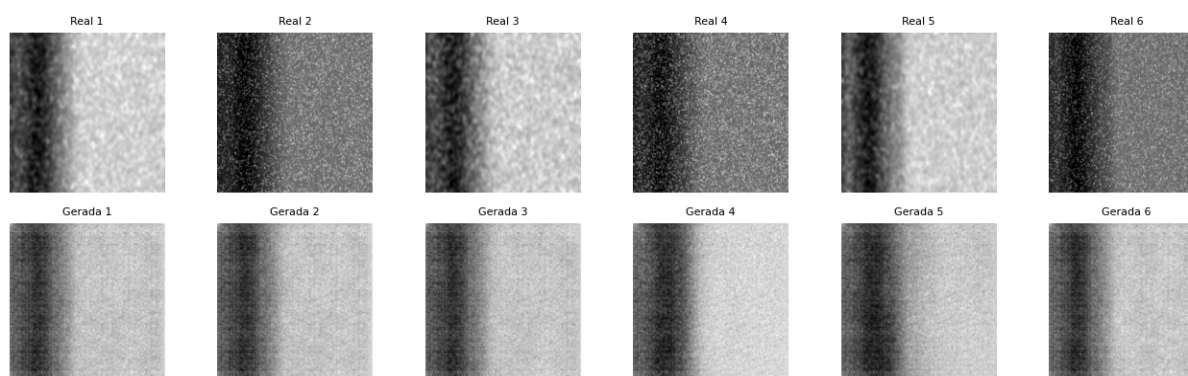


Figura 37 – Comparação entre imagens reais e geradas da classe 1.35



Em síntese, a avaliação visual comparativa entre as imagens reais e as imagens geradas pela DCGAN revelou que o modelo apresenta desempenho satisfatório na replicação das formas e estruturas morfológicas predominantes nas classes analisadas. As imagens geradas mantêm a coerência na forma geral e na posição das regiões de maior densidade, evidenciando a capacidade do modelo em capturar aspectos estruturais relevantes do conjunto de dados. Contudo, a análise também evidenciou limitações: no domínio espacial, as imagens geradas apresentaram tonalidade mais uniforme, com menor variação de intensidade luminosa em comparação às imagens reais, o que indica uma perda na complexidade visual e na riqueza de detalhes.

Esses resultados contribuem para uma melhor compreensão das capacidades e limitações da arquitetura DCGAN no contexto proposto, ressaltando a importância da avaliação visual para aferir a fidelidade das imagens sintéticas geradas por redes generativas.

4.1.3 Avaliação Geral dos Dados com MMD, Histogramas e o t-SNE

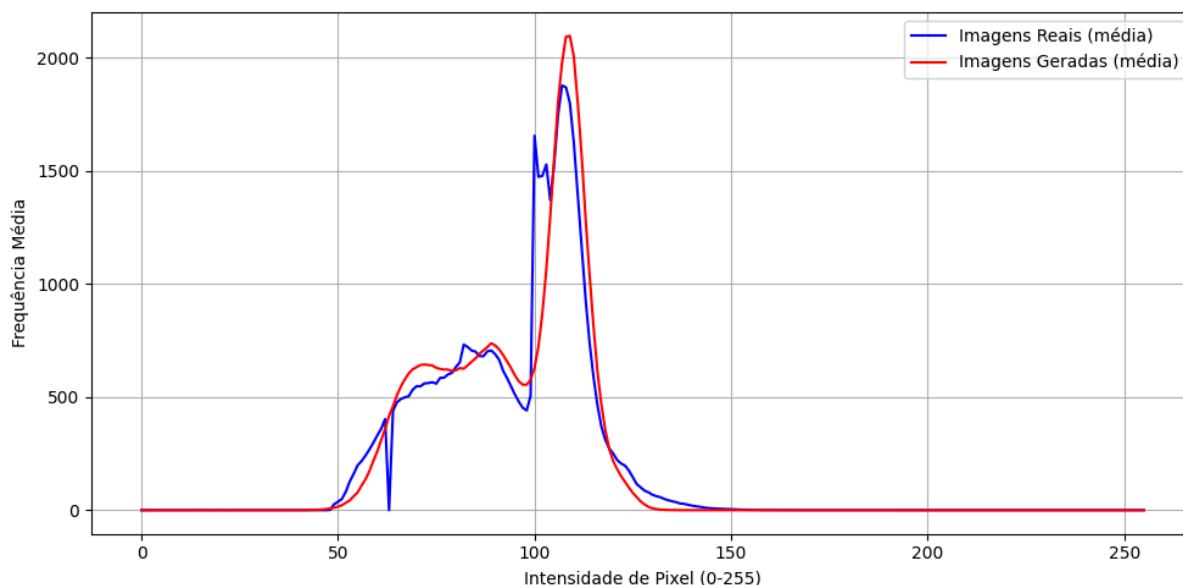
Nesta etapa, a avaliação das imagens geradas pela DCGAN é aprofundada por meio de três abordagens complementares: a métrica MMD, análise de histogramas e redução de dimensionalidade via t-SNE. Cada técnica contribui com uma perspectiva distinta sobre a qualidade estatística e visual das imagens sintéticas geradas.

O valor global do MMD com (todas as classes, reais e sintética) foi de 0.0360, o que representa uma distância pequena entre as distribuições dos dados reais e sintéticos. Como a MMD tende a zero quando as distribuições são semelhantes, esse resultado sugere que a DCGAN foi eficaz na geração de imagens com propriedades estatísticas próximas às amostras reais.

Embora a MMD não possua um limiar teórico fixo, estudos empíricos mostram que valores abaixo de 0.2 são frequentemente associados a boa similaridade entre distribuições. Em particular, (Xu *et al.*, 2018) reportaram valores de MMD variando de 0.194 a 0.270 ao avaliarem diferentes modelos GAN (DCGAN, WGAN, WGAN-GP e LSGAN), mesmo em experimentos com redes convolucionais treinadas em grandes bases de dados. Dessa forma, os valores observados neste trabalho estão abaixo da média esperada na literatura, reforçando a qualidade estatística das imagens geradas pela DCGAN.

A segunda abordagem utilizada, foi o histograma de intensidade para analisar a distribuição dos valores de pixel nas imagens, fornecendo uma representação estatística da variação de brilho e contraste. Para cada imagem, tanto real quanto gerada foi calculado um histograma com 256 bins, representando os níveis de intensidade de 0 (preto) a 255 (branco).

Figura 38 - Comparação de histogramas médios 60 imagens reais e 60 geradas



A curva azul representa as imagens reais, enquanto a curva vermelha corresponde às imagens geradas. Observa-se uma forte sobreposição entre as duas curvas, o que indica que o modelo gerador foi capaz de reproduzir de maneira satisfatória a distribuição global de intensidades de pixels presentes nas imagens reais. O eixo X do gráfico representa os níveis de intensidade dos pixels, variando de 0 (preto) a 255 (branco), enquanto o eixo Y indica a frequência média com que cada valor de intensidade aparece nas imagens analisadas.

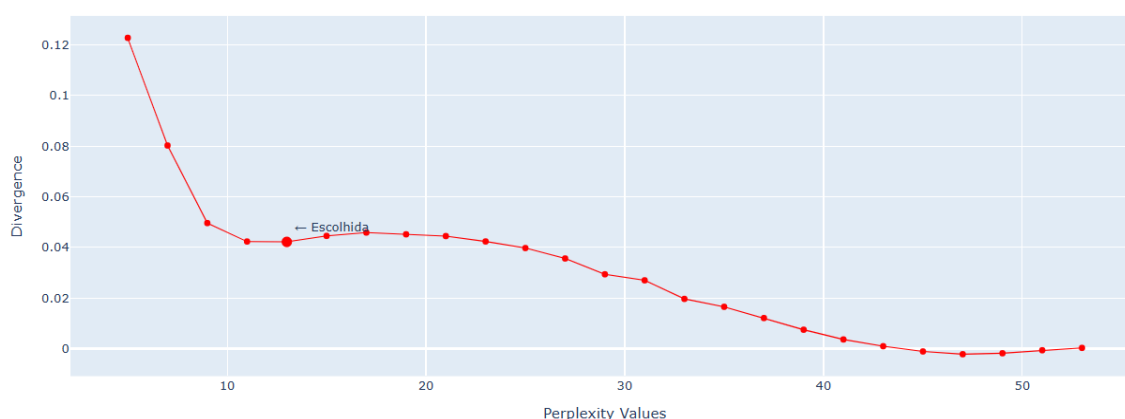
A similaridade entre os histogramas médios sugere que o modelo aprendeu padrões visuais importantes das imagens reais, especialmente no que se refere à distribuição tonal e variação de brilho. Dessa forma, a análise por histogramas reforça a evidência de que as imagens geradas mantêm um bom nível de realismo e coerência estatística.

E o último, é o algoritmo t-SNE, aplicado tanto às imagens reais quanto às imagens geradas, com o objetivo de reduzir a dimensionalidade de ambas as bases de dados e projetá-las em um espaço bidimensional. Essa redução dimensional permitiu uma visualização mais intuitiva da organização dos dados e das relações entre as classes, fornecendo subsídios para uma análise qualitativa da capacidade do modelo em gerar imagens coerentes com os dados reais.

Na Figura 39, usando a divergência de Kullback–Leibler, observa-se que a divergência diminui rapidamente nos primeiros valores e começa a se estabilizar a partir de uma perplexidade 12, indicando um ponto de equilíbrio entre fidelidade estrutural e redução de distorções.

Os testes realizados indicaram que as perplexidades 12 foram as que melhor preservaram a separação entre grupos e a continuidade das trajetórias interpoladas. O procedimento automático, baseado na análise da divergência, confirmou essa escolha. O marcador “Escolhida” no gráfico destaca a perplexidade adotada, evidenciando a convergência entre a avaliação qualitativa (visual) e o critério quantitativo, reforçando a robustez da decisão tomada.

Figura 39 – Perplexidade vs Divergência KL

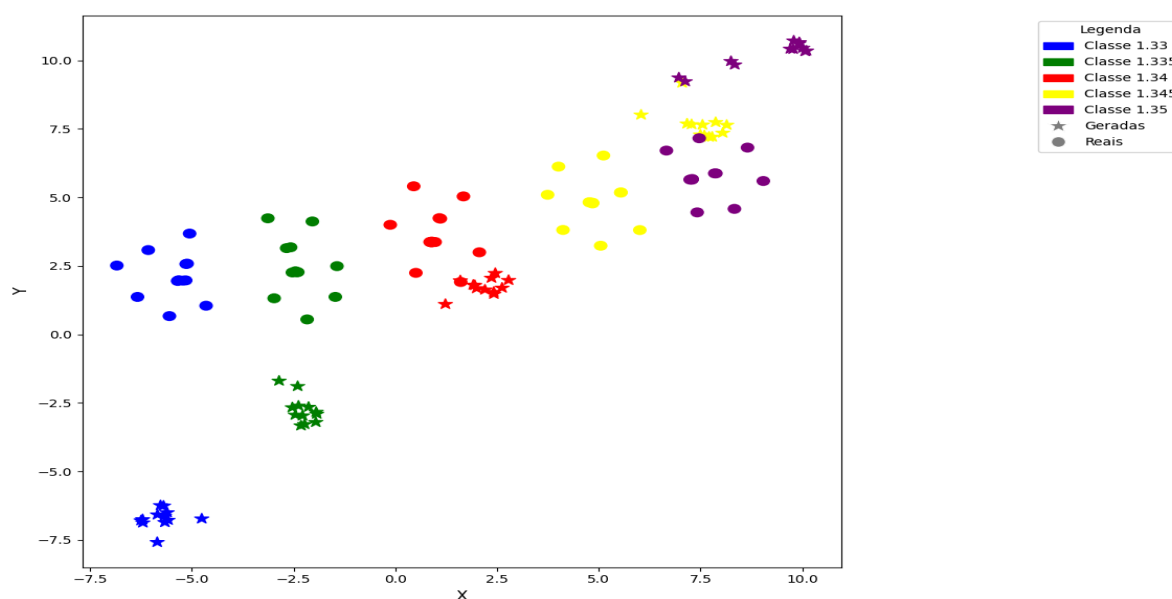


A configuração utilizada do t-SNE incluiu uma perplexidade igual a 12 e semente aleatória igual a 0. A perplexidade controla a quantidade de vizinhos considerados durante a projeção e o valor 12 foi escolhido por oferecer um bom equilíbrio entre a preservação das estruturas locais (agrupamentos próximos) e globais (distribuição geral dos dados). Já a semente aleatória foi definida para garantir reprodutibilidade nos resultados, assegurando que a projeção permaneça consistente em execuções repetidas, desde que os dados sejam apresentados na mesma ordem.

O gráfico da Figura 40 faz uma comparação das posições das imagens geradas vs reais por classe reduzidas para duas dimensões usando o t-SN, essa técnica foi aplicada nos pixels

das imagens. Cada ponto no gráfico representa uma imagem, e as cinco classes (1,33, 1,335, 1,34, 1,345 e 1,35) estão representadas por cores e marcadores distintos, separando imagens reais e geradas. Foram utilizadas 12 imagens por classe, totalizando 60 pontos para cada tipo de imagem. A visualização permite uma representação clara da distribuição das imagens no espaço bidimensional, facilitando a comparação entre os agrupamentos de amostras reais (representadas por círculos) e sintéticas (representadas por estrelas).

Figura 40 - Exibição visual dos componentes do t-SNE para imagens reais e geradas



A análise do gráfico permite observar que as imagens geradas seguem uma distribuição semelhante à dos dados reais. Essa comparação visual dos agrupamentos fornece uma avaliação qualitativa adicional da capacidade do modelo em aprender e representar corretamente a distribuição das diferentes classes. Agrupamentos semelhantes nos dois conjuntos de dados indicam que o modelo gerador foi capaz de captar as características morfológicas que distinguem as classes na base original.

A proximidade entre os agrupamentos de amostras reais e sintéticas, principalmente em relação ao eixo X , evidencia o grau de semelhança morfológica entre elas, demonstrando a eficácia do modelo em reproduzir as características essenciais dos dados reais. Vale destacar também que as imagens do *dataset* (reais) aparecem mais dispersas, uma vez que tais imagens foram geradas por diferentes técnicas, as quais podem apresentar características diferentes,

enquanto que as imagens geradas pela GAN são mais agrupadas por seguirem um padrão mais homogêneo, o que sugere uma tentativa de uniformização da diversidade apresentada no treinamento da GAN.

4.1.4 Avaliação por Classe

Imagens digitais podem ser representadas como matrizes bidimensionais de intensidades de pixels (Gonzalez; Woods, 2018). Uma técnica simples e eficaz para analisar a estrutura de uma imagem é calcular a soma dos valores de intensidade em cada coluna, resultando em uma curva unidimensional que representa a variação da intensidade ao longo do eixo horizontal. Essa curva funciona como uma espécie de assinatura visual da imagem.

Para avaliar a diversidade das classes de imagens geradas pela DCGAN, foi realizada uma análise baseada nas curvas de intensidade da luz refletida ao longo das colunas de pixels. Para cada classe (1,31, 1,32, 1,33, 1,34 e 1,35), foram selecionadas 12 imagens sintéticas aleatoriamente. A partir dessas imagens, foram extraídas as curvas correspondentes, normalizadas e comparadas graficamente.

As Figuras 41 a 45 apresentam as curvas extraídas de imagens geradas para cada uma das cinco classes analisadas. Observa-se que, dentro de cada classe, as curvas seguem um padrão interno consistente, sugerindo que o modelo generativo foi capaz de produzir variações estruturais coerentes e diversificadas.

Figura 41 - Curvas de intensidade extraídas de 12 imagens geradas da classe 1.33

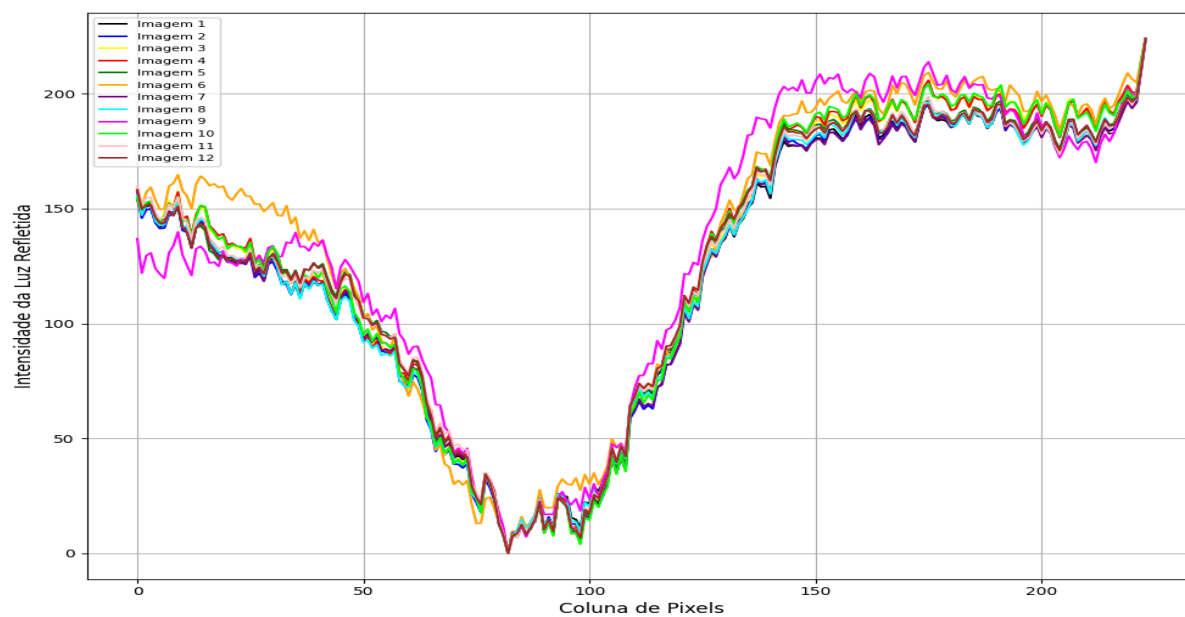


Figura 42 – Curvas de intensidade extraídas de 12 imagens geradas da classe 1.335

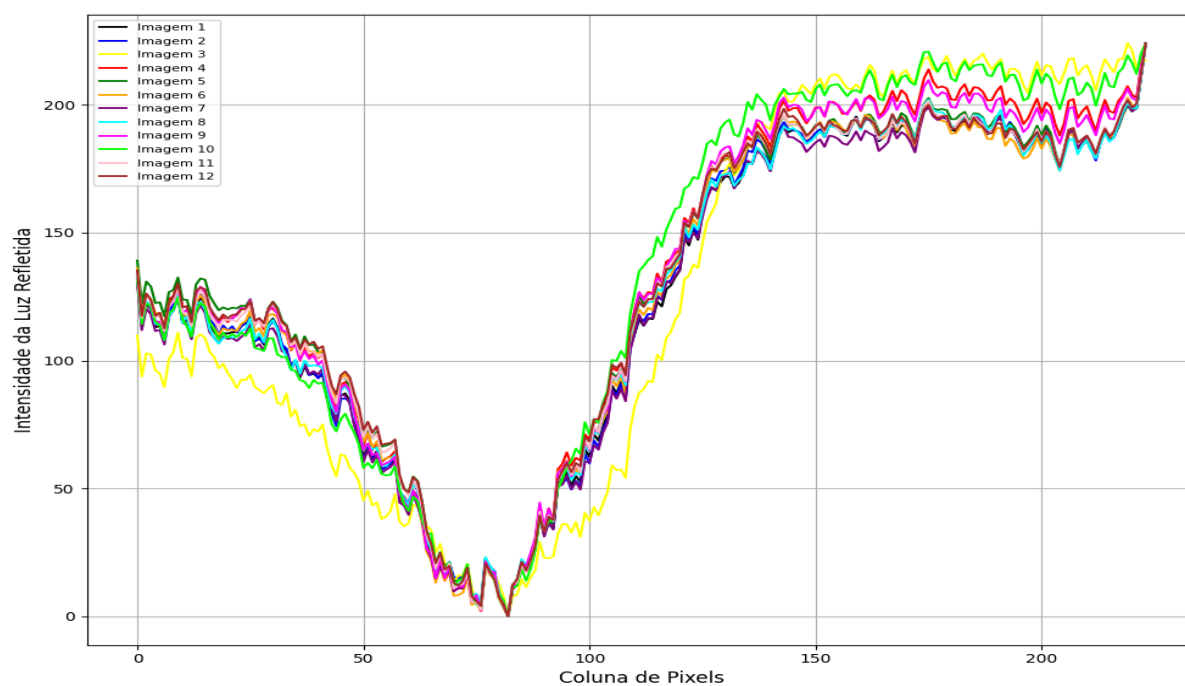


Figura 43 – Curvas de intensidade extraídas de 12 imagens geradas da classe 1.34

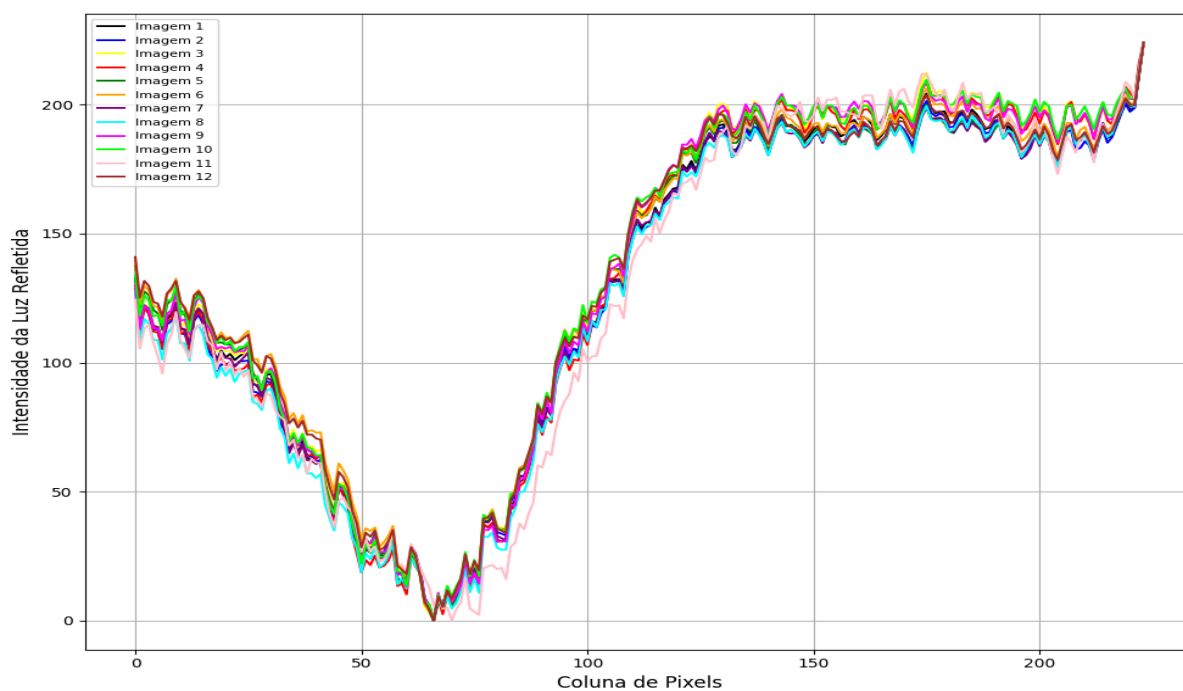


Figura 44 – Curvas de intensidade extraídas de 12 imagens geradas da classe 1.345

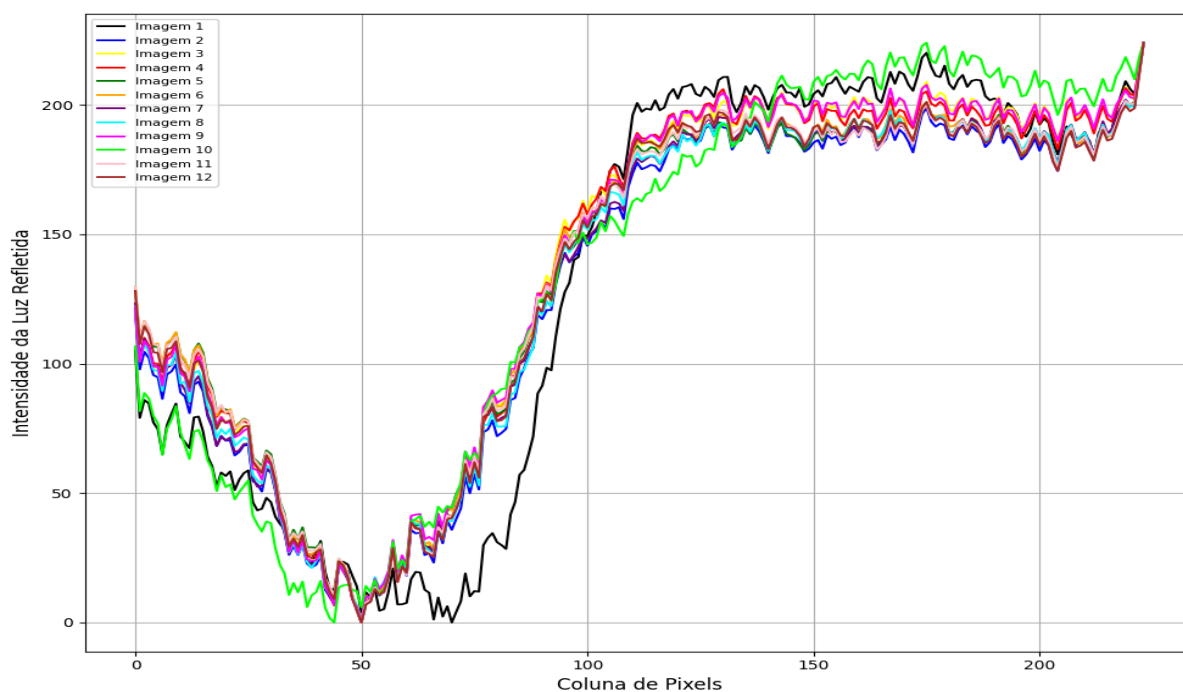
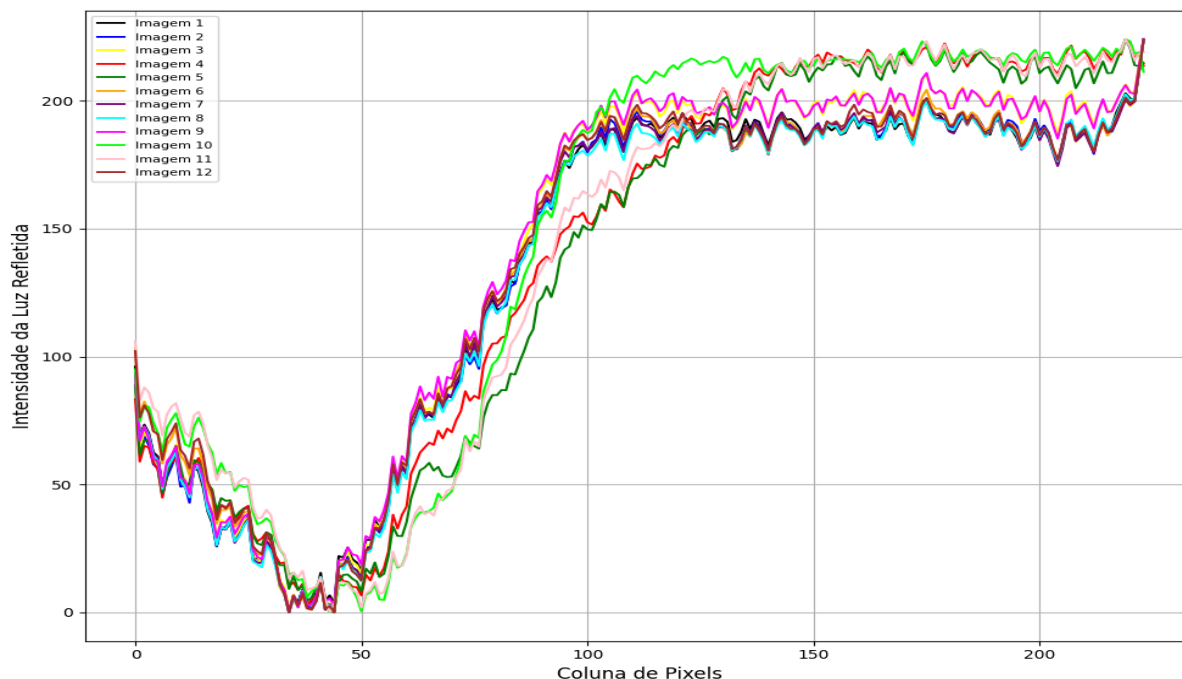


Figura 45 – Curvas de intensidade extraídas de 12 imagens geradas da classe 1.35



Observa-se que as curvas seguem um padrão similar, evidenciando que o modelo conseguiu capturar e reproduzir a morfologia luminosa típica da classe. Pequenas variações entre as curvas indicam diversidade natural entre as amostras, o que é desejável em modelos generativos.

4.1.5 Avaliação Individual

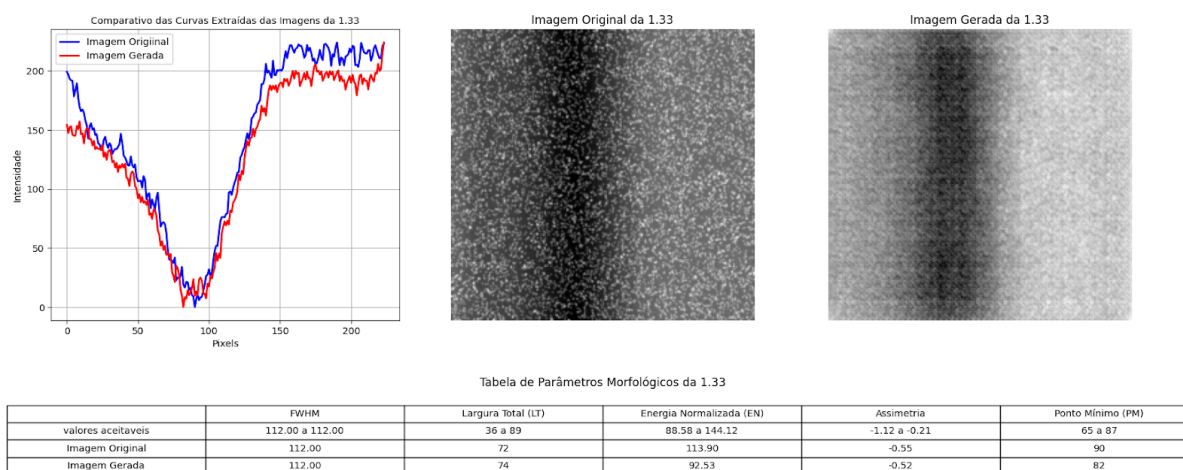
Esta análise tem como objetivo avaliar a fidelidade de imagens geradas por um modelo computacional em relação a imagens reais, com base em parâmetros morfológicos e de intensidade. Os parâmetros observados estão distribuídos nos seguintes intervalos: FWHM, Largura Total (LT), Energia Normalizada (EN), Assimetria (A) e o Ponto Mínimo (PM).

Em seguida, uma imagem gerada de cada classes 1.33, 1.335, 1.34, 1.345 e 1.35 foi selecionada aleatoriamente para comparação com os valores da imagem real da mesma classe e também comparar com os valores aceitáveis de cada classe. A Figura 46 a 50 apresenta o gráfico comparativo entre as curvas da imagem gerada e da imagem real, em relação a

FWHM, largura total, energia normalizada, assimetria e o ponto mínimo entre as imagens original e real.

É importante observar que os intervalos de referência utilizados para avaliar os parâmetros morfológicos foram estabelecidos com base exclusivamente nas imagens reais do conjunto de treinamento, considerando os valores mínimo e máximo encontrados para cada classe. Assim, é natural que, ao se analisar imagens reais pertencentes ao conjunto de teste, alguns valores possam ultrapassar os limites previamente definidos, como observado em alguns casos.

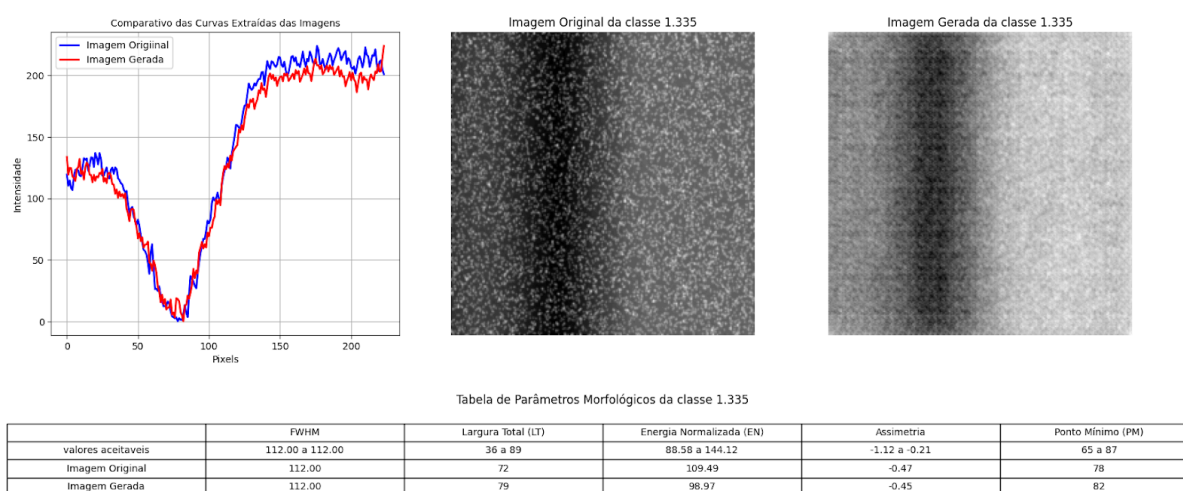
Figura 46 - Comparativo das curvas extraídas das imagens, comparação das imagens original e gerada e comparação dos parâmetros da classe 1.33



Embora a curva da imagem gerada (vermelha) apresente uma intensidade ligeiramente inferior nas regiões inicial e final do perfil, sua forma global é bem preservada em relação à curva da imagem real. Essa diferença sutil pode estar associada a um efeito de suavização ou atenuação introduzido pelo modelo gerador, possivelmente decorrente de mecanismos de regularização ou da presença de ruído no processo de síntese.

Além disso, a imagem gerada para a classe 1.335 demonstrou forte consistência morfológica em comparação com a original, apresentando valores de FWHM, Largura Total (LT), Energia Normalizada (EN), Assimetria e Ponto Mínimo (PM) todos dentro dos limites aceitáveis definidos para essa classe. Esses resultados reforçam a capacidade do modelo em gerar imagens sintéticas com características compatíveis com as amostras reais.

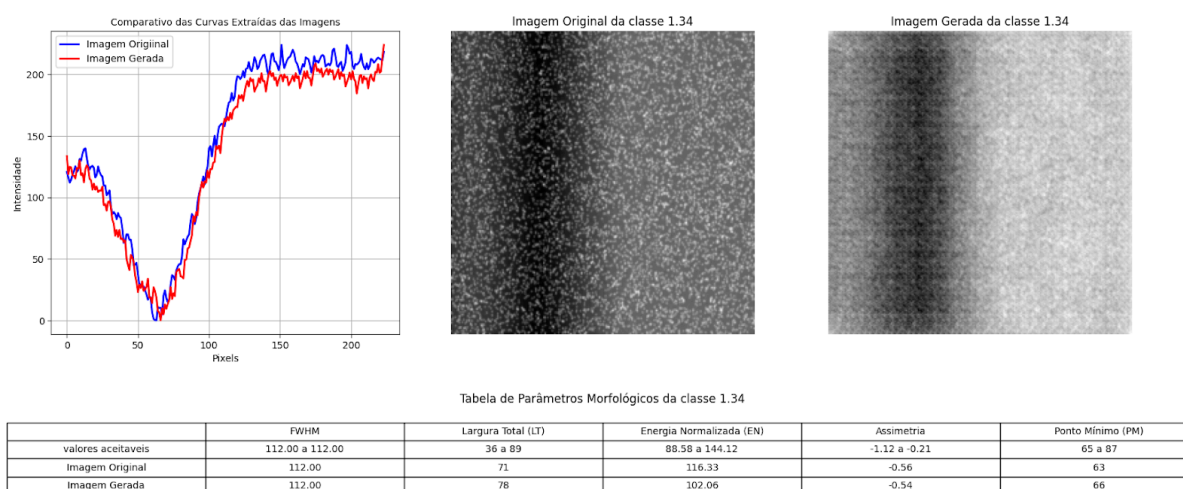
Figura 47 - Comparativo das curvas extraídas das imagens, comparação das imagens original e gerada e comparação dos parâmetros da classe 1.335



Pequenas diferenças entre as curvas são perceptíveis, especialmente nas regiões com ruído ou variações locais, o que é esperado em processos de geração sintética. No entanto, essas discrepâncias não comprometem a similaridade global entre os perfis.

Além disso, a análise é reforçada pela tabela de parâmetros morfológicos, que mostra que todos os valores extraídos da imagem gerada estão dentro ou muito próximos dos intervalos aceitáveis da classe 1.335.

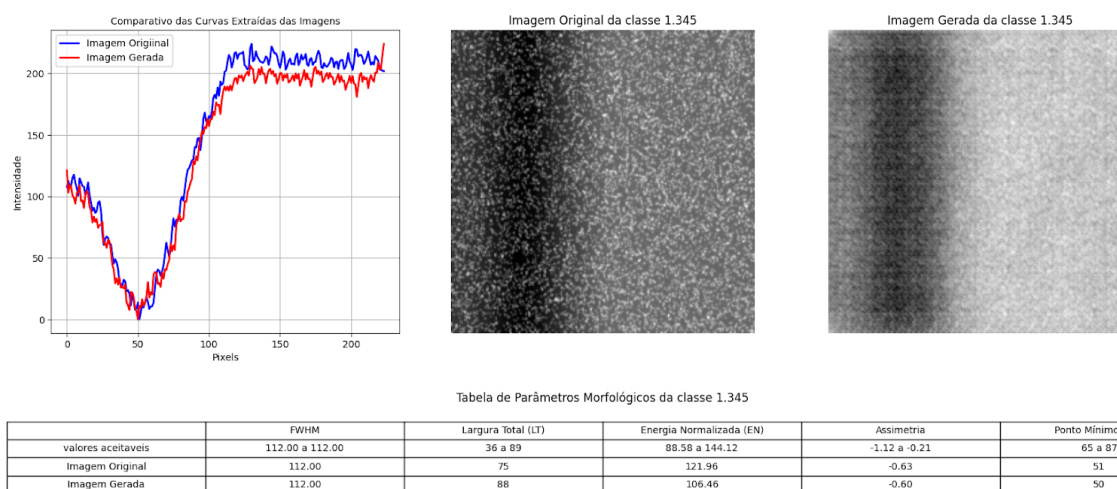
Figura 48 - Comparativo das curvas extraídas das imagens, comparação das imagens original e gerada e comparação dos parâmetros da classe 1.34



Embora a curva de intensidade da imagem gerada (vermelha) apresente valores ligeiramente inferiores em comparação à curva da imagem real (azul), sobretudo nas regiões mais periféricas do perfil, a forma geral da curva é bem preservada. A proximidade entre os padrões indica que a imagem gerada mantém a estrutura morfológica característica da classe 1.34. Essa diferença pode estar relacionada a um efeito de suavização introduzido pelo modelo gerador, possivelmente decorrente de processos de regularização ou da redução do ruído de alta frequência.

Do ponto de vista quantitativo, a imagem gerada para a classe 1.34 apresentou parâmetros morfológicos bastante compatíveis com a imagem real. Todos os valores FWHM (112.00), Largura Total (78), Energia Normalizada (102.06), Assimetria (-0.54) e Ponto Mínimo (66) encontram-se dentro dos limites definidos para a classe.

Figura 49 - Comparativo das curvas extraídas das imagens, comparação das imagens original e gerada e comparação dos parâmetros da classe 1.345



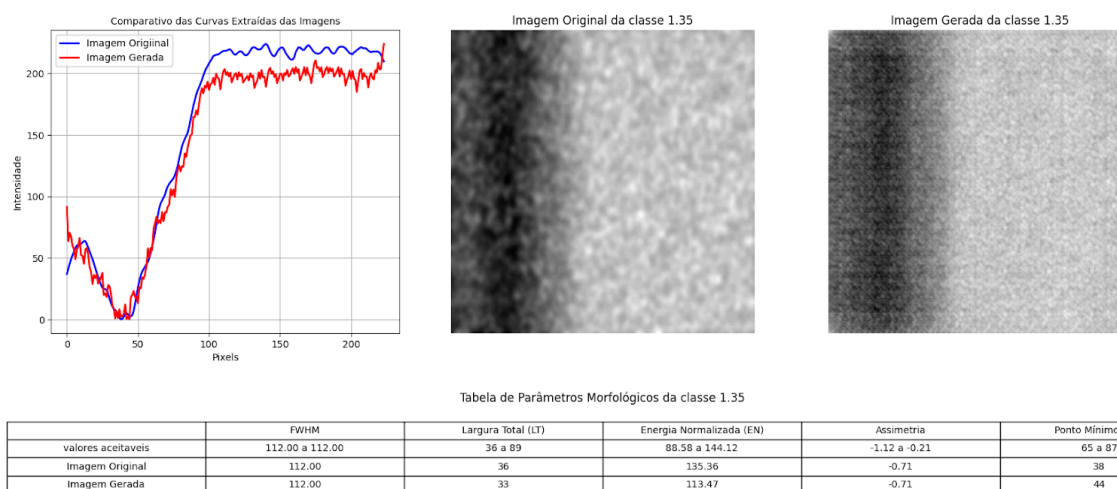
Na figura 49, observa-se que o modelo gerador conseguiu manter a forma geral da curva de intensidade, com boa correspondência entre os perfis, apesar de pequenas variações nos níveis iniciais e finais, o que pode estar relacionado à suavização típica de processos de geração por redes neurais.

Em termos quantitativos, os valores de FWHM, Largura Total (LT), Energia Normalizada (EN) e Assimetria (A) da imagem gerada estão todos dentro dos limites

aceitáveis definidos para a classe, demonstrando consistência com as características morfológicas aprendidas durante o treinamento.

No entanto, o parâmetro Ponto Mínimo (PM) da imagem gerada apresentou um valor de 50, consideravelmente abaixo do intervalo de referência (65 a 87). Esse mesmo comportamento foi observado também na imagem real do conjunto de teste, que apresentou valor de 51, ligeiramente superior ao da imagem gerada, mas igualmente fora da faixa aceitável. Esse desvio reforça a observação de que, como os limites de referência foram estabelecidos exclusivamente com base nas imagens do conjunto de treinamento, é natural que imagens pertencentes ao conjunto de teste apresentem valores fora desses limites. Ainda assim, o fato de o modelo reproduzir esse comportamento sugere que ele foi capaz de capturar aspectos estatísticos relevantes da distribuição original.

Figura 50 - Comparativo das curvas extraídas das imagens, comparação das imagens original e gerada e comparação dos parâmetros da classe 1.35



Na Figura 50, observa-se uma boa correspondência entre as curvas de intensidade, com a imagem gerada reproduzindo a forma geral da curva da imagem real. No entanto, há uma ligeira atenuação na intensidade final da curva da imagem gerada, o que pode ser atribuído a efeitos de suavização introduzidos pelo modelo gerador.

Na avaliação quantitativa, o modelo conseguiu replicar com precisão o valor de FWHM, mantendo-o exatamente dentro do intervalo esperado. Os parâmetros de Energia

Normalizada (EN) e Assimetria (A) também permaneceram dentro dos limites aceitáveis, demonstrando boa fidelidade morfológica.

Contudo, os valores de Largura Total (LT) e Ponto Mínimo (PM) da imagem gerada ficaram abaixo dos respectivos intervalos de referência da classe. A LT foi de 33, inferior ao mínimo aceitável de 36, e o PM foi de 44, consideravelmente abaixo do mínimo de 65. Vale destacar que a imagem real da classe também apresentou PM de 38, reforçando que a própria amostra de teste não está completamente contida no intervalo definido com base apenas no conjunto de treinamento. Esse fato evidencia a limitação natural de intervalos definidos exclusivamente a partir do conjunto de treinamento e mostra que o modelo foi capaz de reproduzir com certa fidelidade até mesmo variações presentes nas imagens do conjunto de teste.

4.2 Geração de Novas Classes

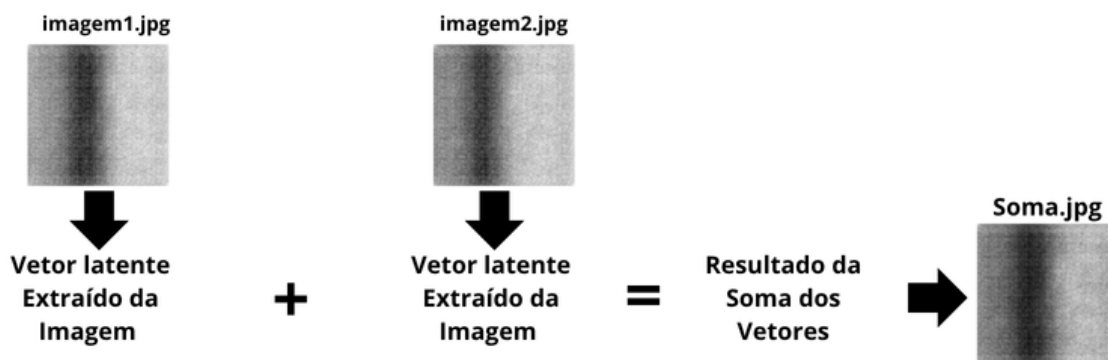
A geração de novas classes, utilizando o modelo proposto, envolve a manipulação do espaço latente para criar variações visuais entre imagens. Técnicas como operações vetoriais, interpolação e transições entre vetores são aplicadas com o objetivo de explorar combinações de características e transições contínuas entre diferentes representações. Essas abordagens contribuem para a diversificação dos dados sintéticos gerados e para a análise da estrutura semântica aprendida pelo modelo.

4.2.1 Operações com Vetores latente

O espaço latente aprendido por modelos generativos, como o DCGAN, permite realizar operações vetoriais com significados visuais. Nesse contexto, é possível explorar manipulações semânticas por meio de operações aritméticas entre vetores latentes, como soma ou subtração. Essas operações visam estudar a capacidade do modelo em capturar relações visuais implícitas no espaço de representação.

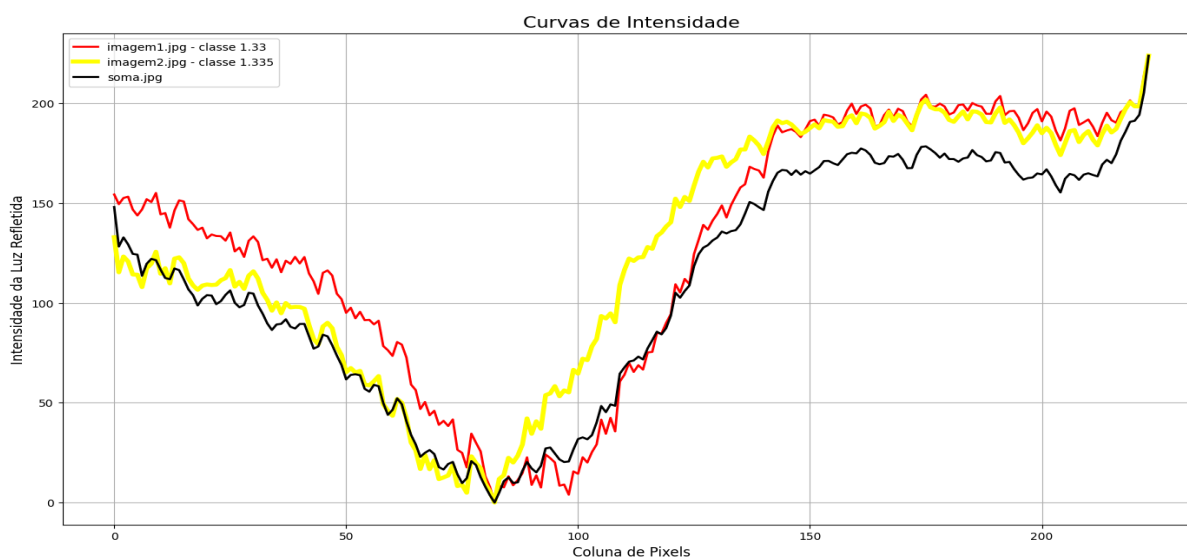
A Figura 51 apresentadas duas imagens: a primeira, pertencente à classe 1.33, com o nome de *imagem1.jpg*; e a segunda, da classe 1.335, com o nome de *imagem2.jpg*. Na parte inferior, observa-se a extração dos vetores latentes de cada imagem e a combinação vetorial entre eles, resultando na geração da imagem final, denominada *Soma.jpg*.

Figura 51 - Apresenta a adição de dois vetores latentes das classes 1.33 e 1.335



A Figura 52 complementa a análise da Figura 51 ao mostrar a distribuição da intensidade da luz refletida nas colunas de pixels de três imagens: classe 1.33, classe 1.335 e a imagem gerada por soma vetorial (*soma.jpg*). A curva da imagem soma (linha preta) apresenta perfil intermediário entre as originais. Em profundidade, o vale de refletância atinge níveis semelhantes aos mínimos das imagens reais. Já na forma, observa-se uma fusão assimétrica: o lado esquerdo do vale segue o padrão da classe 1.335 (linha amarela) e o direito se assemelha à classe 1.33 (linha vermelha).

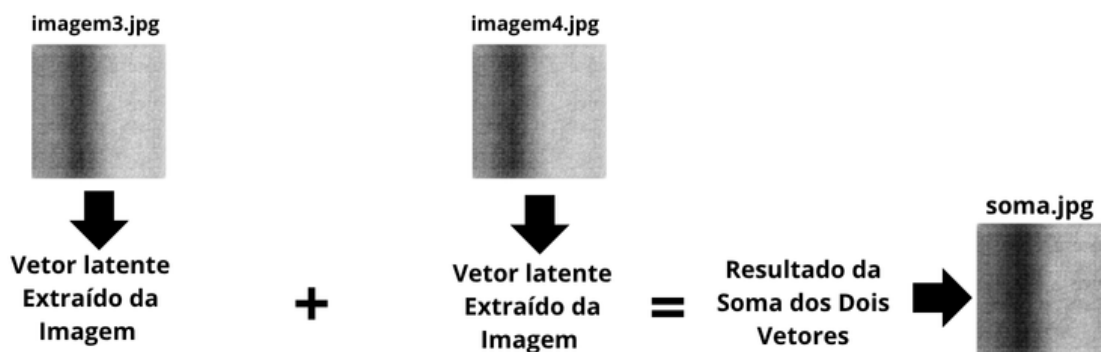
Figura 52 – Análise das curvas de intensidade luminosa das três imagens



A Figura 53 apresenta o mesmo experimento de soma vetorial realizado anteriormente, agora utilizando imagens das classes 1.335 e 1.34. Após a extração e soma dos vetores

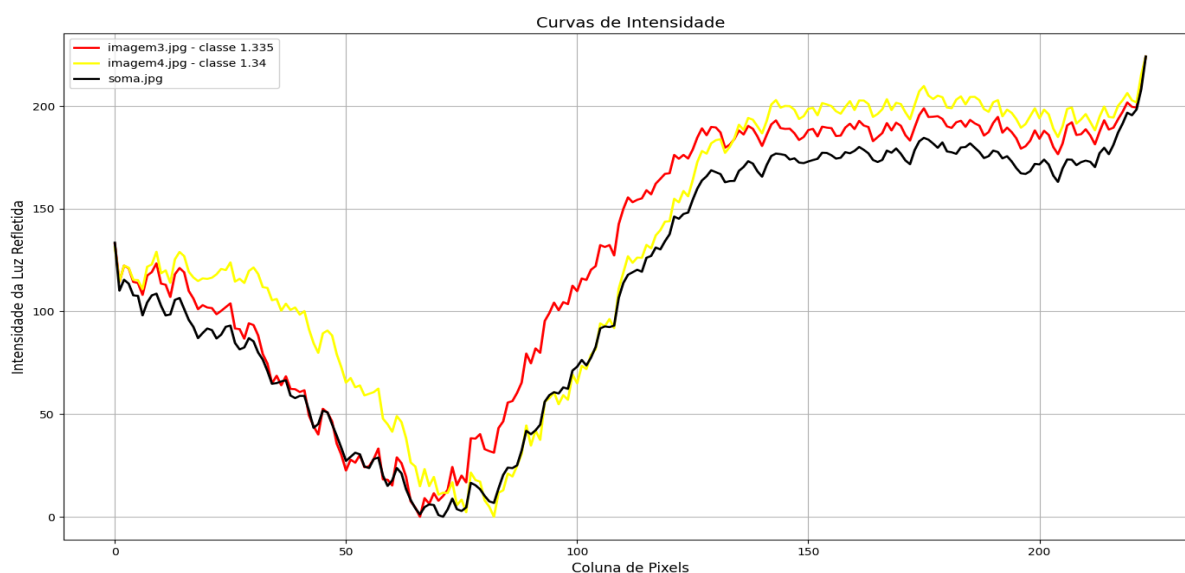
latentes correspondentes, foi gerada a imagem final (*soma.jpg*), cuja interpretação pode ser feita com base nas curvas resultantes.

Figura 53 - apresenta a adição de dois vetores latentes das classes 1.335 e 1.34



Complementando a análise apresentada na Figura 53, a Figura 54 ilustra a distribuição da intensidade da luz refletida ao longo das colunas de pixels para três imagens distintas: a imagem da classe 1.335 (*imagem1.jpg*, linha vermelha), a imagem da classe 1.34 (*imagem2.jpg*, linha amarela) e a imagem gerada pela soma vetorial (*soma.jpg*, linha preta). A curva da *soma.jpg* apresenta um comportamento intermediário entre as duas classes de origem.

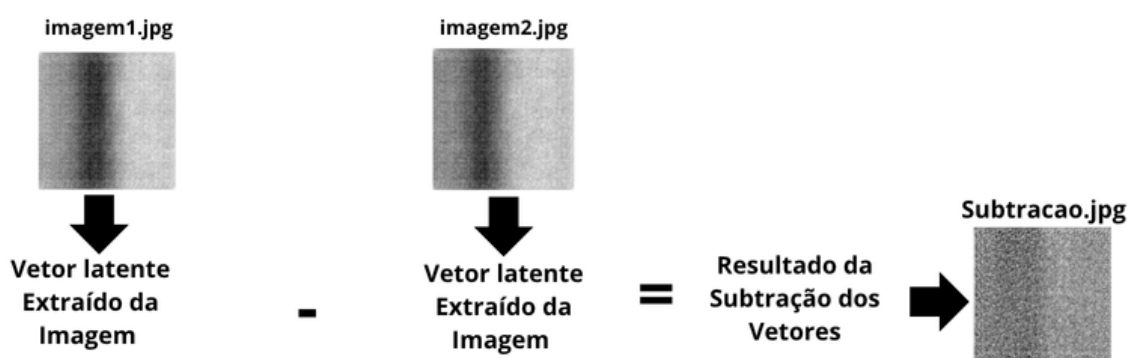
Figura 54 - Análise das curvas de intensidade luminosa das três image



Agora vamos aplicar a subtração de vetores latentes provenientes de diferentes classes, é possível gerar novas imagens que representam a diferença visual entre as características originais de cada classe. Essa abordagem permite investigar como as informações são diferenciadas no espaço latente, revelando quais atributos são exclusivos ou modificados entre as representações.

A Figura 55 ilustra uma subtração entre dois vetores. Na parte superior, são apresentadas duas imagens: a primeira, pertencente à classe 1.33, com o nome de *imagem1.jpg*; e a segunda, da classe 1.335, com o nome de *imagem2.jpg*. Na parte inferior, observa-se a extração dos vetores latentes de cada imagem e a combinação vetorial entre eles, resultando na geração da imagem final, denominada *subtracao.jpg*.

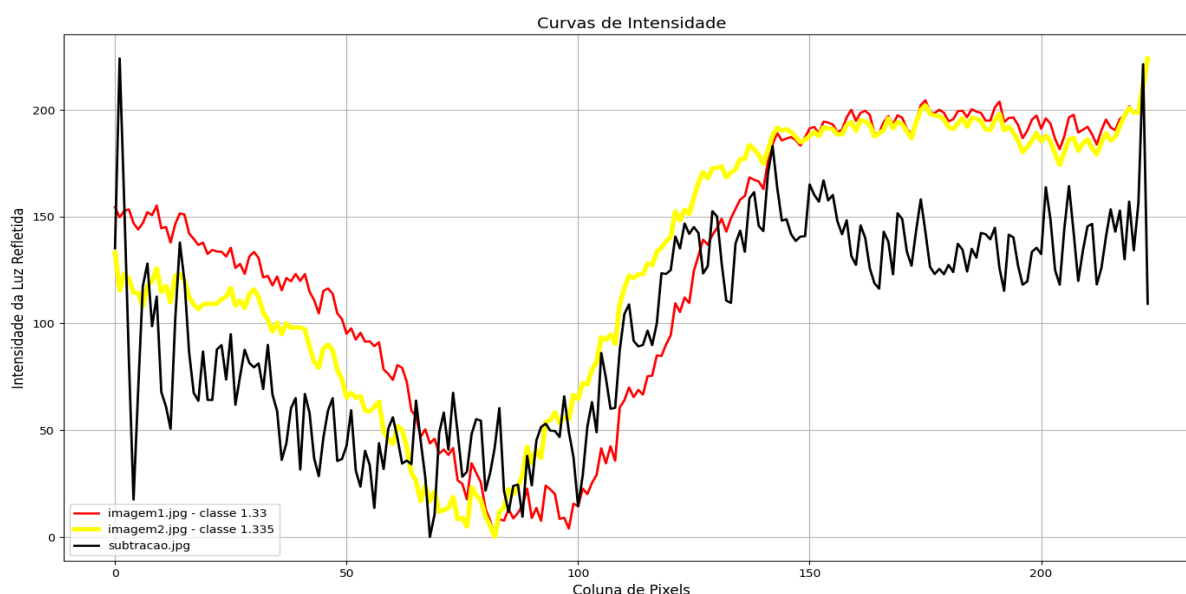
Figura 55 - apresenta a subtração de dois vetores latentes das classes 1.33 e 1.335.



Complementando a análise apresentada na Figura 55, a Figura 56 ilustra a distribuição da intensidade da luz refletida ao longo das colunas de pixels para três imagens distintas: a imagem da classe 1.33 (*imagem1.jpg*, linha vermelha), a imagem da classe 1.335 (*imagem2.jpg*, linha amarela) e a imagem gerada pela subtração vetorial entre essas duas classes (*subtracao.jpg*, linha preta). A curva correspondente à *subtracao.jpg* apresenta um padrão distinto, marcado por grande variação de intensidade e flutuações abruptas, especialmente nas regiões de menor refletância.

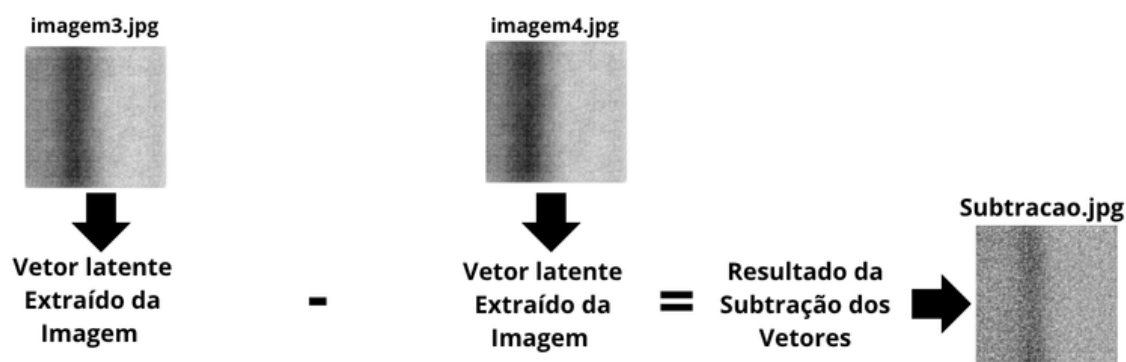
A avaliação da imagem gerada por subtração vetorial revelou uma significativa degradação estrutural em comparação com as imagens originais das classes 1.33 e 1.335. A curva de intensidade refletida da imagem subtraída apresentou um comportamento altamente oscilante e menos suave, evidenciando perda de coesão visual.

Figura 56 - Análise das curvas de intensidade luminosa das três imagens



Por fim, a Figura 57 ilustra outra subtração vetorial. Na parte superior, são apresentadas duas imagens: a primeira, pertencente à classe 1.335, com o nome de *imagem3.jpg*; e a segunda, da classe 1.34, com o nome de *imagem4.jpg*. Na parte inferior, observa-se a extração dos vetores latentes de cada imagem e a combinação vetorial entre eles, resultando na geração da imagem final, denominada *Subtracao.jpg*.

Figura 57 - apresenta a subtração de dois vetores latentes das classes 1.335 e 1.34.

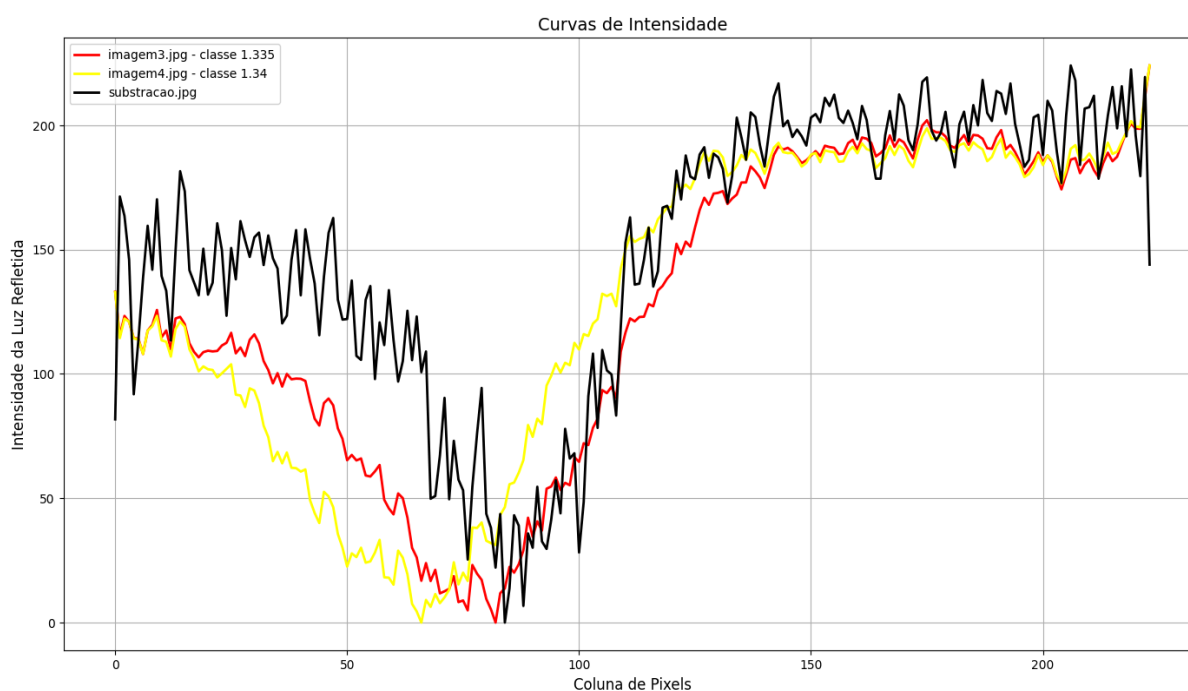


A Figura 58 ilustra a distribuição da intensidade da luz refletida ao longo das colunas de pixels para três imagens distintas: a imagem da classe 1.335 (*imagem3.jpg*, linha vermelha), a imagem da classe 1.34 (*imagem4.jpg*, linha amarela) e a imagem gerada pela subtração vetorial entre essas duas classes (*subtracao.jpg*, linha preta). A curva

correspondente à *subtracao.jpg* revela um comportamento irregular e altamente oscilante, especialmente nas regiões de menor refletância.

A avaliação da imagem gerada por subtração vetorial entre as classes 1.335 e 1.34 evidenciou uma degradação estrutural significativa em relação às imagens originais. A curva de intensidade refletida da imagem resultante apresentou um padrão altamente oscilante e irregular, indicando perda de suavidade e coerência visual.

Figura 58 - Análise das curvas de intensidade luminosa das três imagens



Os experimentos com operações vetoriais no espaço latente, especificamente somas e subtrações de vetores oriundos de diferentes classes, evidenciaram tanto a capacidade quanto os limites do modelo DCGAN em representar relações estruturais e ópticas entre distintos padrões de imagem. As somas vetoriais apresentaram comportamento consistente e alinhado nas curvas de intensidade refletidas, indicando que o modelo foi capaz de sintetizar imagens com morfologia e distribuição óptica coerentes, resultando em uma fusão visual gradual entre as classes envolvidas.

Em contrapartida, as subtrações vetoriais revelaram efeito oposto: as imagens geradas exibiram fortes oscilações nas curvas de intensidade refletidas. Conclui-se, portanto, que as operações aritméticas no espaço latente constituem um ferramental útil e interpretável para manipular e avaliar as capacidades do modelo generativo. Enquanto a soma se mostrou eficaz na geração de imagens intermediárias, a subtração evidenciou limitações ligadas à estabilidade da representação latente e à preservação de atributos estruturais. Esse comportamento pode estar relacionado ao fato de que a subtração não preserva, de forma consistente, os atributos das classes originais, gerando instabilidades na síntese das imagens e acentuando diferenças em vez de produzir uma representação coerente.

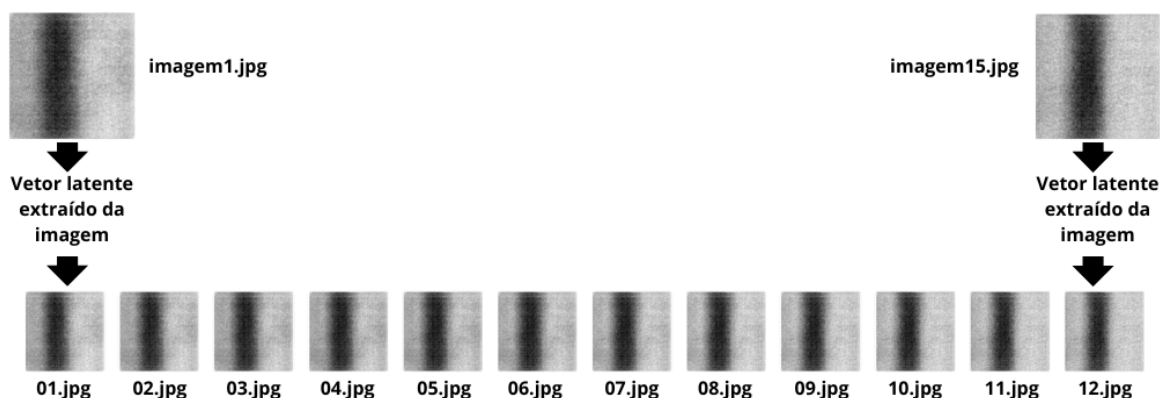
4.2.2 Interpolações Entre Vetores Latentes

A interpolação linear e esférica de vetores latentes utilizando redes DCGAN é uma técnica fundamental para avaliar e explorar a qualidade e a estrutura do espaço latente aprendido pelo modelo gerador. Por meio da interpolação entre dois vetores latentes, é possível gerar uma sequência contínua de imagens que apresentam uma fusão progressiva de características visuais entre duas amostras sintéticas. Essa abordagem permite verificar se o modelo aprendeu uma representação coerente dos dados, na qual pequenas variações no espaço latente resultam em mudanças graduais e consistentes nas imagens geradas.

A interpolação linear consiste em calcular combinações convexas entre dois vetores latentes, traçando uma linha reta entre eles no espaço latente. Embora seja uma abordagem computacionalmente simples e direta, essa técnica pode atravessar regiões de baixa densidade de dados, o que pode resultar em imagens intermediárias com menor qualidade.

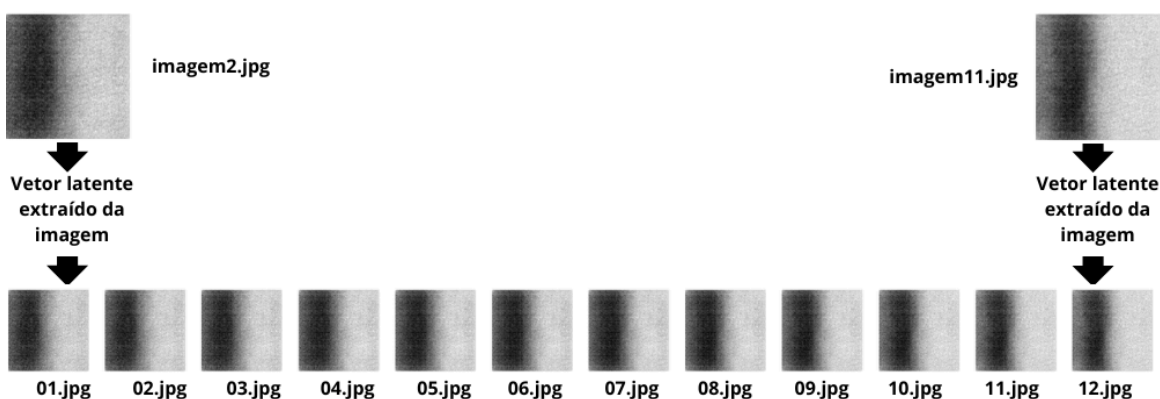
Figura 59 apresenta uma interpolação linear entre dois vetores latentes. O primeiro vetor foi extraído da imagem (*imagem1.jpg*), da classe 1.33 (à esquerda), e o segundo da imagem (*imagem15.jpg*), da classe 1.335 (à direita). Logo abaixo, é exibida uma sequência de 12 imagens geradas artificialmente, baseadas nesses vetores-alvo. Essas imagens representam a interpolação linear entre as imagens no espaço latente.

Figura 59 - interpolação linear entre dois vetores das classes 1.33 e 1.335



Outro exemplo de interpolação linear é apresentado na Figura 60 entre dois vetores latentes. O primeiro vetor foi extraído da imagem (*imagem2.jpg*), da classe 1.335 (à esquerda), e o segundo imagem (*imagem11.jpg*), da classe 1.34 (à direita). Logo abaixo, é exibida uma sequência de 12 imagens geradas artificialmente, baseadas nesses vetores-alvo. Essas imagens representam a interpolação linear entre as imagens no espaço latente.

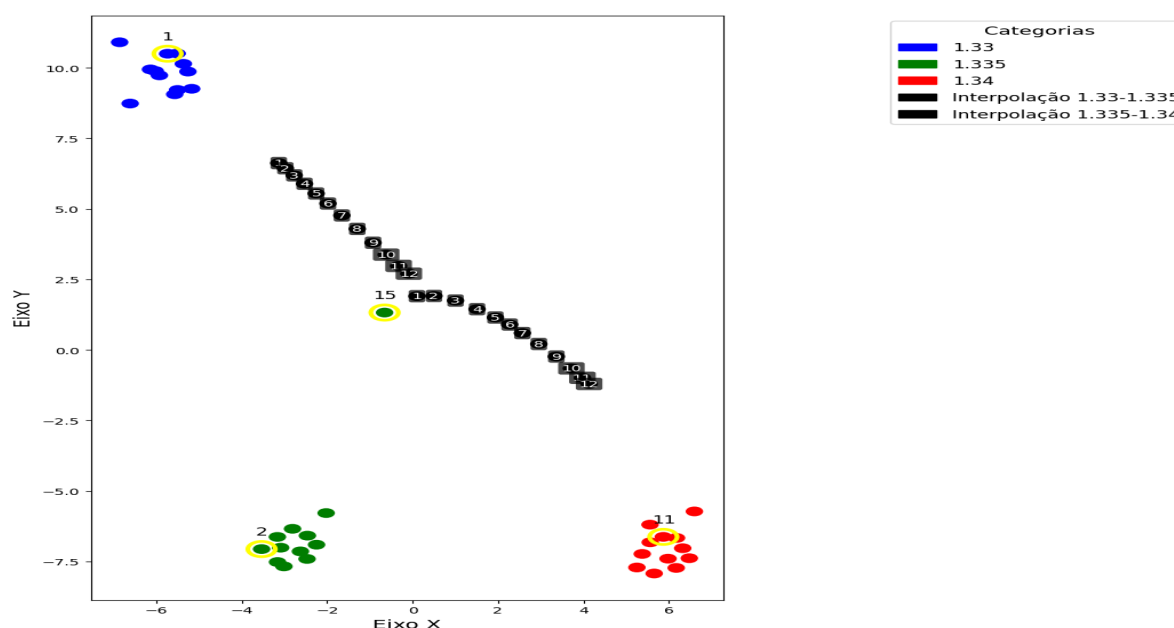
Figura 60 - interpolação linear entre dois vetores das classes 1.335 e 1.34



Complementando a análise das Figura 59 e 60, a Figura 61, apresenta uma representação bidimensional das imagens geradas pela DCGAN para três classes distintas: 1.33 (em azul, com 12 amostras), 1.335 (em verde, com 12 amostras) e 1.34 (em vermelho, também com 12 amostras). Além disso, são exibidas duas sequências de imagens interpoladas: a primeira entre as classes 1.33 e 1.335, e a segunda entre as classes 1.335 e 1.34, ambas representadas por quadrados pretos numerados de 1 a 12. Os pontos de início e fim de cada interpolação estão destacados com círculos amarelos, indicando os vetores

latentes reais que deram origem às trajetórias. As duas sequências formam caminhos contínuos no espaço latente, conectando gradualmente as distribuições das classes adjacentes. O método t-SNE foi configurado com perplexidade 12 e semente aleatória fixada em 0.

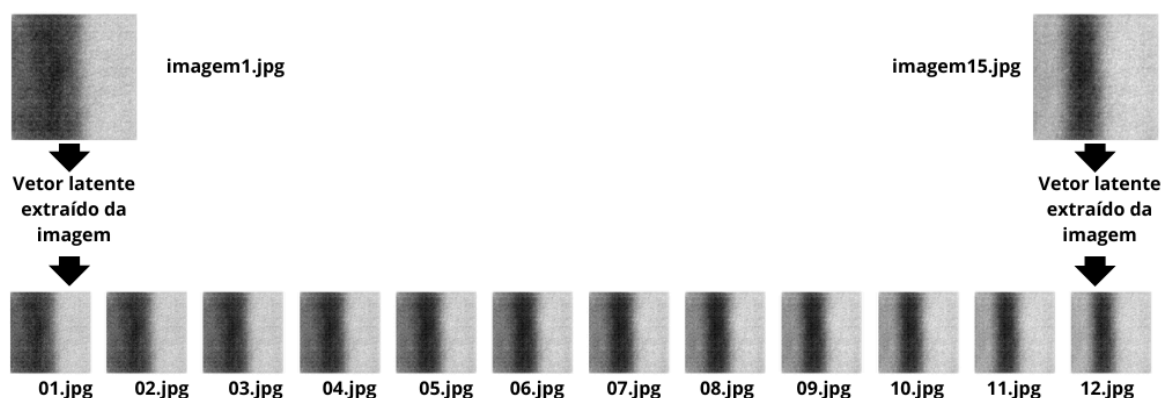
Figura 61 - Trajetórias de interpolação linear entre as classes 1.33 - 1.335 e 1.335 - 1.34, visualizadas via t-SNE



Além da interpolação linear, também foi utilizada a *interpolação esférica* para gerar transições entre vetores no espaço latente. Nesse método, os vetores são considerados como pontos sobre a superfície de uma esfera, e a transição ocorre ao longo de um arco, em vez de uma linha reta. Essa abordagem permite que a interpolação permaneça dentro da região mais confiável do espaço latente, resultando em uma transição mais suave e realista entre as imagens geradas.

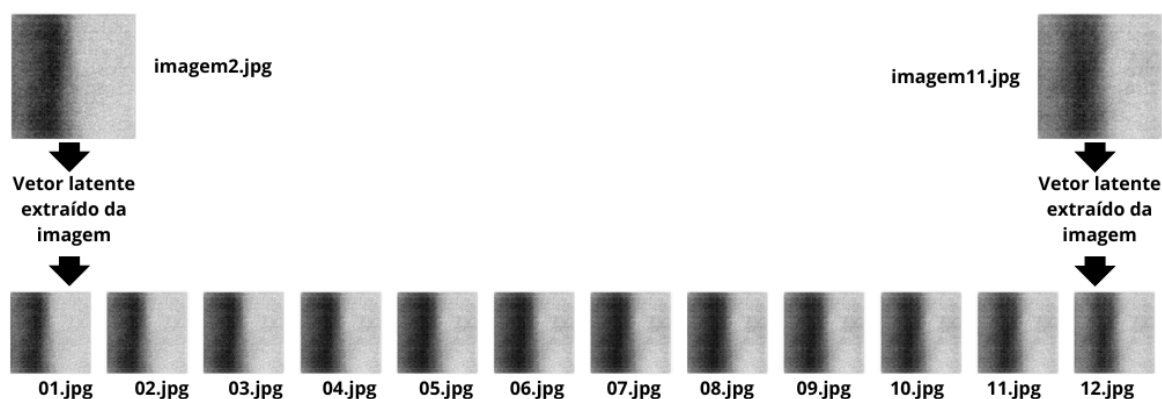
Na Figura 62, apresenta uma interpolação esférica entre os vetores latentes de duas imagens. O primeiro vetor foi extraído da imagem (*imagem1.jpg*), da classe 1.33, e o segundo da imagem (*imagem15.jpg*), da classe 1.335. Logo abaixo, é exibida uma sequência de 12 imagens geradas artificialmente, baseadas nesses vetores-alvo. Essas imagens representam a interpolação esférica entre as imagens no espaço latente.

Figura 62 - interpolação esférica entre dois vetores das classes 1.33 e 1.335



Também na Figura 63 ilustra mais um caso de interpolação esférica entre dois vetores latentes. O primeiro foi obtido a partir de uma imagem da classe 1.335 (*imagem2.jpg*) e o segundo, de uma imagem da classe 1.34 (*imagem11.jpg*). Abaixo, é apresentada uma sequência com 12 imagens geradas artificialmente com base nesses vetores, refletindo a transição esférica entre as representações no espaço latente.

Figura 63 - interpolação esférica entre dois vetores das classes 1.335 e 1.34



Para representar o posicionamento das imagens geradas por interpolação esférica, aplicou-se a técnica t-SNE aos dados apresentados nas Figuras 62 e 63. O resultado dessa projeção está ilustrado na Figura 63. Embora o t-SNE não preserve a geometria esférica original do espaço latente, sua aplicação possibilita visualizar a transição morfológica entre as classes no espaço da imagem, evidenciando a continuidade perceptiva entre os extremos e a suavidade da transição promovida por essa abordagem.

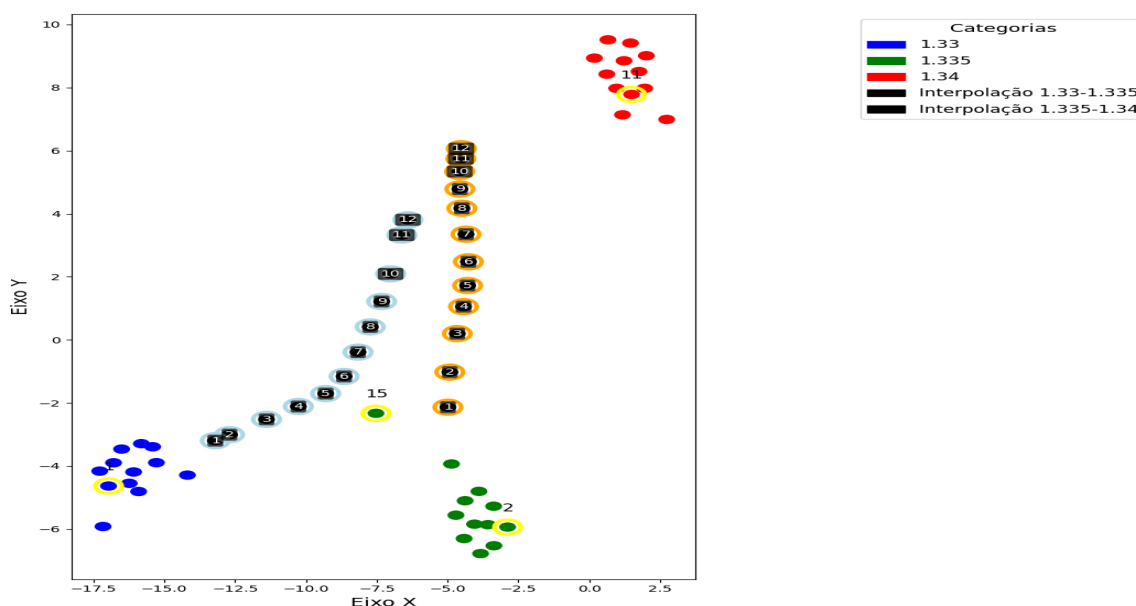
A Figura 64 apresenta uma representação bidimensional das imagens geradas pela DGAN, abrangendo as classes 1.33 (em azul, com 12 amostras), 1.335 (em verde, também com 12 amostras) e 1.34 (em vermelho, com 12 amostras). Além das amostras originais, a figura inclui os resultados da interpolação esférica realizada entre as classes 1.33 e 1.335, e entre as classes 1.335 e 1.34.

Na primeira interpolação, foram selecionadas duas imagens: 1.jpg, pertencente à classe 1.33, e 15.jpg, da classe 1.335. A partir dessas imagens extremas, foi aplicada uma interpolação esférica, gerando imagens intermediárias numeradas de 1.jpg a 12.jpg. Esses pontos interpolados estão representados na figura por pontos pretos numerados de 1 a 12, circulos em azul, indicando o início e o fim da transição entre as classes 1.33 e 1.335.

Na segunda interpolação, utilizaram-se as imagens 2.jpg, da classe 1.335, e 11.jpg, da classe 1.34. Novamente, foi realizada uma interpolação esférica, resultando em imagens intermediárias também numeradas de 1.jpg a 12.jpg. Esses pontos estão indicados por pontos pretos numerados de 1 a 12, desta vez circulos em laranja, representando a transição entre as classes 1.335 e 1.34.

O t-SNE foi configurado com perplexidade 12 e semente aleatória fixada em 0. Essa parametrização permitiu observar de forma mais clara a continuidade perceptiva entre as classes, mesmo que a técnica não preserve diretamente a geometria esférica do espaço latente.

Figura 64 - Trajetórias de interpolação esférica entre Classes 1.33, 1.335 e 1.34 visualizadas via t-SNE



As visualizações indicam que, embora a interpolação esférica favoreça trajetórias suaves no espaço latente, os vetores interpolados nem sempre seguem rotas diretas entre as classes. Isso sugere que o espaço latente aprendido pode conter regiões distorcidas ou não lineares, dificultando uma transição precisa entre as representações das classes. Ainda assim, os resultados confirmam que a interpolação funciona como ferramenta de análise, permitindo verificar a coerência do espaço latente. A interpolação linear evidenciou limitações ao atravessar regiões de baixa densidade, enquanto a esférica apresentou transições mais suaves e consistentes. Dessa forma, ambas revelam tanto o potencial quanto às restrições do modelo em capturar a estrutura contínua dos dados.

4.2.3 Transição Entre Vetores Latente

O processo de transição consiste na busca de um vetor latente que, ao ser passado pelo gerador (G), produza uma imagem o mais semelhante possível a uma imagem-alvo. Isso é feito por meio de um processo de otimização, no qual um vetor latente inicial é ajustado iterativamente utilizando gradiente descendente com base no erro entre a imagem gerada e a imagem-alvo.

Inicialmente, utiliza-se um vetor latente de 100 dimensões, extraído da imagem-alvo e configurado como um parâmetro otimizável. Esse vetor é atualizado em múltiplas iterações com o otimizador Adam, visando minimizar a função de perda baseada no Erro Quadrático Médio.

Para evitar otimizações desnecessárias, ou seja, continuar atualizando o vetor latente mesmo quando já não há melhoria significativa, é empregado um critério de parada antecipada, que interrompe o processo caso a função de perda não apresente avanço por cinco épocas consecutivas. A cada época, o vetor latente e a imagem gerada são armazenados, permitindo a visualização da transição gradual da imagem inicial para uma representação mais próxima da imagem-alvo.

Durante os experimentos, observou-se dois comportamentos distintos no processo de transição: em alguns casos, houve parada antecipada antes do limite máximo de 120 épocas, indicando rápida convergência; em outros, o processo completou todas as épocas disponíveis. Esses casos são detalhados a seguir.

4.2.3.1 Casos com Parada Antecipada

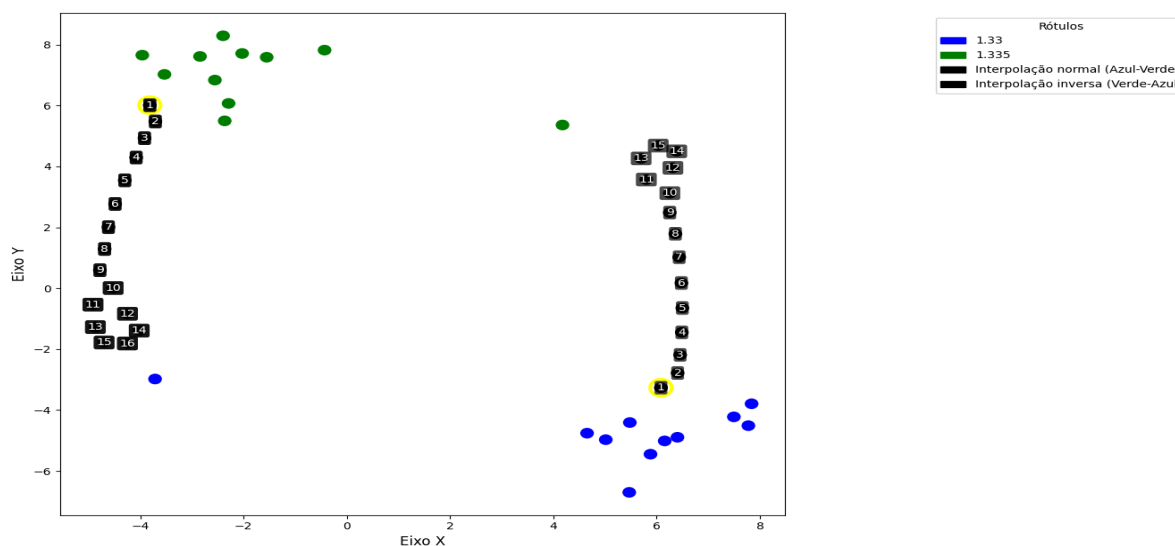
Em determinadas transições, o processo foi interrompido automaticamente antes de atingir o limite máximo de 120 épocas. Isso ocorreu quando o critério de parada antecipada detectou estagnação na melhoria da perda, sugerindo que o vetor latente havia alcançado uma configuração suficientemente próxima da imagem-alvo. Esses casos indicam que, para algumas imagens, a otimização é eficiente e converge rapidamente, possivelmente devido à boa posição inicial do vetor latente ou à menor complexidade da imagem.

A Figura 65 exibe a projeção bidimensional gerada pelo t-SNE, utilizando perplexidade 12 e semente aleatória fixada em 0. Na visualização, observa-se a distribuição das imagens geradas das classes 1.33 (em azul, com 12 amostras) e 1.335 (em verde, também com 12 amostras), além desse dois grupo, temos as imagens transicionais, representadas em preto, totalizam 15 amostras da classe 1330 para 1335 e mais 16 amostras com sentido de 1335 a 1330.

Os vetores extremos do processo de transição estão destacados por círculos amarelos, sendo o primeiro pertencente à classe 1.33, com o nome *imagem3.jpg*, e o segundo à classe

1.335, com o nome *imagem2.jpg*, indicando os pontos de partida e de chegada entre as classes.

Figura 65 - Trajetória bidirecional no espaço latente (1.330 a 1.335, 1.335 a 1.330)



Complementando a análise da Figura 65, a Figura 66 apresenta uma representação visual das imagens geradas pela DGAN da classe 1.330, destacadas em azul no gráfico, com 12 amostras. A Figura 67 exibe uma representação semelhante da classe 1.335, com as imagens indicadas em verde, também com 12 amostras. Já a Figura 68 mostra a trajetória das imagens no espaço latente durante a interpolação no sentido 1.330 a 1.335, representada pela cor preta. Por fim, a Figura 69 apresenta a trajetória no sentido inverso, ou seja, de 1.335 a 1.330, igualmente representada pela cor preta.

Figura 66 - Imagens geradas da classe 1.330

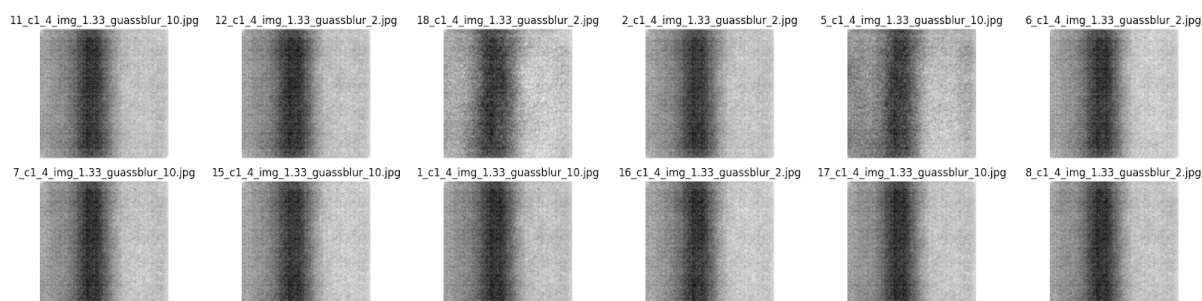


Figura 67 - Imagens geradas da classe 1.335

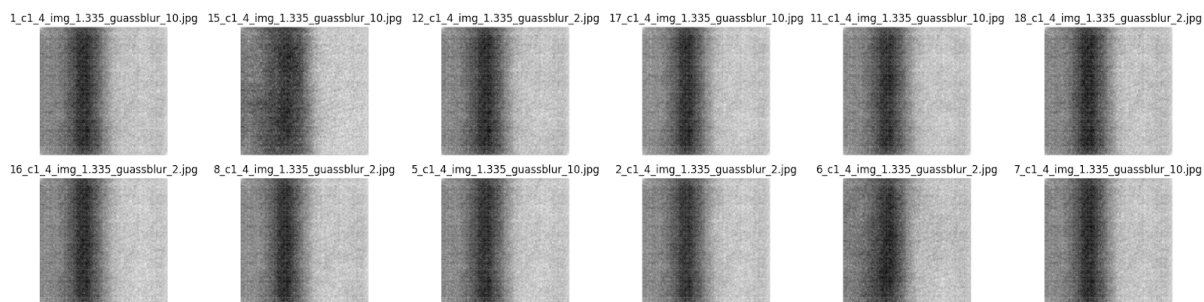


Figura 68 - Trajetória das imagens transicional entre as classe (1.330 a 1.335)

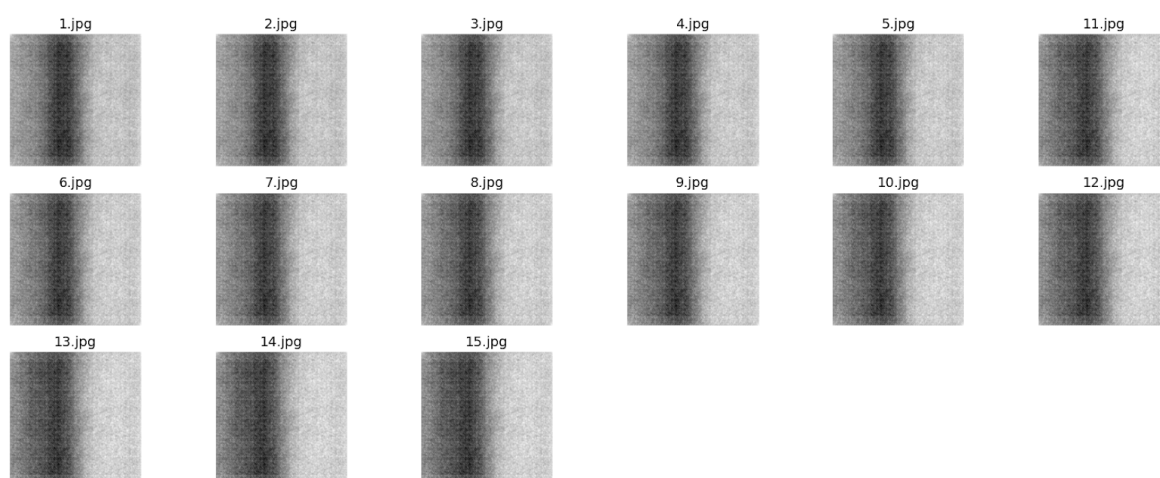
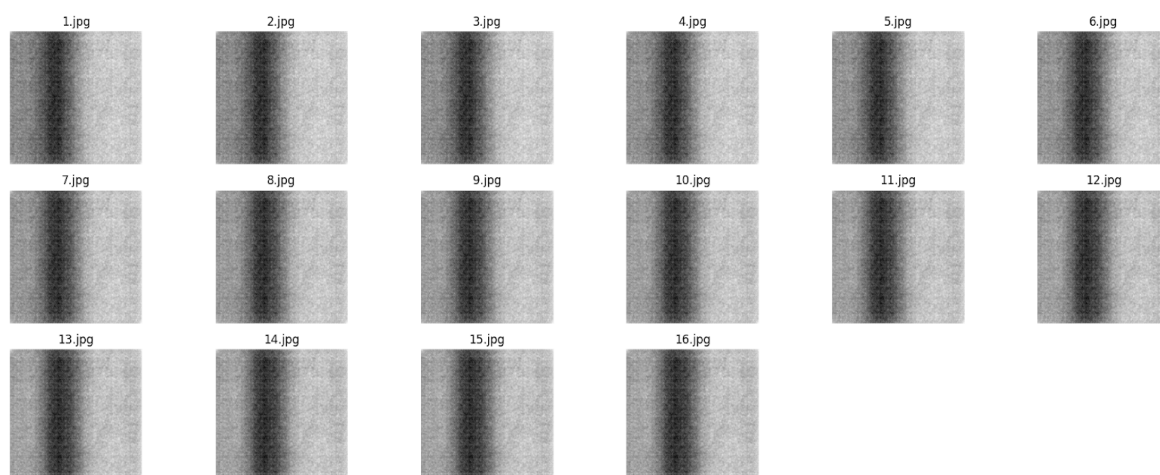


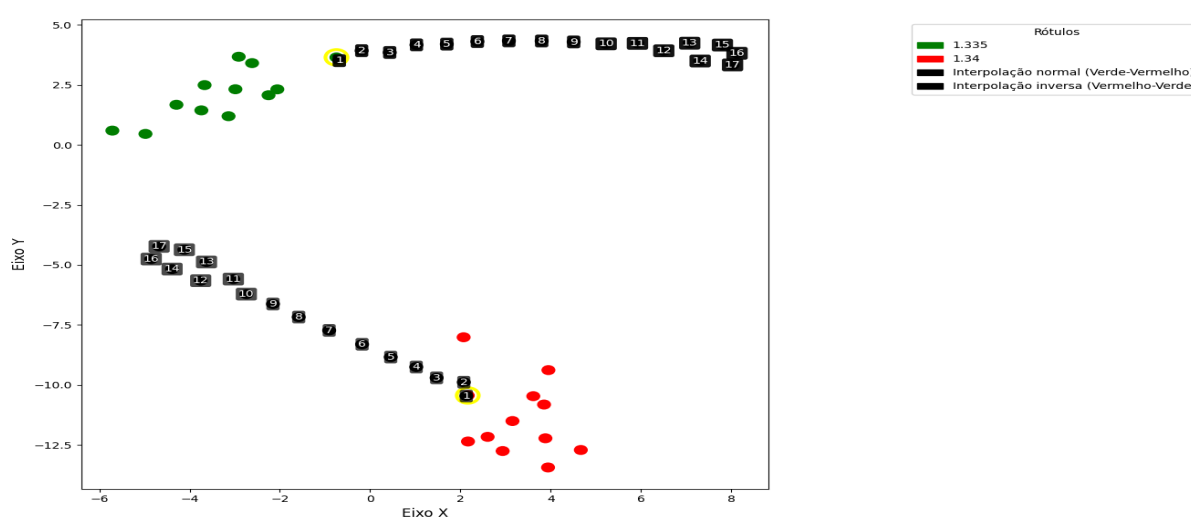
Figura 69 - Trajetória das imagens transicional entre as classe (1.335 a 1.330)



Outro exemplo com classes diferentes, é ilustrado na Figura 70, ela exibe a projeção bidimensional gerada pelo algoritmo t-SNE, utilizando perplexidade igual a 12 e semente

aleatória fixada em 0. Na visualização, observa-se a distribuição das imagens geradas das classes 1.335 (em verde, com 12 amostras) e 1.34 (em vermelha, também com 12 amostras), além desse dois grupo, temos as imagens transicionais, representadas em preto, totalizam 17 amostras da classe 1335 para 134 e mais 17 amostras com sentido oposto de 134 a 1335. Os vetores extremos do processo de transição estão destacados por círculos amarelos, sendo o primeiro pertencente à classe 1.335, com o nome *imagem5.jpg*, e o segundo à classe 1.34, com o nome *imagem4.jpg*, indicando os pontos de partida e de chegada entre as classes.

Figura 70 - Trajetória bidirecional no espaço latente (1.335 a 1.34, 1.34 a 1.335)



As Figuras 71 e 72 apresentam, respectivamente, 12 imagens geradas pela DGAN para as classes 1.335 (em verde) e 1.34 (em vermelho), complementando a análise iniciada na Figura 70. Já as Figuras 73 e 74 ilustram as trajetórias percorridas no espaço latente durante as transições entre essas classes: da 1.335 para a 1.34 e no sentido inverso, ambas destacadas em preto no gráfico.

Figura 71 - Imagens geradas da classe 1.335

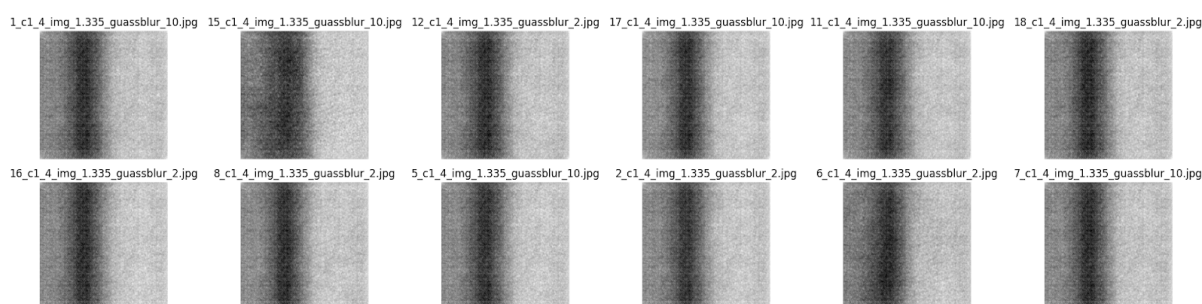


Figura 72 - imagens geradas da classe 1.34

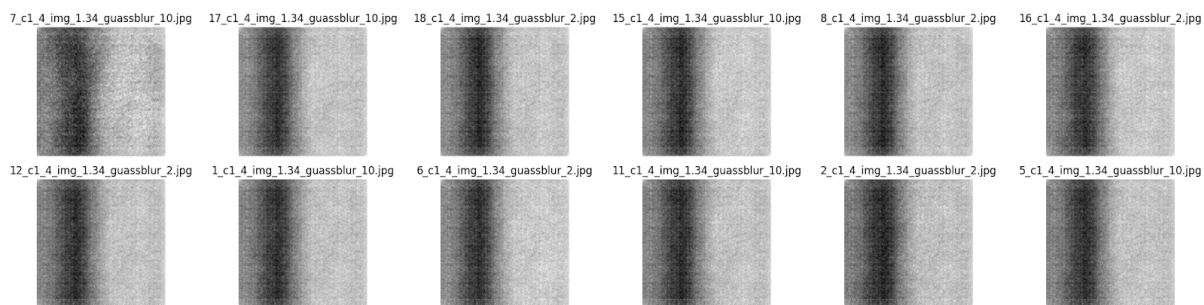


Figura 73 - Trajetória das imagens transicional entre as classe (1.335 a 1.34)

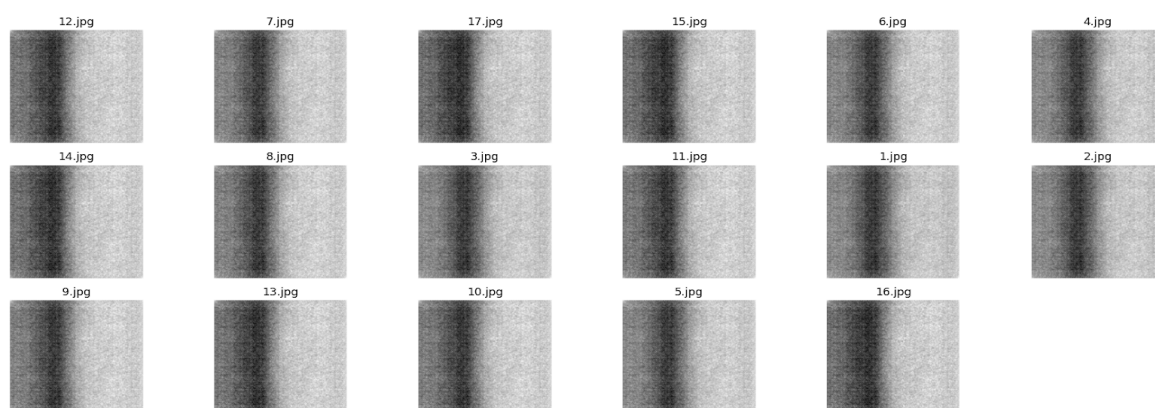
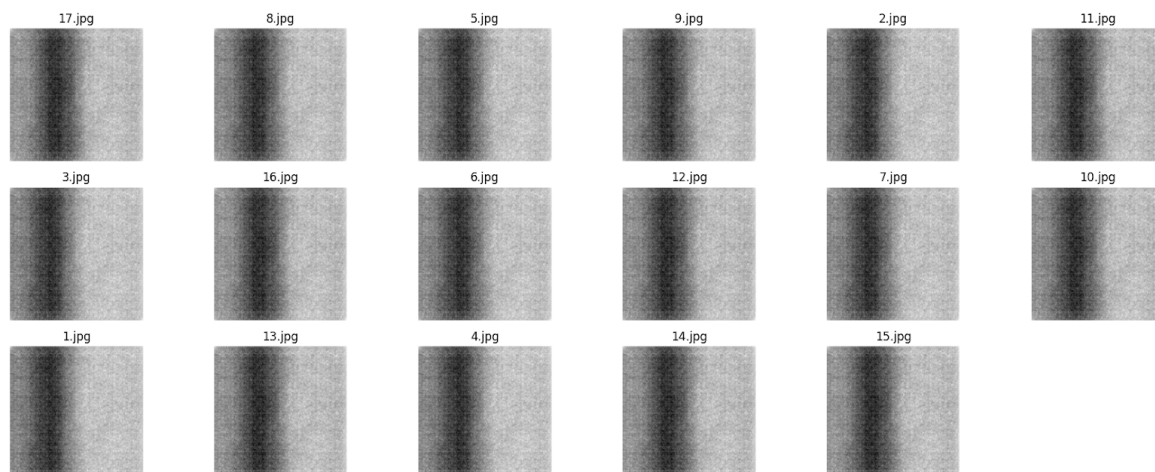


Figura 74 - Trajetória das imagens transicional entre as classe (1.34 a 1.335)



Os casos analisados demonstram que transições com baixa divergência semântica ou vetores latentes bem posicionados permitem a parada antecipada do treinamento, otimizando

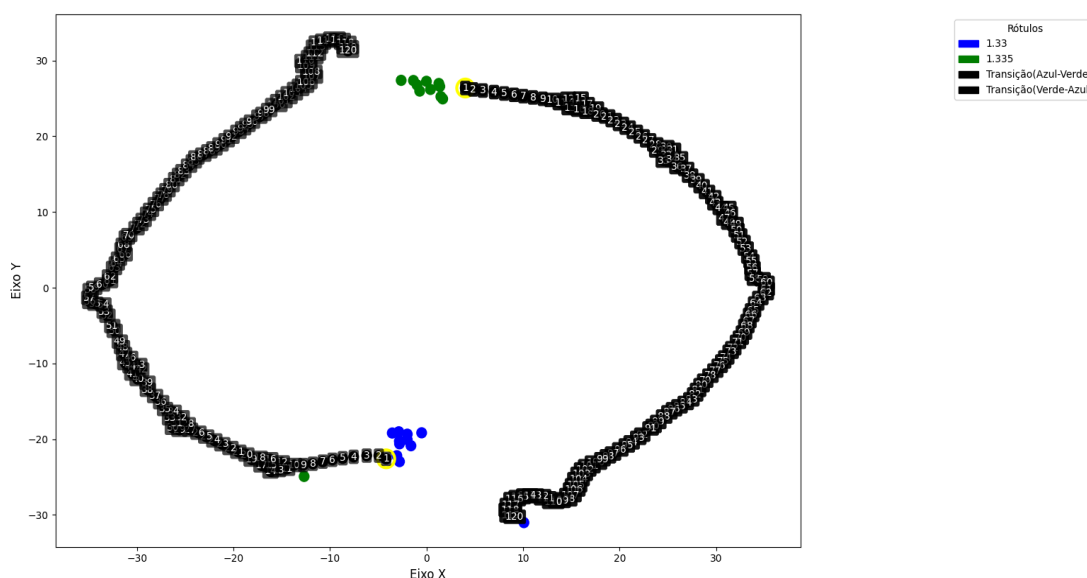
recursos computacionais. A visualização via t-SNE reforça a coerência espacial das transições, mesmo em cenários distintos.

4.2.3.2 Casos que Exigiram Todas as Épocas

Em contrapartida, houve situações em que o processo de transição precisou utilizar todas as 120 épocas disponíveis, sem que o critério de parada antecipada fosse acionado. Isso indica que a função de perda continuava a apresentar pequenas melhorias ao longo do tempo, embora sem estagnação suficiente para interromper o processo. Esses casos sugerem maior complexidade na otimização do vetor latente, podendo estar relacionados a uma distância inicial maior no espaço latente, maior variação semântica entre as imagens ou dificuldade do gerador em representar certos padrões. A Figura 75 apresenta um exemplo dessa situação, com uma convergência mais lenta e progressiva.

A Figura 75 exibe a projeção bidimensional gerada pelo t-SNE, utilizando perplexidade 12 e semente aleatória fixada em 0. Na visualização, observa-se a distribuição das imagens geradas das classes 1.33 (em azul, com 12 amostras) e 1.335 (em verde, também com 12 amostras) no espaço latente. As imagens transicionais, representadas em preto, totalizam 120 amostras da classe 1330 para 1335 e mais 120 amostras no sentido de 1335 a 1330.

Figura 75 - Trajetória bidirecional no espaço latente (1.330 a 1.335, 1.335 a 1.330)



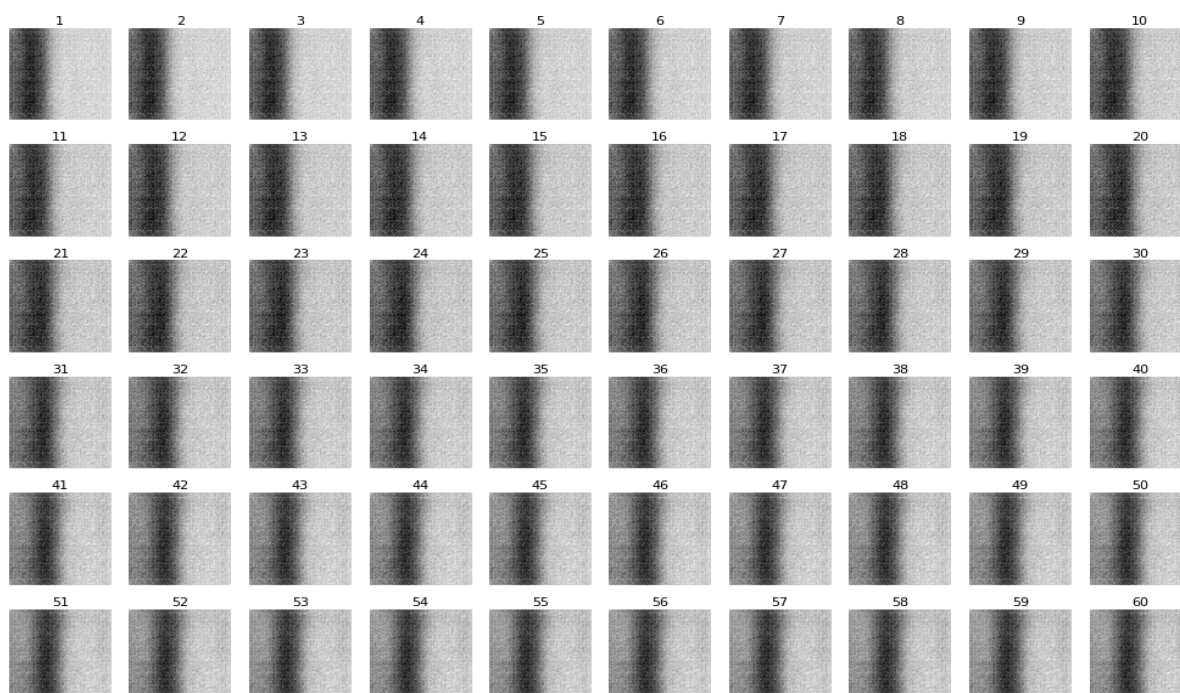
As imagens transicionais representadas em preto no gráfico, correspondem a 120 amostras no sentido da classe 1330 para 1335, e outras 120 no sentido inverso (1335 para 1330). As imagens correspondentes à Figura 76 estão disponíveis no repositório: disponível em: <https://github.com/GESyCA/SPR-GAN/tree/main/Figura73>. Acesso em: 5 ago. 2025.

O exemplo via t-SNE da Figura 76, demonstra que transições entre classes com alta divergência semântica exigem o uso completo das épocas de treinamento, refletindo a complexidade na otimização do espaço latente.

O *dataset* utilizado neste trabalho contempla as classes 1.330, 1.335, 1.34, 1.345 e 1.35. Para compreender melhor o comportamento do modelo ao longo do espaço latente, focamos em outro exemplo envolvendo imagens pertencentes às classes extremas. A transição se inicia em uma imagem, da classe 1.35, e segue até a uma imagem, da classe 1.330, isso assegura que o processo de geração de imagens vai passar por todas as classe, gerando 60 vetores latentes em sequência contínua em vez de 120 vetores, os quais são utilizados para produzir 60 imagens.

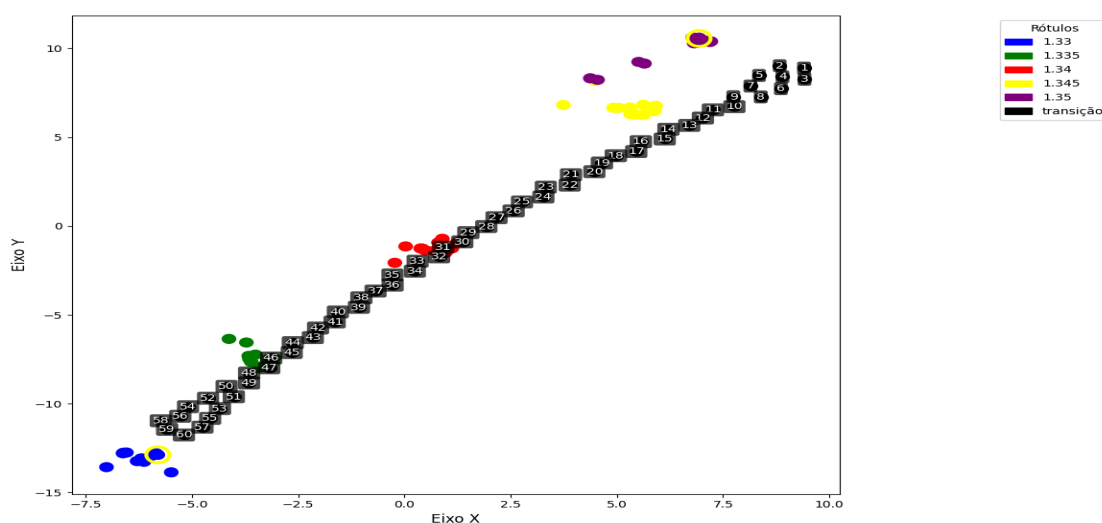
A Figura 76 apresenta as 60 imagens obtidas nesse percurso, permitindo observar visualmente a evolução das características morfológicas ao longo da transição entre as classes 1.35 e 1.330.

Figura 76 - Ilustra a transição entre classes



Complementando esta análise, a Figura 77 apresenta uma projeção bidimensional, via t-SNE, das imagens geradas pelas classes da DGAN. Cada classe é representada por 12 imagens, enquanto as imagens de transição (em preto, numeradas de 1 a 60) correspondem aos vetores em transição. Os extremos, estão destacados por círculos amarelos indicam o início e o fim do processo. A trajetória formada por esses pontos revela uma transição contínua. O t-SNE foi configurado com perplexidade igual a 13 e semente aleatória igual a 0, garantindo a reprodutibilidade da visualização. As imagens correspondentes à Figura 76 estão disponíveis em: <https://github.com/GESyCA/SPR-GAN/tree/main/Figura74>. Acesso em: 5 ago. 2025.

Figura 77 - Ilustra a plotagem de todas as classe sintética e a transição



Por fim, as 60 imagens geradas são compiladas em um vídeo, criando uma visualização dinâmica e contínua da transição entre as duas classes, disponível no link¹. Essa abordagem permite uma análise mais intuitiva da transformação no espaço latente, facilitando a compreensão das relações transicionais entre quaisquer classes presentes no *dataset*. Além disso, os resultados indicam que a técnica de transição se destacou entre as demais, pois inicia em um vetor latente específico e evolui de forma contínua até o destino, garantindo maior precisão na aproximação de imagens-alvo e evidenciando a coerência do espaço latente aprendido.

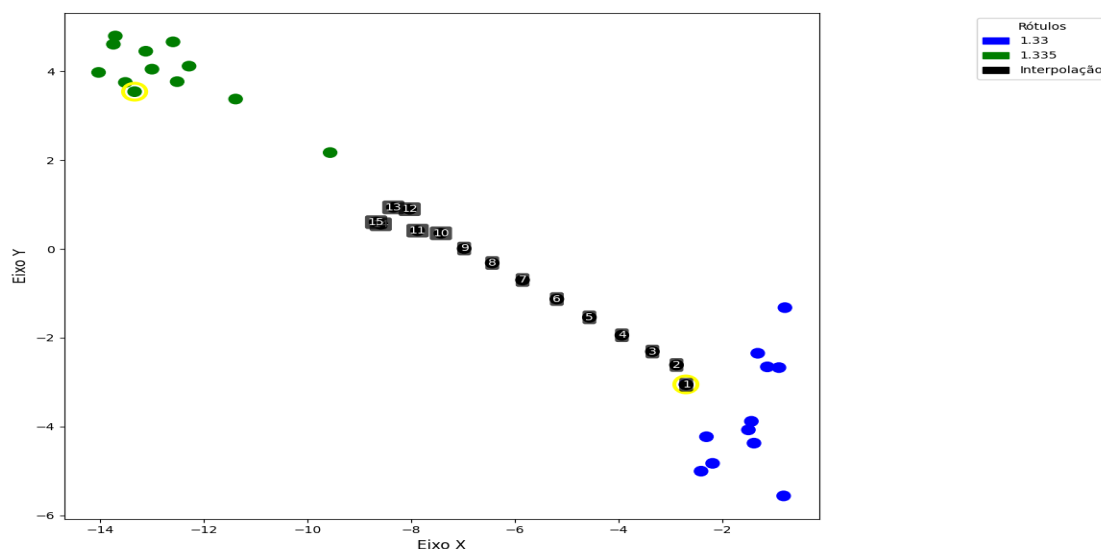
4.2.4 Classes Estimadas

Para exemplificar o processo de geração de novas classes intermediárias, foi desenvolvido uma técnica de transição vetorial entre as classes 1.330 e 1.335(as imagens das classes estão representadas nas Figuras 66 e 67. A partir dessas classes, foram extraídas imagens (*imagem1.jpg* da classe 1.330 e *imagem15.jpg* da classe 1.335) para servir como ponto inicial e final da transição. A partir dessas imagens, geraram-se 15 vetores intermediários, os quais deram origem a 15 novas imagens sintéticas.

¹ https://github.com/GESyCA/SPR-GAN/blob/main/video/transicao_latente.mp4. Acesso em: 5 ago. 2025.

Em seguida, aplicou-se o algoritmo t-SNE para projetar as imagens das classes 1.330, 1.335 e das 15 imagens intermediárias em um espaço bidimensional. A Figura 78 ilustra essa projeção: os pontos azuis representam as amostras da classe 1.330 e os verdes, da classe 1.335. Destacam-se, com círculo amarelo, as imagens selecionadas como referência de cada classe. Entre esses dois grupos, há uma sequência de 15 pontos pretos, numerados de 1 a 15.

Figura 78 - Espaço latente das imagens



A partir dessa projeção, foram extraídas as coordenadas (X , Y) de cada ponto referente às imagens interpoladas. Com essas informações, foi possível calcular a distância euclidiana total entre o ponto inicial (classe 1.330) e o ponto final (classe 1.335), resultando em um valor de 7,25.

Em seguida, foi realizada a subtração entre os valores das classes final e inicial, ou seja:

$$1.335 - 1.330 = 0,005.$$

Após isso, calculou-se a distância euclidiana individual (DEI) entre o ponto inicial (1.jpg) e cada um dos demais pontos (2.jpg, 3.jpg, 4.jpg, ..., 15.jpg). Cada uma dessas distâncias foi, então, dividida pela distância total (DET), resultando em uma proporção relativa da posição de cada ponto ao longo do caminho entre as duas classes.

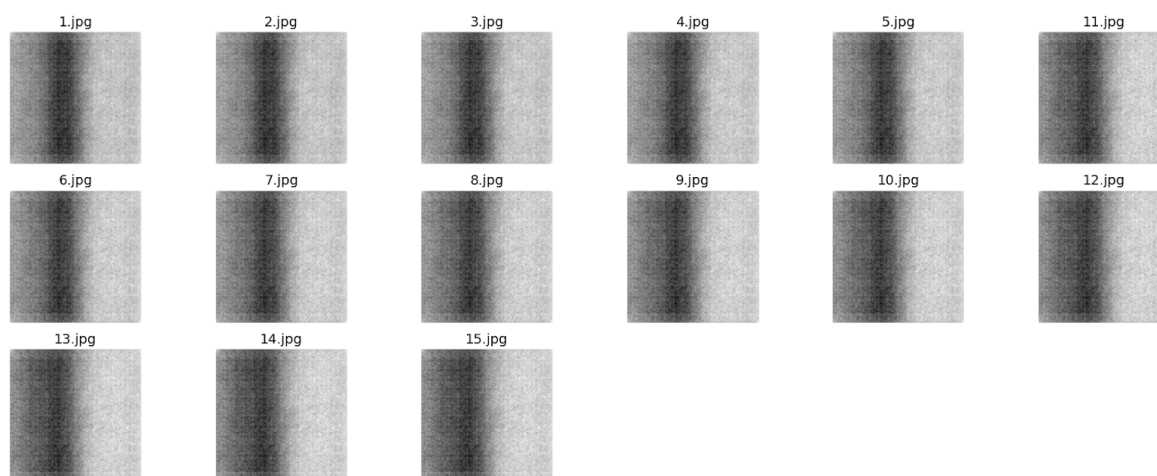
Por fim, estimou-se a classe estimada (CE) correspondente a cada ponto intermediário, utilizando a seguinte equação:

$$CE = 1.330 + (DEI / DET) \times 0,005$$

Dessa forma, imagens mais próximas do ponto inicial receberam valores como 1.331 ou 1.332, enquanto aquelas mais próximas do ponto final foram rotuladas como 1.334 ou 1.335.

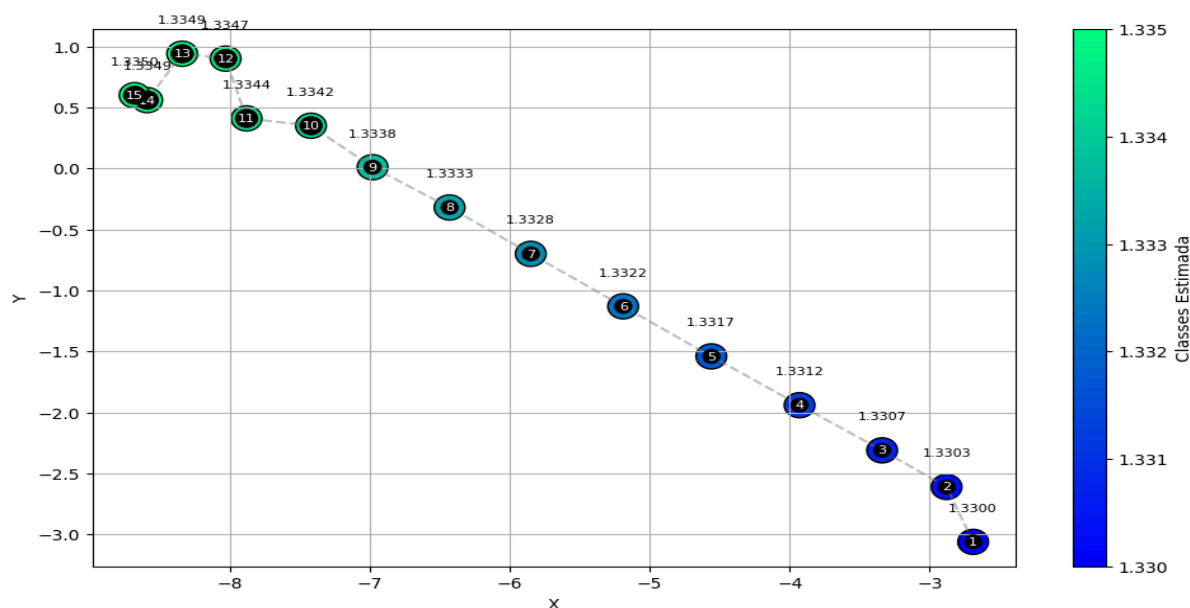
A Figura 79 ilustra as imagens da transição entre duas imagens representativas das classes 1.330 e 1.335, numeradas de (1.jpg a 15.jpg). Cada imagem corresponde a um ponto de transição ao longo do espaço latente entre as classes, evidenciando uma transição morfológica gradual e contínua. Observa-se uma mudança progressiva na região escura central das imagens, que se desloca e suaviza de forma coerente, mantendo, entretanto, o padrão geral de distribuição da intensidade luminosa.

Figura 79 – Imagens de transição entre as classes 1.330 e 1.335



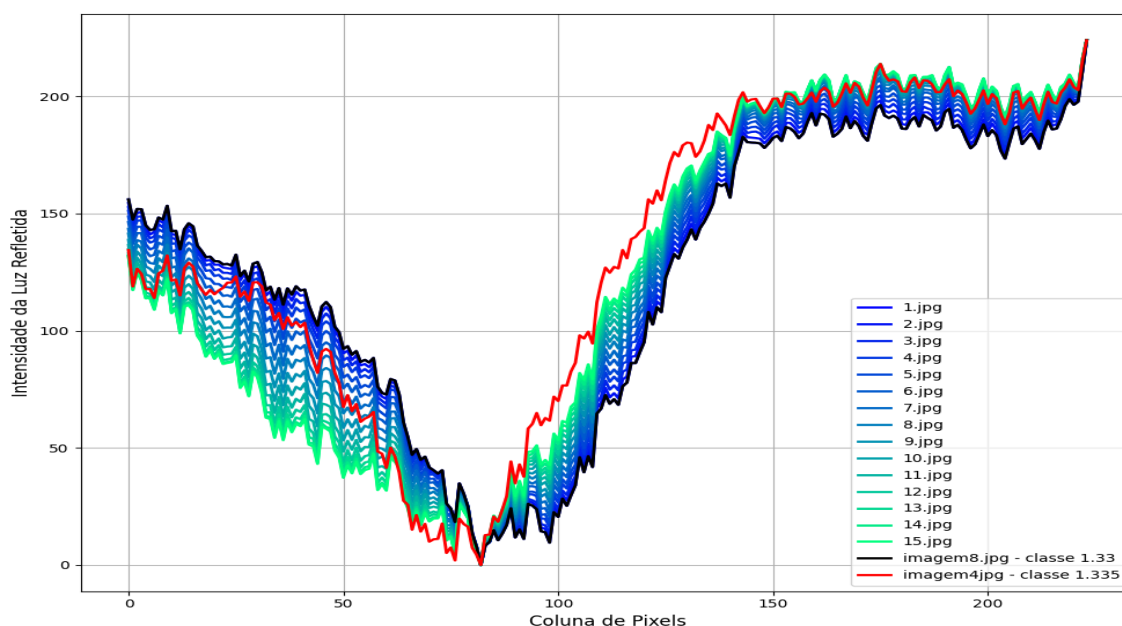
A Figura 80 apresenta o posicionamento de cada imagem no espaço latente. Cada ponto manteve sua posição original da projeção t-SNE, mas agora foi colorido de acordo com o valor da classe estimada. Essa visualização evidencia a *progressão suave* ao longo da trajetória e reforça a capacidade do método em estimar classes contínuas a partir da posição relativa das imagens.

Figura 80 - Classe estima no espaço latente (projeção t-SNE)



Para validar a consistência visual e morfológica entre as imagens geradas, a Figura 81 apresenta as curvas de intensidade da luz refletidas ao longo das colunas de pixels das imagens de transição. Cada curva, representando uma imagem de *1.jpg* a *15.jpg*, é exibida em gradiente de cores do azul ao verde. As imagens que marcam o início (*imagem8.jpg*) e o fim (*imagem4.jpg*) são destacadas em preto e vermelho, respectivamente. Apesar das leves variações, todas as curvas apresentam um padrão morfológico semelhante, com um vale central e bordas ascendentes, o que reforça a preservação das características fundamentais do sinal ao longo da transição.

Figura 81 - Curvas de intensidade da luz refletida nas imagens de transição (1.330 a 1.335)



Como etapa complementar, foi realizada a identificação precisa da posição do mínimo das curvas de intensidade, com o objetivo de quantificar pequenas variações ópticas. Para isso, aplicou-se um ajuste polinomial de segundo grau em uma janela de 50 pixels ao redor da região de mínimo. A Figura 81 ilustra o resultado desse processo, destacando os centróides obtidos (círculos verdes), os quais servem como medida sensível para variações nas condições experimentais.

5. CONCLUSÃO

O presente trabalho teve como objetivo principal investigar a capacidade de uma Rede Generativa Adversarial em gerar imagens sintéticas representativas das respostas ópticas de um sensor SPR, contemplando classes de índice de refração de 1.330 a 1.350. Para tanto, foram desenvolvidas e aplicadas diferentes estratégias de avaliação, como qualitativa, quantitativa e morfológica, a fim de garantir a fidelidade e a coerência dos dados gerados.

Durante o processo de treinamento, observou-se uma convergência estável entre o gerador e o discriminador, sem indícios de colapso de modo, o que indica um aprendizado bem-sucedido da rede. A avaliação visual das imagens mostrou que as amostras sintéticas

apresentam alta similaridade com as imagens reais, tanto na forma quanto na distribuição de intensidade, o que foi confirmado por transformadas de Fourier, curvas de intensidade e comparações paramétricas específicas como FWHM, largura, energia e ponto mínimo.

No aspecto estatístico, a métrica MMD evidenciou baixas discrepâncias entre distribuições de imagens reais e geradas, reforçando a qualidade do modelo. As análises com histogramas revelaram que as imagens geradas mantêm características globais e locais compatíveis com os dados originais.

A aplicação do t-SNE permitiu observar uma boa separação entre os grupos originais, com os clusters correspondentes às diferentes classes claramente definidas. As trajetórias de interpolação, tanto na direção normal quanto inversa, foram representadas de forma visualmente coerente, iniciando e terminando próximo ao clusters de origem e destino. Em todos os casos, a sequência numérica das interpolações apresentou continuidade e ordenação, sem sobreposições desordenadas, evidenciando que, mesmo com as distorções globais inerentes ao método, a coerência local foi preservada. Observa-se que, em algumas figuras 63, 64 e 69, as trajetórias não seguem exatamente a “linha mais curta” entre os clusters, o que é esperado, dado que o t-SNE não preserva distâncias globais, priorizando a manutenção das relações de vizinhança locais.

Adicionalmente, foi demonstrado que o espaço latente aprendido pelo modelo permite operações vetoriais com significados visuais consistentes. Técnicas como soma, subtração e interpolação linear e esférica, foram aplicadas com sucesso, resultando em imagens intermediárias com transições suaves e morfologicamente plausíveis. A abordagem de transição vetorial permitiu ainda a geração de classes intermediárias, por exemplo, entre 1.330 e 1.335, com imagens visual e estatisticamente coerentes.

Em síntese, os resultados confirmam a robustez do modelo, tanto na reprodução de classes conhecidas quanto na geração de novas classes sintéticas. A DCGAN mostrou-se eficaz na simulação e expansão de bases de dados aplicadas a sensores SPR, abrindo possibilidades para aplicações em contextos onde a obtenção de dados reais é limitada. No entanto, observou-se uma leve suavização das imagens geradas, com perda de detalhes de alta frequência em relação às imagens reais, o que indica uma limitação ao modelo utilizado. Essa limitação aponta para a necessidade de investigações futuras com arquiteturas mais

sofisticadas, capazes de preservar com maior precisão as texturas finas das imagens do sensor SPR.

5.1 Trabalhos Futuros

Apesar dos resultados positivos alcançados neste trabalho, há diversas oportunidades para avanços na geração de imagens sintéticas de sensores SPR e em aplicações correlatas. A seguir, são apresentadas direções estratégicas para pesquisas futuras, fundamentadas nas limitações identificadas ao longo deste estudo.

Em primeiro lugar, sugere-se a ampliação do intervalo de índices de refração utilizado na base de dados, com foco na criação de amostras para valores intermediários mais densos. Por exemplo, entre as classes 1.340 e 1.345 ou entre 1.345 e 1.350, pode-se gerar novas imagens em incrementos menores, como 1.341, 1.342 etc. Essa densificação permitirá avaliar com maior precisão a capacidade do modelo de generalizar para classes não vistas e gerar respostas contínuas e coerentes.

Outro ponto relevante é a exploração do espaço latente do modelo para isolar atributos físicos específicos, por exemplo, a largura da curva SPR ou a posição do mínimo de ressonância, viabilizando um controle mais preciso durante a geração de imagens. Essa linha de pesquisa pode ainda incorporar a aplicabilidade prática das operações sobre os vetores latentes, ampliando as possibilidades de manipulação interpretável e controlada.

Adicionalmente, sugere-se investigar arquiteturas consolidadas de extração e operação sobre estilos, bem como avaliar o desempenho do modelo utilizando métricas quantitativas de realismo e diversidade (como FID e IS), de modo a identificar eventuais indícios de colapso de modo em determinadas classes.

No campo das arquiteturas, sugere-se aprofundar as análises já realizadas com Autoencoders Variacionais (VAEs) e modelos de Difusão, investigando sua aplicação em bases mais densas e com métricas quantitativas de realismo e diversidade. Além disso, recomenda-se explorar novas abordagens, como Transformers, a fim de verificar se essas técnicas oferecem maior robustez e capacidade de síntese.

Por fim, destaca-se a integração dos dados sintéticos em redes neurais artificiais voltadas para tarefas de classificação, como as arquiteturas Xception, InceptionResNetV2 ou EfficientNet. Essa aplicação permitirá validar a utilidade prática das imagens geradas, verificando se elas contribuem efetivamente para a melhoria do desempenho em cenários supervisionados.

Essas direções visam não apenas avançar o estado da arte na geração de imagens sintéticas para SPR, mas também garantir que futuras implementações sejam robustas, replicáveis e alinhadas com as demandas reais da comunidade científica.

REFERÊNCIA

ABDAL, R.; QIN, Y.; WONKA, P. **Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?**. [S. l.]: arXiv, 2019a. Disponível em: <http://arxiv.org/abs/1904.03189>. Acesso em: 29 jul. 2025.

SAAD, M. M.; REHMANI, M. H.; O'REILLY, R. Early Stopping Criteria for Training Generative Adversarial Networks in Biomedical Imaging. arXiv preprint arXiv:2405.20987, 2024.

ABDAL, R.; QIN, Y.; WONKA, P. **Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?**. [S. l.]: arXiv, 2019b. Disponível em: <http://arxiv.org/abs/1904.03189>. Acesso em: 29 jul. 2025.

AGGARWAL, C. C. **Data Mining: The Textbook**. Cham: Springer International Publishing, 2015. Disponível em: <https://link.springer.com/10.1007/978-3-319-14142-8>. Acesso em: 29 jul. 2025.

ARJOVSKY, M.; CHINTALA, S.; BOTTOU, L. Wasserstein Generative Adversarial Networks. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2017. **Proceedings of the 34th International Conference on Machine Learning**. [S. l.]: PMLR, 2017. p. 214–223. Disponível em: <https://proceedings.mlr.press/v70/arjovsky17a.html>. Acesso em: 28 jul. 2025.

BARRETTO, J.; ATTUX, R. Fundamentals of Deep Neural Networks - application based on MLPS. **Revista dos Trabalhos de Iniciação Científica da UNICAMP**, [s. l.], n. 27, p. 1–1, 2019.

BATISTA, J. C. S. **Filtro de dados com redes neurais multinível para remoção de ruídos na resposta de sensores baseados na ressonância Deplasmon de superfície**. 2020. Mestrado - UFERSA, Mossoró, 2020. Disponível em: <https://repositorio.ufersa.edu.br/handle/prefix/7049>. Acesso em: 8 ago. 2025.

BENGIO, Y. *et al.* **Better Mixing via Deep Representations**. [S. l.]: arXiv, 2012. Disponível em: <http://arxiv.org/abs/1207.4404>. Acesso em: 29 jul. 2025.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. **Representation Learning: A Review and New Perspectives**. [S. l.]: arXiv, 2014. Disponível em: <http://arxiv.org/abs/1206.5538>. Acesso em: 6 ago. 2025.

BISHOP, C. M. **Pattern recognition and machine learning**. New York: Springer, 2006. (Information science and statistics).

BORN & WOLF. **BW.pdf**. [S. l.], 1999. Disponível em: <https://www.phys.uni-sofia.bg/~svetivanov/seminars/uchebnici/optics/BW.pdf>. Acesso em: 6 ago. 2025.

DHARIWAL, P.; NICHOL, A. **Diffusion Models Beat GANs on Image Synthesis**. [S. l.]: arXiv, 2021. Disponível em: <http://arxiv.org/abs/2105.05233>. Acesso em: 29 jul. 2025.

FERREIRA, W. D. **Detecção de imagens falsificadas baseada em descritores locais de textura e rede neural convolucional**. [S. l.], 2020.

GOMES, J. C. M.; SOUZA, L. C.; OLIVEIRA, L. C. SmartSPR sensor: Machine learning approaches to create intelligent surface plasmon based sensors. **Biosensors and Bioelectronics**, [s. l.], v. 172, p. 112760, 2021.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing, Global Edition**. [S. l.]: Pearson Education, 2018.

GOODFELLOW, I. J. *et al.* **Generative Adversarial Networks**. [S. l.]: arXiv, 2014. Disponível em: <http://arxiv.org/abs/1406.2661>. Acesso em: 28 jul. 2025.

GOODFELLOW ET AL., J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. **Genetic Programming and Evolvable Machines**, [s. l.], v. 19, n. 1, p. 305–307, 2018.

GRETTON, A. *et al.* A kernel two-sample test. **J. Mach. Learn. Res.**, [s. l.], v. 13, n. null, p. 723–773, 2012.

GULRAJANI, I. *et al.* **Improved Training of Wasserstein GANs**. [S. l.]: arXiv, 2017. Disponível em: <http://arxiv.org/abs/1704.00028>. Acesso em: 6 ago. 2025.

HASTIE, T.; FRIEDMAN, J.; TIBSHIRANI, R. **The Elements of Statistical Learning**. New York, NY: Springer, 2001. (Springer Series in Statistics). Disponível em: <http://link.springer.com/10.1007/978-0-387-21606-5>. Acesso em: 29 jul. 2025.

HAYKIN, S. Neural networks expand SP's horizons. **IEEE Signal Processing Magazine**, [s. l.], v. 13, n. 2, p. 24–49, 1996.

HO, J.; JAIN, A.; ABBEEL, P. Denoising Diffusion Probabilistic Models. *In:* , 2020. **Advances in Neural Information Processing Systems**. [S. l.]: Curran Associates, Inc., 2020. p. 6840–6851. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html. Acesso em: 28 jul. 2025.

HOMOLA, J. Surface plasmon resonance sensors for detection of chemical and biological species. **Chemical Reviews**, [s. l.], v. 108, n. 2, p. 462–493, 2008.

JIANG, J. *et al.* Freeform Diffractive Metagrating Design Based on Generative Adversarial Networks. **ACS Nano**, [s. l.], v. 13, n. 8, p. 8872–8878, 2019.

KARRAS, T.; LAINE, S.; AILA, T. **A Style-Based Generator Architecture for Generative Adversarial Networks**. [S. l.]: arXiv, 2019. Disponível em: <http://arxiv.org/abs/1812.04948>. Acesso em: 6 ago. 2025.

KINGMA, D. P.; BA, J. **Adam: A Method for Stochastic Optimization**. [S. l.]: arXiv, 2017. Disponível em: <http://arxiv.org/abs/1412.6980>. Acesso em: 29 jul. 2025.

KINGMA, D. P.; WELLING, M. **Auto-Encoding Variational Bayes**. [S. l.]: arXiv, 2022. Disponível em: <http://arxiv.org/abs/1312.6114>. Acesso em: 29 jul. 2025.

KOBAK, D.; BERENS, P. The art of using t-SNE for single-cell transcriptomics. **Nature Communications**, [s. l.], v. 10, n. 1, p. 5416, 2019.

KRETSCHMANN, E.; RAETHER, H. Notizen: Radiative Decay of Non Radiative Surface Plasmons Excited by Light. **Zeitschrift für Naturforschung A**, [s. l.], v. 23, n. 12, p. 2135–2136, 1968.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: , 2012. **Advances in Neural Information Processing Systems**. [S. l.]: Curran Associates, Inc., 2012. Disponível em: <https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html>. Acesso em: 6 ago. 2025.

KULLBACK, S.; LEIBLER, R. A. On Information and Sufficiency. **The Annals of Mathematical Statistics**, [s. l.], v. 22, n. 1, p. 79–86, 1951.

LAI, M. **Giraffe: Using Deep Reinforcement Learning to Play Chess**. [S. l.]: arXiv, 2015. Disponível em: <http://arxiv.org/abs/1509.01549>. Acesso em: 6 ago. 2025.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, [s. l.], v. 521, n. 7553, p. 436–444, 2015.

LI, J.; SONG, G.; ZHANG, M. Occluded offline handwritten Chinese character recognition using deep convolutional generative adversarial network and improved GoogLeNet. **Neural Computing and Applications**, [s. l.], v. 32, n. 9, p. 4805–4819, 2020.

MA, W. *et al.* Deep learning for the design of photonic structures. **Nature Photonics**, [s. l.], v. 15, n. 2, p. 77–90, 2021.

MALINICK, A. S. *et al.* Surface plasmon resonance imaging (SPRi) in combination with machine learning for microarray analysis of multiple sclerosis biomarkers in whole serum. **Biosensors and Bioelectronics**: X, v. 10, p. 100127, 2022. Disponível em: <https://doi.org/10.1016/j.biosx.2022.100127>. Acesso em: 6 ago. 2025.

MCCARTHY, J. *et al.* **A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence**. [S. l.]: s. n., 1955. Disponível em: <http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html>.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, [s. l.], v. 5, n. 4, p. 115–133, 1943.

MIRZA, M.; OSINDERO, S. **Conditional Generative Adversarial Nets**. [S. l.]: arXiv, 2014. Disponível em: <http://arxiv.org/abs/1411.1784>. Acesso em: 6 ago. 2025.

MONARD, M. C.; BARANAUSKAS, J. A. Aplicações de inteligência artificial: uma visão geral. **Anais**, [s. l.], 2000. Disponível em: <https://repositorio.usp.br/item/001106450>. Acesso em: 6 ago. 2025.

MOREIRA, C. da S. **Projeto e realização de um biochip óptico para aplicações biológicas baseado no princípio de ressonância de plásmons de superfície**. [S. l.], 2010.

OLIVEIRA, L. C. *et al.* Intelligent Surface Plasmon Resonance Sensor for Refractive Index Substance Identification With Convolutional Neural Network Image-Based Model. **IEEE Transactions on Instrumentation and Measurement**, [s. l.], v. 74, p. 1–10, 2025.

OLIVEIRA, L. C. *et al.* **Surface Plasmon Resonance Sensors: A Materials Guide to Design, Characterization, Optimization, and Usage**. Cham: Springer International Publishing, 2019. (Springer Series in Surface Sciences). v. 70 Disponível em: <http://link.springer.com/10.1007/978-3-030-17486-6>. Acesso em: 2 abr. 2024.

OTTO, A. Excitation of nonradiative surface plasma waves in silver by the method of frustrated total reflection. **Zeitschrift für Physik A Hadrons and nuclei**, [s. l.], v. 216, n. 4, p. 398–410, 1968.

PASZKE, A. *et al.* **PyTorch: An Imperative Style, High-Performance Deep Learning Library**. [S. l.]: arXiv, 2019. Disponível em: <http://arxiv.org/abs/1912.01703>. Acesso em: 29 jul. 2025.

PEARSON, K. Contributions to the Mathematical Theory of Evolution. **Philosophical Transactions of the Royal Society of London. A**, [s. l.], v. 185, p. 71–110, 1894.

PINGI, S. T.; NAYAK, R.; BASHAR, M. A. Conditional Generative Adversarial Network for Early Classification of Longitudinal Datasets Using an Imputation Approach. **ACM Trans. Knowl. Discov. Data**, [s. l.], v. 18, n. 5, p. 132:1-132:25, 2024.

PRADHYUMNA, P.; MOHANA. A Survey of Modern Deep Learning based Generative Adversarial Networks (GANs). *In*: 2022 6TH INTERNATIONAL CONFERENCE ON COMPUTING METHODOLOGIES AND COMMUNICATION (ICCMC), 2022. **2022 6th International Conference on Computing Methodologies and Communication (ICCMC)**. [S. l.: s. n.], 2022. p. 1146–1152. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9753782>. Acesso em: 6 ago. 2025.

RADFORD, A.; METZ, L.; CHINTALA, S. **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks**. [S. l.]: arXiv, 2016. Disponível em: <http://arxiv.org/abs/1511.06434>. Acesso em: 28 jul. 2025.

RAUBER, T. **Redes_neurais_artificiais.pdf**. [S. l.: s. n.], 2005.

RITCHIE, R. H. Plasma Losses by Fast Electrons in Thin Films. **Physical Review**, [s. l.], v. 106, n. 5, p. 874–881, 1957.

SHI, X. *et al.* Metasurface inverse design using machine learning approaches. **Journal of Physics D Applied Physics**, [s. l.], v. 53, p. 275105, 2020.

SILVER, D. *et al.* **Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm**. [S. l.]: arXiv, 2017. Disponível em: <http://arxiv.org/abs/1712.01815>. Acesso em: 6 ago. 2025.

SIMONYAN, K.; ZISSERMAN, A. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. [S. l.]: arXiv, 2015. Disponível em: <http://arxiv.org/abs/1409.1556>. Acesso em: 29 jul. 2025.

SOHL-DICKSTEIN, J. *et al.* **Deep Unsupervised Learning using Nonequilibrium Thermodynamics**. [S. l.]: arXiv, 2015. Disponível em: <http://arxiv.org/abs/1503.03585>. Acesso em: 29 jul. 2025.

SOILLE, P. **Morphological Image Analysis**. Berlin, Heidelberg: Springer, 2004. Disponível em: <http://link.springer.com/10.1007/978-3-662-05088-0>. Acesso em: 29 jul. 2025.

SUTTON, R. S.; BARTO, A. **Reinforcement learning: an introduction**. Nachdrucked. Cambridge, Massachusetts: The MIT Press, 2018. (Adaptive computation and machine learning).

TAFNER. **O Que São as Redes Neurais Artificiais**. [S. l.], 1998. Disponível em: <https://cerebromente.org.br/n05/tecnologia/rna.htm>. Acesso em: 11 maio 2024.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. Introduction to Data Mining. [s. l.],

TAN, Pang-Ning; STEINBACH, Michael; KUMAR, Vipin. Introduction to Data Mining. Boston: Pearson Addison Wesley, 2006.

THADSON, K.; VISITSATTAPONGSE, S.; PECHPRASARN, S. Deep learning-based single-shot phase retrieval algorithm for surface plasmon resonance microscope based refractive index sensing application. **Scientific Reports**, [s. l.], v. 11, n. 1, p. 16289, 2021.

TIWARI, P. B. *et al.* SPRD: a surface plasmon resonance database of common factors for better experimental planning. **BMC Molecular and Cell Biology**, [s. l.], v. 22, n. 1, p. 17, 2021.

TOM M. MITCHELL. **Machine Learning Tom Mitchell**. [S. l.: s. n.], 1997. Disponível em: http://archive.org/details/machine-learning-tom-mitchell_202402. Acesso em: 6 ago. 2025.

TURING, A. M. Computing Machinery and Intelligence. In: EPSTEIN, R.; ROBERTS, G.; BEBER, G. (org.). **Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer**. Dordrecht: Springer Netherlands, 1950. p. 23–65. Disponível em: https://doi.org/10.1007/978-1-4020-6710-5_3. Acesso em: 28 jul. 2025.

VAN DER MAATEN, Laurens; HINTON, Geoffrey. **Visualizing data using t-SNE**. **Journal of Machine Learning Research**, v. 9, p. 2579-2605, 2008. Disponível em: <https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf>. Acesso em: 29 jul. 2025.

WEN, J. *et al.* **Diamond in the rough: Improving image realism by traversing the GAN latent space.** [S. l.]: arXiv, 2021. Disponível em: <http://arxiv.org/abs/2104.05518>. Acesso em: 29 jul. 2025.

XIA, W. *et al.* GAN Inversion: A Survey. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [s. l.], p. 1–17, 2022.

XU, Q. *et al.* **An empirical study on evaluation metrics of generative adversarial networks.** [S. l.]: arXiv, 2018. Disponível em: <http://arxiv.org/abs/1806.07755>. Acesso em: 29 jul. 2025.

YAMAN, M. Y. *et al.* **Learning and predicting photonic responses of plasmonic nanoparticle assemblies via dual variational autoencoders.** [S. l.]: arXiv, 2022. Disponível em: <http://arxiv.org/abs/2208.03861>. Acesso em: 8 ago. 2025.

ZHAO, S.; SONG, J.; ERMON, S. **Towards Deeper Understanding of Variational Autoencoding Models.** [S. l.]: arXiv, 2017. Disponível em: <http://arxiv.org/abs/1702.08658>. Acesso em: 28 jul. 2025.

ZHU, J.-Y. *et al.* **Generative Visual Manipulation on the Natural Image Manifold.** [S. l.]: arXiv, 2018. Disponível em: <http://arxiv.org/abs/1609.03552>. Acesso em: 29 jul. 2025.