UNIVERSIDADE FEDERAL DE SÃO CARLOS– UFSCAR
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA– CCET
DEPARTAMENTO DE COMPUTAÇÃO– DC
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO– PPGCC

BRUNO ANTHONY SHIMURA

APRENDIZADO AUTOSSUPERVISIONADO CONTRASTIVO ORIENTADO PELA ESTRUTURA GEOMÉTRICA DO ESPAÇO LATENTE

São Carlos
2025

Bruno Anthony Shimura

Aprendizado Autossupervisionado Contrastivo Orientado pela Estrutura Geométrica do Espaço Latente

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Metodologias e Técnicas de Computação

Orientador: Prof. Dr. Pedro Henrique Bugatti

São Carlos
2025

Shimura, Bruno Anthony

Aprendizado Autossupervisionado Contrastivo Orientado pela Estrutura Geométrica do Espaço Latente / Bruno Anthony Shimura -- 2025.
118f.

Dissertação (Mestrado) - Universidade Federal de São Carlos, campus São Carlos, São Carlos
Orientador (a): Pedro Henrique Bugatti
Banca Examinadora: Pedro Henrique Bugatti, Renato Bueno, Claiton de Oliveira
Bibliografia

1. Aprendizado autossupervisionado. 2. Aprendizado contrastivo. 3. Visão computacional. I. Shimura, Bruno Anthony. II. Título.

Ficha catalográfica desenvolvida pela Secretaria Geral de Informática (SIn)

DADOS FORNECIDOS PELO AUTOR

Bibliotecário responsável: Arildo Martins - CRB/8 7180

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Bruno Anthony Shimura, realizada em 31/10/2025.

Comissão Julgadora:

Prof. Dr. Pedro Henrique Bugatti (UFSCar)

Prof. Dr. Renato Bueno (UFSCar)

Prof. Dr. Claiton de Oliveira (UTFPR)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

Agradecimentos

Gostaria de agradecer, primeiramente, a Deus pela oportunidade de realizar esse projeto. Agradeço aos meus pais, irmãos e toda a minha família por todo o suporte, amor e confiança ao decorrer do programa. Meus sinceros agradecimentos ao meu orientador, Prof. Dr. Pedro Henrique Bugatti, que, apesar dos obstáculos no decorrer do ano, me deu todo o suporte necessário. Gostaria também de agradecer à UFSCar pela estrutura de ensino oferecida e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo financiamento e confiança em minha pesquisa.

“O homem sábio é forte,
e o homem de conhecimento consolida a força.”
(Provérbios 24:5 - ACF)

Resumo

O aprendizado autossupervisionado (Self-Supervised Learning — SSL) tem se consolidado como um paradigma promissor no campo da visão computacional, por permitir o aprendizado de representações visuais robustas sem a necessidade de grandes volumes de dados rotulados. Entre suas vertentes, o aprendizado contrastivo destaca-se por induzir a formação de representações discriminativas por meio da aproximação de pares positivos e distanciamento de pares negativos. Entretanto, a seleção aleatória desses pares limita a exploração eficiente da estrutura geométrica subjacente aos dados, resultando em representações com separabilidade subótima entre classes. Com o objetivo de mitigar essa limitação, este trabalho propõe o Aprendizado Contrastivo Guiado por Distância (Distance-Guided Contrastive Learning — DGCL), uma abordagem autossupervisionada que introduz um mecanismo de seleção de pares baseado na estrutura geométrica do espaço latente. A proposta consiste em identificar, para cada amostra âncora, os exemplos mais informativos, as amostras positivas mais distantes dentro da mesma classe (hard positives) e as amostras negativas mais próximas de classes diferentes (hard negatives), utilizando projeções obtidas via t-Distributed Stochastic Neighbor Embedding (t-SNE). Essa política de seleção orientada pela distância é aplicada iterativamente, refinando progressivamente o espaço de representações. Os experimentos realizados nos conjuntos de imagens CIFAR-10, FER-13, KDEF e RAF-DB demonstram ganhos expressivos de desempenho em relação a modelos sem aprendizado contrastivo. Observa-se, em especial, que o DGCL promove uma compactação intra-classe mais pronunciada e uma separação inter-classe mais consistente, resultando em representações latentes mais coerentes e discriminativas. As análises qualitativas baseadas em projeções t-SNE e as matrizes de confusão confirmam a capacidade do método em estruturar o espaço latente de maneira semanticamente significativa. O código-fonte do presente trabalho está disponível em: https://github.com/brunoshimura/dgcl

Palavras-chave: Aprendizado autossupervisionado, aprendizado contrastivo, seleção de pares, aprendizado por distância, espaço latente, visão computacional.

Abstract

Self-Supervised Learning (SSL) has emerged as a powerful paradigm in computer vision, enabling models to learn meaningful feature representations directly from unlabeled data. Among SSL approaches, contrastive learning has gained particular prominence for its ability to induce discriminative embeddings by pulling together positive pairs and pushing apart negatives. However, random sampling of such pairs often disregards the underlying geometric structure of the latent space, leading to suboptimal representation quality and inconsistent class separation. To address this limitation, this work introduces Distance-Guided Contrastive Learning (DGCL), a self-supervised approach that systematically selects informative sample pairs based on their geometric configuration in the latent manifold. For each anchor sample, DGCL identifies the farthest intra-class examples (hard positives) and the nearest inter-class examples (hard negatives) through t-Distributed Stochastic Neighbor Embedding (t-SNE) projections. By iteratively refining these relationships across training cycles, the method progressively enhances intra-class compactness and inter-class separability. Experiments conducted on the CIFAR-10, FER-13, KDEF, and RAF-DB datasets demonstrate substantial improvements over conventional models trained without contrastive learning. The results reveal that DGCL yields geometrically consistent latent representations, characterized by reduced intra-class variance and well-structured semantic clusters. The source codes of the proposed approach is publicly available at https://github.com/brunoshimura/dgcl.

Keywords: Self-supervised learning, contrastive learning, pair selection, distance-guided representation learning, latent space, computer vision.

Lista de ilustrações

Lista de tabelas

Lista de siglas

BYOL Bootstrap Your Own Latent

CNN Convolutional Neural Network

DGCL Distance-Guided Contrastive Learning

JSON JavaScript Object Notation

MoCo Momentum Contrast

PCA Principal Component Analysis

ResNet Residual Networks

SwAV Swapping Assignments Between Views

SimCLR Simple Contrastive Learning of Representations

SSL Self-Supervised Learning

t-SNE t-Distributed Stochastic Neighbor Embedding

Sumário

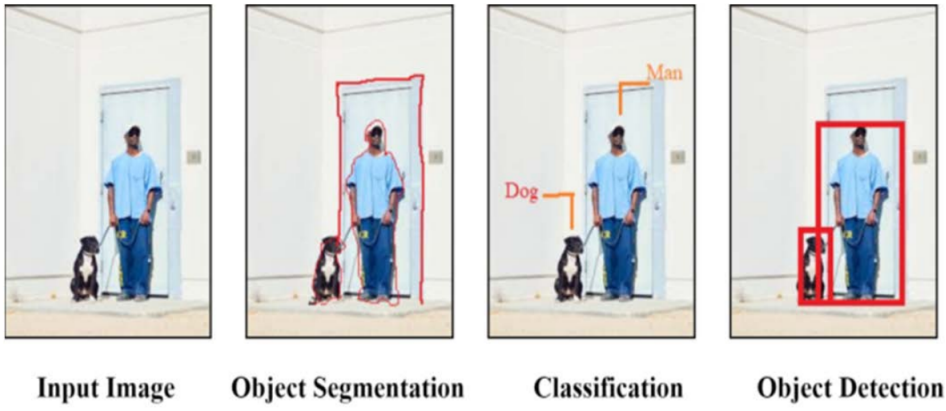
Capítulo 1

Introdução

A habilidade de computadores coletarem dados, retirar informações e tomar decisões baseadas em resultados do passado e do presente é o principal objetivo do aprendizado de máquina (MAHADEVKAR et al., 2022). Hoje é possível encontrar diversas aplicações no nosso cotidiano, como na recomendações de filmes em serviços de *streaming* até predições no mercado financeiro.

O aprendizado de máquina também é aplicado em visão computacional ao identificar e rastrear ações complexas realizadas facilmente pelos seres humanos como: segmentação, localização, classificação de imagens e detecção de objetos (ver Figura 1). Muito utilizada por exemplo na área médica para detecção de doenças, na agricultura para monitoramento das culturas e em carros autônomos.

Figura 1 – Segmentação, classificação de imagens e detecção de objetos



Fonte: Mahadevkar et al. (2022)

Segundo (MAHADEVKAR et al., 2022) algumas das técnicas mais utilizadas de aprendizado de máquina em visão computacional são: *zero-shot learning, active learning, contrastive learning, self-supervised learning, life-long learning, semi-supervised learning, ensemble learning, sequential learning* e *multi-view learning*.

A capacidade de sistemas computacionais aprenderem padrões a partir de dados e generalizarem tais conhecimentos para novas situações é o cerne do aprendizado de máquina (*Machine Learning*). Essa área tem impulsionado avanços notáveis em diversas aplicações da visão computacional, como segmentação, classificação e detecção de objetos, sendo essas tarefas essenciais em diferentes domínios.

Entre as abordagens contemporâneas, o aprendizado autossupervisionado (Self-Supervised Learning (SSL)) vem ganhando destaque por reduzir a dependência de grandes volumes de dados rotulados. Ao gerar seus próprios sinais de supervisão, o SSL permite o aprendizado de representações ricas a partir de dados não anotados, tornando-se particularmente relevante em contextos onde a rotulagem manual é onerosa ou impraticável.

Uma das estratégias mais promissoras dentro desse paradigma é o aprendizado contrastivo, que busca maximizar a similaridade entre pares positivos e minimizar a similaridade entre pares negativos no espaço de representação. Essa técnica tem se mostrado eficiente na construção de *embeddings* discriminativos, que preservam propriedades semânticas e favorecem a generalização em tarefas *downstream*. Métodos como Simple Contrastive Learning of Representations (SimCLR) (CHEN et al., 2020), Momentum Contrast (MoCo) (HE et al., 2020), Bootstrap Your Own Latent (BYOL) (GRILL et al., 2020) e Swapping Assignments Between Views (SwAV) (CARON et al., 2021) consolidaram o aprendizado contrastivo como um dos pilares do SSL moderno.

Entretanto, um desafio persistente refere-se à seleção aleatória de pares positivos e negativos, amplamente empregada em abordagens contrastivas tradicionais. Essa estratégia ignora a estrutura geométrica subjacente do espaço latente e tende a incluir amostras pouco informativas (pares triviais ou redundantes) que contribuem minimamente para o refinamento das fronteiras de decisão. Como consequência, o modelo pode apresentar convergência lenta, gradientes inconsistentes e representações menos coerentes do ponto de vista semântico.

Diante desse cenário, o presente trabalho propõe uma abordagem geométrica adaptativa para aprendizado contrastivo, denominada Aprendizado Contrastivo Guiado por Distância (*Distance-Guided Contrastive Learning* - DGCL). O método introduz uma política sistemática de seleção de pares, fundamentada na análise da configuração espacial das amostras no espaço latente. Utilizando projeções obtidas via t-SNE, o DGCL identifica os pares mais informativos, as amostras positivas mais distantes (*hard positives*) e as negativas mais próximas (*hard negatives*), e orienta o processo de otimização com base nessas relações.

Ao longo de ciclos iterativos de aprendizado, a estrutura latente é continuamente refinada, promovendo maior compactação intra-classe e melhor separabilidade entre classes distintas. Essa retroalimentação geométrica confere ao modelo uma capacidade aprimorada de organi-

zar semanticamente o espaço de representação, resultando em embeddings mais consistentes e discriminativos.

Os experimentos realizados em múltiplos conjuntos de imagens públicos (CIFAR-10, FER-13, KDEF e RAF-DB) demonstram que o DGCL supera significativamente modelos treinados sem o componente contrastivo, apresentando ganhos expressivos de acurácia e representatividade. Além disso, as análises qualitativas indicam uma clara evolução na coerência das representações ao longo das iterações de aprendizado, evidenciada pelas projeções t-SNE e pelas matrizes de confusão obtidas.

Este trabalho contribui, portanto, para o avanço das técnicas de aprendizado autossupervisionado, ao introduzir um mecanismo de seleção de pares guiado por distância que respeita a topologia do espaço latente, unindo conceitos de aprendizado contrastivo, análise de *manifolds* e aprendizado métrico em um único arcabouço.

1.1 Objetivos

1.1.1 Objetivo geral

Investigar e desenvolver uma abordagem de aprendizado autossupervisionado contrastivo guiado por distância, capaz de aprimorar a qualidade das representações aprendidas por meio de uma política adaptativa de seleção de pares baseada na estrutura geométrica do espaço latente.

1.1.2 Objetivos específicos

- ❏ Estudar os principais métodos de aprendizado autossupervisionado contrastivo presentes na literatura;
- ❏ Propor o método Distance-Guided Contrastive Learning (DGCL), integrando seleção geométrica de pares positivos e negativos;
- ❏ Implementar o algoritmo de treinamento do DGCL e avaliá-lo em diferentes cenários;
- ❏ Realizar experimentos em múltiplos conjuntos de imagens públicos (e.g. CIFAR-10, FER-13, KDEF e RAF-DB) e de diferentes domínios para comparar o desempenho contrastivo e não contrastivo;
- ❏ Analisar qualitativamente o comportamento do espaço de representação através de projeções t-SNE e matrizes de confusão;
- ❏ Discutir as contribuições, limitações e perspectivas futuras da abordagem proposta.

1.2 Organização

O presente trabalho está organizado nos seguintes capítulos:

- Capítulo 2 apresenta os conceitos fundamentais e uma revisão dos principais métodos de aprendizado autossupervisionado e contrastivo;
- Capítulo 3 descreve em detalhes o método proposto DGCL, incluindo sua formulação matemática, o algoritmo de treinamento e a política de seleção de pares;
- Capítulo 4 discute os experimentos realizados, os resultados quantitativos e qualitativos, bem como as análises das matrizes de confusão e das projeções t-SNE;
- Capítulo 5 apresenta as conclusões do trabalho, limitações do mesmo e sugestões de trabalhos futuros.

Capítulo 2

Revisão bibliográfica

Neste capítulo são apresentados os principais conceitos teóricos e fundamentos que subsidiam a proposta deste trabalho. São abordadas as bases do aprendizado autossupervisionado (Self-Supervised Learning — SSL) e do aprendizado contrastivo, além da discussão de métodos correlatos que inspiram a formulação do modelo Distance-Guided Contrastive Learning (DGCL). Por fim, são exploradas técnicas de visualização e análise de representações, como o *t-Distributed Stochastic Neighbor Embedding* (t-SNE), que desempenham papel central na proposta apresentada.

2.1 Conceitos básicos

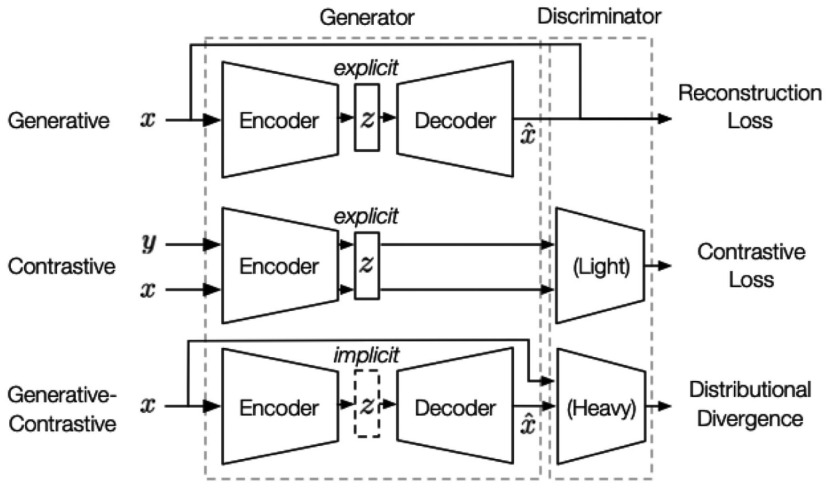
2.1.1 Aprendizado autossupervisionado (Self-Supervised Learning)

O aprendizado autossupervisionado (SSL) pode ser entendido como uma ponte entre o aprendizado supervisionado e o não supervisionado. Nesse paradigma, o modelo é treinado a partir de sinais de supervisão gerados automaticamente pelos próprios dados, dispensando a necessidade de rótulos manuais. Dessa forma, parte dos dados serve como "entrada" e outra parte é utilizada como "alvo" em tarefas de pretexto que orientam o aprendizado (RANI et al., 2023).

Liu et al. (2023) resume SSL em três categorias gerais e subsidiárias detalhadas:

- Generativa: busca reconstruir a entrada original a partir de uma codificação intermediária (ex.: autoencoders);
- Contrastiva: aprende representações discriminativas por meio da comparação entre amostras semelhantes e distintas;

Figura 2 – Três categorias de SSL.



Fonte: (LIU et al., 2023)

❑ Gerativo-contrastiva: combina as duas abordagens anteriores, integrando codificadores e decodificadores com discriminadores (ex.: GANs).

A Figura 2 ilustra o pipeline das três categorias gerais do SSL. Tal paradigma vem sendo amplamente adotado em tarefas de visão computacional, pois permite extrair representações visuais robustas em cenários com dados não rotulados. Essa característica o torna particularmente relevante em aplicações como diagnóstico médico, inspeção industrial e reconhecimento facial, nas quais a obtenção de anotações manuais é dispendiosa. Importante ressaltar que o foco do presente trabalho foi aplicado em abordagens contrastivas.

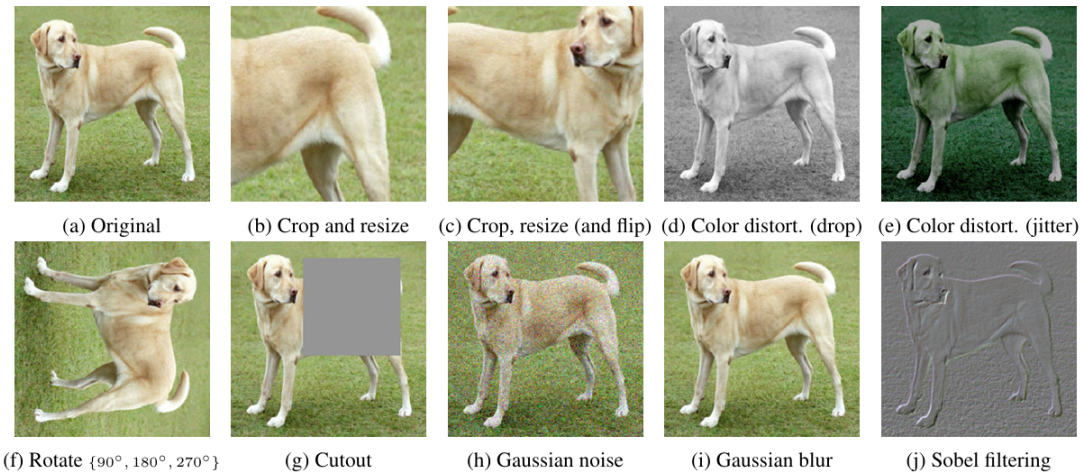
2.1.2 Aumento de dados

Grande parte dos métodos de aprendizado autossupervisionado, especialmente os contrastivos, depende de técnicas de aumento de dados (*data augmentation*) para gerar múltiplas visões de uma mesma imagem. Essas transformações introduzem variação nos dados e incentivam o modelo a aprender características invariantes, favorecendo a robustez das representações.

A Figura 3 ilustra alguns exemplos de imagens "aumentadas": (a) imagem original, (b) crop e resize, (c) crop, resize e flip, (d) distorção de cores (drop), (e) distorção de cores (jitter), (f) rotação, (g) cutout, (h) adição de ruído gaussiano, (i) suavizacão gaussiana, (j) detecção de bordas por Sobel.

Entre as transformações mais utilizadas estão: rotações, recortes aleatórios, inversões horizontais, variações de cor, borramento gaussiano e ajustes de brilho. O uso de augmentações é um componente essencial em frameworks como SimCLR (CHEN et al., 2020), que demonstram

Figura 3 – Exemplo de operadores de aumento de dados.



Fonte: (CHEN et al., 2020)

que a diversidade das transformações está diretamente relacionada à qualidade das representações aprendidas.

2.1.3 Aprendizado por Contraste

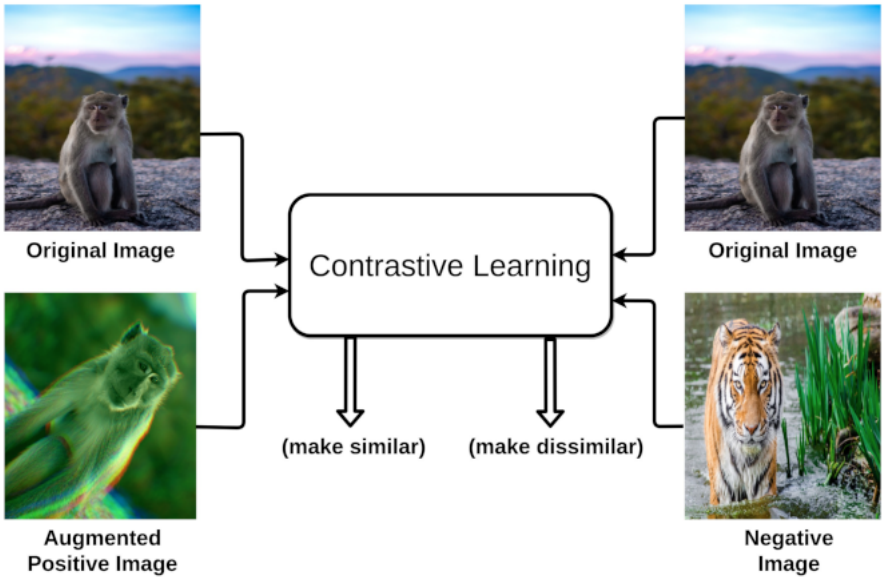
O aprendizado por contraste (*contrastive learning*) é uma abordagem de aprendizado de máquina que se destaca por sua capacidade de aprender representações úteis de dados, utilizando uma técnica de aprendizado não supervisionado. Trata-se de uma abordagem que busca aprender representações de dados de forma que exemplos semelhantes fiquem próximos no espaço de representação, enquanto exemplos diferentes fiquem distantes. O princípio fundamental é baseado na idéia de contraste entre pares de dados.

Em outras palavras, é uma abordagem que tem como objetivo agrupar amostras semelhantes ou seja mais próximas da imagem âncora, e amostras diferentes distanciar umas das outras, como na Figura 4. Para que isso ocorra, é utilizado uma métrica de semelhança para mensurar a proximidade entre as duas imagens. Em tarefas de visão computacional, utiliza-se função de perda para guiar o treinamento do modelo. A perda contrastiva (*contrastive loss*) penaliza o modelo se ele não conseguir manter os exemplos similares próximos e os diferentes distantes.

Existem vários métodos de *contrastive learning*, como por exemplo: SwAV, MoCo e SimCLR, que possuem abordagens modificadas mas que produzem resultados comparáveis. É possível encontrar também métodos com o BYOL que descartam amostragens negativas e aprendem por meio da previsão de diferentes transformações da mesma imagem âncora (LIU et al., 2023). Tais métodos serão detalhados na Seção 2.2.

O *contrastive learning* pode reduzir a necessidade de grandes quantidades de rótulos anotados, tornando-o útil para cenários com poucos dados rotulados. Além disso, as representações

Figura 4 – Demonstração básica do paradigma de Contrastive Learning



Fonte: Jaiswal et al. (2020, p. 2)

aprendidas via *contrastive learning* tendem a generalizar bem para várias tarefas, pois capturam características profundas e discriminativas dos dados.

No entanto, a escolha de pares de dados para contrastar pode ser desafiadora e influenciar significativamente o desempenho do modelo, bem como treinar modelos contrastivos pode ser computacionalmente intensivo, especialmente para grandes conjuntos de dados.

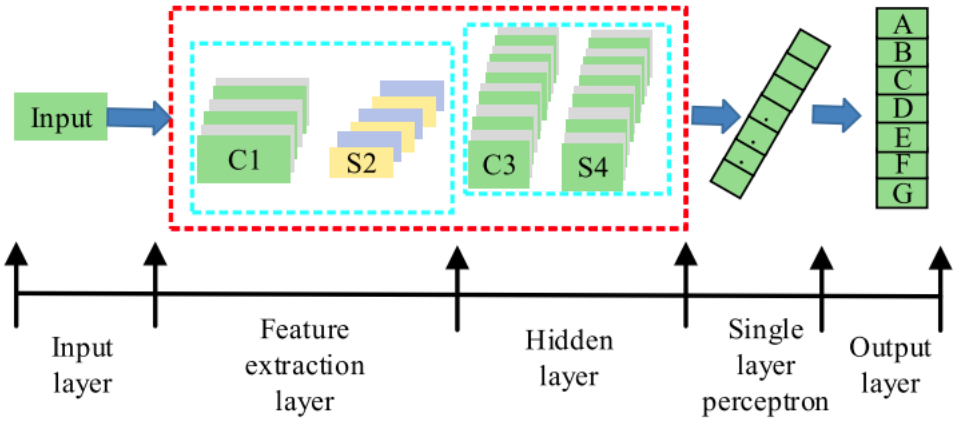
2.1.4 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (Convolutional Neural Network (CNN)) são um tipo especializado de rede neural projetado principalmente para processar dados estruturados em forma de grade, como imagens e vídeos. Elas se destacam em tarefas de classificação de imagens, detecção de objetos e segmentação semântica, pois aprendem automaticamente hierarquias espaciais de características relevantes a partir dos dados de entrada.

De forma geral, uma CNN é composta por três partes principais: camada de entrada, camadas ocultas e camada de saída (TIAN, 2020). A camada de entrada recebe a imagem original, enquanto a camada de saída fornece o resultado final da classificação. Já as camadas ocultas são responsáveis pela extração progressiva de características e são formadas por estruturas não lineares complexas, compostas por camadas de convolução e de subamostragem (*pooling*). A Figura 5 ilustra o pipeline básico de tal arquitetura.

Essas camadas ocultas realizam a extração e a classificação das características, e a otimização de seus parâmetros (especialmente dos filtros convolucionais e do perceptron de camada

Figura 5 – Estrutura de uma rede neural convolucional.



Fonte: Tian (2020)

única) pode melhorar significativamente a precisão da extração de características e o desempenho geral da classificação.

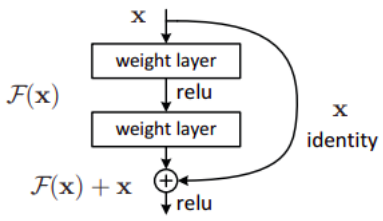
No contexto deste trabalho, a arquitetura CNN foi utilizada exclusivamente para a coleta de representações, buscando identificar as estruturas que melhor distinguem os diferentes conjuntos de imagens utilizados nos experimentos.

2.1.4.1 Arquitetura ResNet50

A Residual Networks (ResNet)50 é uma rede neural convolucional profunda composta por 50 camadas, projetada para resolver o problema do *vanishing gradient*, comum em redes muito profundas. Essa arquitetura foi introduzida no trabalho seminal de He et al. (2015), intitulado *Deep Residual Learning for Image Recognition*, que conquistou a competição ImageNet 2015.

A principal inovação da ResNet50 é o uso de **conexões residuais** (ou *skip connections*), que permitem ao modelo treinar redes extremamente profundas sem sofrer degradação de desempenho. Essas conexões possibilitam que o gradiente flua de forma mais estável entre as camadas, mitigando o problema de desaparecimento do gradiente e favorecendo a aprendizagem de representações mais robustas (LECUN; BENGIO; HINTON, 2015). A Figura 6 ilustra o bloco principal de uma arquitetura ResNet o qual configura-se como um bloco residual.

Figura 6 – Aprendizagem residual.



Fonte: He et al. (2015)

2.1.5 t-Distributed Stochastic Neighbor Embedding (t-SNE)

O *t-Distributed Stochastic Neighbor Embedding* (t-SNE) é uma técnica de redução de dimensionalidade amplamente utilizada para visualizar dados de alta dimensionalidade em espaços bidimensionais (2D) ou tridimensionais (3D). Seu principal objetivo é preservar a **estrutura local** dos dados, garantindo que amostras semelhantes no espaço original permaneçam próximas também no espaço reduzido (MAATEN; HINTON, 2008; MAATEN, 2009).

Ao contrário da Principal Component Analysis (PCA), que busca capturar a variância global dos dados, o t-SNE concentra-se em manter as **relações locais**, permitindo uma visualização mais eficaz de agrupamentos (*clusters*) e padrões intrínsecos em conjuntos de dados complexos (JUNG et al., 2024).

O t-SNE é particularmente útil em aplicações que envolvem dados de alta dimensionalidade, como imagens, texto e sinais espectrais, sendo frequentemente empregado para:

- ❏ Visualizar *clusters* e padrões em dados complexos de alta dimensão;
- ❏ Reduzir a dimensionalidade dos dados de forma não linear, facilitando sua exploração, interpretação e apresentação;
- ❏ Avaliar a separabilidade entre classes ou categorias aprendidas por modelos de aprendizado de máquina (SAKIB; SIDDIQUE; RAHMAN, 2020).

Apesar de sua eficácia para visualização, o t-SNE possui algumas limitações: apresenta alto custo computacional, não fornece mapeamento explícito para novos dados (isto é, não é trivial aplicar o modelo treinado a novos pontos) e pode não preservar bem a estrutura global dos dados. Estas questões são alvo de extensões recentes do método (ROCA et al., 2021; SUN; HAN; FAN, 2022). Essa compreensão crítica contribui para a seleção consciente de técnicas em contextos de análise exploratória e apresentação de resultados.

2.2 Trabalhos relacionados

2.2.1 Swapping Assignments Between Views - SwAV

O algoritmo *Swapping Assignments Between Views* (SwAV) (CARON et al., 2021) é uma técnica avançada de aprendizado de máquina que aborda a questão de como otimizar a aprendizagem em cenários com múltiplas representações ou "visões" dos dados. Esse conceito é relevante em contextos onde diferentes representações dos mesmos dados podem fornecer informações complementares, e o objetivo é combinar essas representações de maneira eficaz para melhorar o desempenho do modelo.

O SwAV busca melhorar a eficiência e a precisão dos modelos de aprendizado de máquina ao explorar como as "atribuições" ou rótulos podem ser trocados entre diferentes visões ou representações dos dados. Em outras palavras, a ideia é encontrar a melhor maneira de redistribuir ou ajustar as atribuições dos dados entre diferentes visões para maximizar a qualidade do aprendizado.

Em muitos problemas de aprendizado de máquina, pode-se ter múltiplas representações dos dados, cada uma capturando diferentes aspectos das informações. Por exemplo, pode-se ter imagens em diferentes resoluções ou com diferentes filtros aplicados.

Cada uma dessas representações pode fornecer informações únicas e valiosas, e o desafio é como combinar ou "trocar" as atribuições entre essas visões para melhorar o aprendizado.

O algoritmo SwAV pode ser descrito em etapas gerais:

1. Geração de Múltiplas Visões: Primeiro, o algoritmo considera diferentes representações dos dados. Cada visão pode ser uma versão transformada ou uma representação diferente dos mesmos dados brutos;

2. Atribuição Inicial: Inicialmente, cada visão pode ter um conjunto de atribuições ou rótulos que foram obtidos através de um processo de treinamento inicial ou de uma abordagem padrão;

3. Troca de Atribuições: O núcleo do SwAV envolve a troca ou redistribuição das atribuições entre as visões. Isso pode ser feito de várias maneiras, dependendo da estratégia específica do algoritmo:

   - ❏ Troca Direta: Alterar diretamente as atribuições de uma visão para outra;
   - ❏ Ajuste Baseado em Desempenho: Usar métricas de desempenho para guiar a troca de atribuições, de modo a melhorar a precisão geral;
   - ❏ Otimização: Aplicar técnicas de otimização para encontrar a melhor configuração de atribuições que maximiza um critério específico, como a acurácia ou a consistência entre as visões.

4. Treinamento e Avaliação: Após a troca de atribuições, o modelo é re-treinado usando as novas atribuições para avaliar se houve uma melhoria no desempenho. Isso pode envolver a repetição do processo de troca e ajuste até que um critério de desempenho satisfatório seja alcançado.

A troca de atribuições pode ajudar a combinar a informação complementar de diferentes visões, resultando em um modelo mais robusto e preciso. O SwAV pode ser aplicado em uma ampla gama de contextos, desde a visão computacional até o processamento de linguagem natural e aprendizado multimodal.

Ao redistribuir as atribuições, o algoritmo pode melhorar a eficiência do treinamento, potencialmente reduzindo a necessidade de grandes quantidades de dados rotulados.

Assim, o SwAV é uma abordagem que lida com a complexidade de múltiplas representações dos dados ao otimizar como as atribuições são distribuídas entre essas visões. Ao explorar e ajustar essas atribuições, o SwAV pode melhorar significativamente o desempenho do modelo e a eficiência do treinamento, aproveitando ao máximo as informações complementares fornecidas pelas diferentes representações dos dados.

2.2.2 Momentum Contrast - MoCo

O algoritmo *Momentum Contrast* (MoCo) (HE et al., 2020) é uma técnica de aprendizado de máquina usada para tarefas de aprendizado autossupervisionado, especialmente em representações de aprendizado profundo. Desenvolvido por pesquisadores da Facebook AI Research (FAIR), o MoCo tem se destacado por sua capacidade de aprender representações eficazes de dados sem a necessidade de rótulos anotados.

Vale ressaltar que em aprendizado autossupervisionado, o objetivo é aprender boas representações dos dados a partir de uma grande quantidade de dados não rotulados. Isso é particularmente útil em cenários onde rotular dados é caro ou demorado. O MoCo é uma abordagem que visa melhorar a qualidade das representações aprendidas, permitindo que modelos de redes neurais generalizem melhor para diversas tarefas.

O MoCo se baseia na idéia de contrastive learning (aprendizado contrastivo), onde o modelo é treinado para distinguir entre pares de exemplos semelhantes e dissimilares. Em vez de usar pares de dados explícitos (como em métodos tradicionais de aprendizado supervisionado), o MoCo cria pares positivos e negativos de dados de maneira auto-supervisionada, sem necessidade de rótulos externos. Os componentes principais do método são:

- ❏ Encoder: O MoCo utiliza uma rede neural (o encoder) para extrair representações dos dados. Esse encoder transforma os dados brutos em vetores de características;

- ❏ Momentum Encoder: Além do encoder principal, o MoCo usa um segundo encoder chamado de momentum encoder. Este encoder é uma cópia do encoder principal, mas com

atualizações mais lentas, baseadas em uma média móvel das atualizações do encoder principal;

- ❏ Queue: Para criar uma base de dados de exemplos negativos, o MoCo mantém uma fila (queue) de representações de exemplos passados. Isso permite que o modelo tenha acesso a uma grande quantidade de exemplos negativos durante o treinamento, o que é crucial para o aprendizado contrastivo eficaz.

De maneira geral o MoCo realiza os seguintes passos:

1. Geração de Representações: Dados são passados pelo encoder principal e pelo momentum encoder para gerar representações. A ideia é que o encoder principal produz representações que são contrastadas com representações da fila, enquanto o momentum encoder fornece representações para atualização mais estável;

2. Criação de Pares Positivos e Negativos: O modelo gera pares positivos (representações de exemplos semelhantes, como diferentes augmentações da mesma imagem) e pares negativos (representações de exemplos dissimilares, como imagens diferentes);

3. Contrastive Loss: O algoritmo utiliza uma função de perda contrastiva para treinar o modelo. Essa perda penaliza o modelo se ele não conseguir distinguir entre pares positivos e negativos corretamente. O MoCo usa uma variação da função de perda InfoNCE (Noise Contrastive Estimation) para otimizar a separação entre pares positivos e negativos;

4. Atualização do Momentum Encoder: O momentum encoder é atualizado com base nas atualizações do encoder principal. Isso é feito usando uma média móvel das atualizações, o que proporciona uma representação mais estável e consistente ao longo do treinamento;

5. Atualização da Fila: A fila é atualizada com novas representações negativas durante o treinamento. Isso garante que o modelo tenha acesso a uma grande variedade de exemplos negativos.

As principais vantagens do método é que o uso de uma fila para armazenar representações negativas permite que o MoCo seja treinado de forma eficiente, mesmo com grandes conjuntos de dados. Além disso, a técnica de momentum fornece atualizações mais estáveis para o encoder, o que ajuda a evitar a instabilidade que pode ocorrer com métodos tradicionais de aprendizado contrastivo.

O *Momentum Contrast* (MoCo) é uma abordagem poderosa para o aprendizado autossupervisionado que combina técnicas de aprendizado contrastivo com estratégias inovadoras para manter uma fila de exemplos negativos e usar encoders com atualizações baseadas em momentum. Isso permite que o MoCo aprenda representações robustas e de alta qualidade a partir de grandes volumes de dados não rotulados, o que pode ser altamente benéfico em uma variedade de tarefas de aprendizado de máquina.

2.2.3 Simple Contrastive Learning of Representations - SimCLR

O *Simple Contrastive Learning of Representations* (SimCLR) (CHEN et al., 2020) é um algoritmo de aprendizado autossupervisionado desenvolvido por pesquisadores do Google Brain. Assim como os outros métodos o objetivo final do mesmo é aprender representações de dados sem a necessidade de rótulos anotados. O SimCLR é uma das abordagens mais influentes no campo do aprendizado contrastivo e tem sido fundamental na evolução das técnicas de aprendizado autossupervisionado.

O SimCLR se destaca por sua simplicidade e eficácia em aprender representações úteis através de um processo de contraste, onde o modelo é treinado para distinguir entre pares de exemplos semelhantes e dissimilares.

O método abordado baseia-se na técnica de aprendizado contrastivo. Dessa forma, o objetivo é maximizar a similaridade entre pares positivos (exemplos semelhantes) e minimizar a similaridade entre pares negativos (exemplos dissimilares).

O SimCLR embasa-se em alguns componentes principais que são um encoder, augmentações e uma denominada cabeça de projeção. O modelo utiliza uma rede neural, tipicamente uma rede convolucional profunda como a ResNet, para extrair representações de dados brutos. Assim, como em outros métodos o encoder transforma imagens em vetores de características de alta dimensão.

Para criar pares positivos, o SimCLR aplica diferentes transformações (ou augmentações) nas mesmas imagens. Após a extração das representações pelo encoder, o SimCLR usa uma cabeça de projeção, geralmente uma rede neural de duas camadas, para mapear as representações para um espaço de menor dimensão onde a comparação contrastiva é realizada.

Dados os componentes principais do método em questão o funcionamento do mesmo segue algumas fases. Primeiramente os dados (como imagens) são passados pelo encoder para gerar representações. Essas representações são obtidas após aplicar um conjunto de augmentações às imagens.

O SimCLR então cria pares positivos ao aplicar diferentes augmentações na mesma imagem e pares negativos ao utilizar representações de imagens diferentes.

O método utiliza a função de perda contrastiva chamada InfoNCE (*Noise Contrastive Estimation*). A InfoNCE penaliza a similaridade entre pares negativos e recompensa a similaridade entre pares positivos. A função de perda é projetada para maximizar a distância entre representações de diferentes imagens e minimizar a distância entre representações de augmentações da mesma imagem.

O modelo é treinado para minimizar a perda contrastiva, ajustando os parâmetros do encoder e da cabeça de projeção para melhorar a qualidade das representações. Após o treinamento, as representações aprendidas pelo encoder podem ser utilizadas para diversas tarefas downstream, como classificação e detecção, ajustando-se para tarefas específicas com uma pequena quantidade de dados rotulados.

Dessa forma, o SimCLR é conhecido por sua abordagem relativamente simples e direta

para aprendizado contrastivo, sem a necessidade de técnicas complexas para o treinamento. O modelo tem mostrado desempenho competitivo na aprendizagem de representações com uma arquitetura de rede neural padrão e sem o uso de exemplos negativos explicitamente fornecidos.

2.2.4 Bootstrap Your Own Latent - BYOL

O algoritmo *Bootstrap Your Own Latent* (BYOL) (GRILL et al., 2020) foi proposto por pesquisadores da Facebook AI Research (FAIR), e representa um avanço significativo em comparação com métodos tradicionais de aprendizado contrastivo.

Técnicas de aprendizado contrastivo, como o MoCo (*Momentum Contrast*), dependem fortemente de exemplos negativos (dados dissimilares) para treinar o modelo. O BYOL foi desenvolvido para superar algumas limitações dessas técnicas contrastivas, especialmente a necessidade de exemplos negativos, proporcionando uma abordagem mais direta e eficaz.

O BYOL é baseado na idéia de aprendizado autossupervisionado sem contraste. Em vez de utilizar exemplos negativos para treinar o modelo, o BYOL usa uma estratégia de bootstrap, onde o objetivo é maximizar a similaridade entre as representações de duas versões augmentadas da mesma entrada, mas sem a necessidade de comparar com representações negativas.

O BYOL usa uma rede neural para transformar dados brutos em representações vetoriais. O modelo consiste em dois encoders: o encoder principal e o encoder target. O encoder target é uma cópia do encoder principal, mas com atualizações mais lentas. A atualização é baseada em uma média móvel das atualizações do encoder principal. O algoritmo aplica diferentes transformações (ou augmentações) nos dados para criar duas versões distintas da mesma entrada. Essas versões são usadas para gerar representações que serão comparadas.

De forma geral o BYOL processa os dados por meio de ambos os encoders (principal e target) para gerar representações. O encoder principal gera uma representação “online”, enquanto o encoder target gera uma representação “target” com atualizações mais lentas. O algoritmo aplica duas diferentes augmentações na mesma entrada de dados para criar duas versões distintas. Por exemplo, pode aplicar rotações, cortes ou ajustes de brilho nas imagens.

O objetivo do BYOL é maximizar a similaridade entre as representações das duas versões augmentadas da mesma entrada. Isso é feito utilizando uma função de perda que incentiva que as representações das versões augmentadas sejam próximas, sem a necessidade de comparar com exemplos negativos.

O encoder target é atualizado usando uma média móvel das atualizações do encoder principal. Isso fornece uma representação mais estável e consistente, que é crucial para a eficácia do algoritmo. O modelo é treinado para minimizar a perda entre as representações das versões augmentadas, incentivando o encoder principal a gerar representações que sejam semelhantes às do encoder target.

O BYOL simplifica o treinamento autossupervisionado ao eliminar a necessidade de exemplos negativos, tornando o processo mais direto e menos dependente da escolha de exemplos negativos.

Assim, o BYOL é uma abordagem para aprendizado autossupervisionado que oferece uma alternativa aos métodos contrastivos tradicionais. Ao focar na maximização da similaridade entre representações de versões augmentadas da mesma entrada e eliminar a necessidade de exemplos negativos, o BYOL proporciona uma maneira eficiente e eficaz de aprender representações de alta qualidade a partir de dados não rotulados.

2.3 Conclusão

O presente capítulo apresentou os fundamentos teóricos que sustentam o desenvolvimento deste trabalho, abordando os principais conceitos relacionados ao *aprendizado autossupervisionado* e ao *aprendizado contrastivo*. Foram discutidos os princípios que orientam a formação de representações latentes a partir de dados não rotulados, bem como a importância das funções de perda contrastivas e das estratégias de aumento de dados na construção de *embeddings* discriminativos.

A análise dos métodos clássicos como *SimCLR*, *MoCo*, *BYOL* e *SwAV*, permitiu compreender as diferentes abordagens de estruturação do aprendizado contrastivo e suas limitações, em especial no que se refere à **seleção aleatória de pares positivos e negativos**, que tende a negligenciar relações geométricas mais informativas entre amostras.

Além disso, foi discutida a técnica de análise e visualização de manifolds, *t-Distributed Stochastic Neighbor Embedding (t-SNE)*, cuja aplicação transcende a mera visualização, servindo como ferramenta para estimar a geometria do espaço latente.

A partir dessas discussões, observou-se que a eficiência do aprendizado contrastivo depende fortemente da capacidade do modelo em explorar a **estrutura geométrica intrínseca** dos dados. Essa constatação motivou a formulação do método *Distance-Guided Contrastive Learning (DGCL)*, que integra de forma inédita o aprendizado contrastivo e a análise geométrica adaptativa.

Em síntese, este capítulo estabelece o embasamento conceitual necessário para a proposta metodológica apresentada no capítulo seguinte, consolidando as bases teóricas que orientam a formulação, implementação e avaliação do método DGCL.

Capítulo 3

Materiais e métodos

Este capítulo apresenta os fundamentos metodológicos e experimentais que sustentam o desenvolvimento do método proposto. Inicialmente, são descritos os aspectos gerais da arquitetura e da estratégia de treinamento utilizada. Em seguida, é detalhado o método proposto DGCL (Distance-Guided Contrastive Learning), incluindo sua formulação, política de seleção de pares e algoritmo. Por fim, são apresentados os conjuntos de imagens empregados nas análises do mesmo.

3.1 Método Proposto

O método proposto **Distance-Guided Contrastive Learning (DGCL)** integra extração de características, projeção no espaço latente, seleção de pares e atualização do modelo em ciclos iterativos. A arquitetura geral é apresentada na Figura 7.

O pipeline proposto para o aprendizado contrastivo guiado por distância é composto por três estágios principais:

1. **Extração de representações iniciais:** Um codificador convolucional f_θ é treinado para mapear cada imagem de entrada x em um vetor de características $\mathbf{z} = f_\theta(x)$, que representa sua posição em um espaço latente de dimensão reduzida. Nesta etapa, as amostras são processadas de forma independente, sem ainda considerar relações geométricas entre elas.

2. **Projeção e análise geométrica:** As representações latentes obtidas são projetadas em um espaço bidimensional utilizando o t-Distributed Stochastic Neighbor Embedding (t-SNE), técnica que preserva a proximidade local entre amostras. Essa projeção possibilita calcular distâncias euclidianas aproximadas entre amostras no manifold subjacente, permitindo identificar relações de similaridade e dissimilaridade.

3. **Seleção guiada de pares e atualização contrastiva:** A partir das distâncias no espaço projetado, o modelo seleciona pares positivos e negativos de forma informada, priorizando amostras que promovem maior ganho de informação contrastiva. Os pares selecionados são então utilizados para computar a perda contrastiva e atualizar os parâmetros do codificador $f_\theta(x)$.

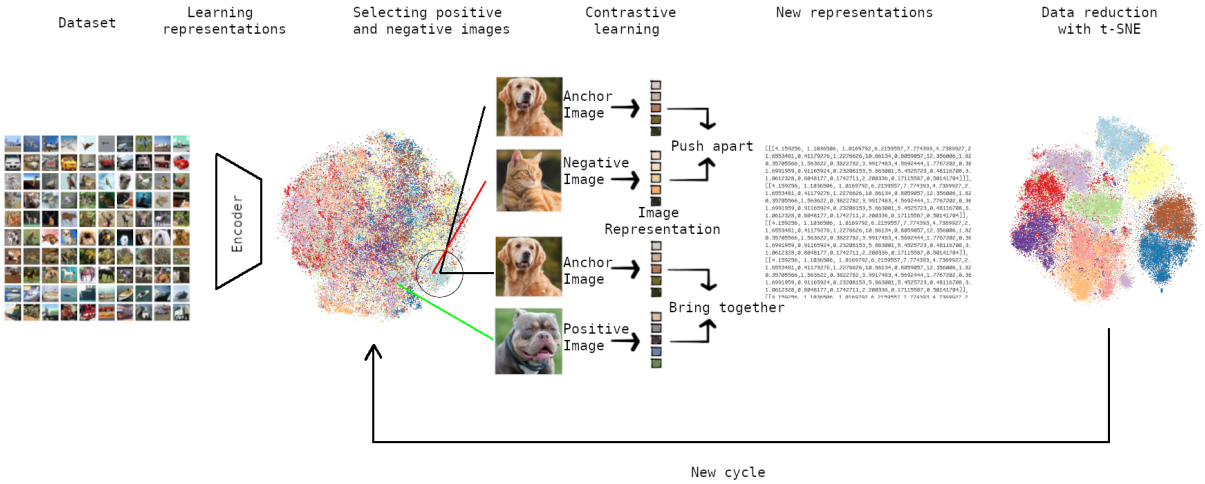


Figura 7 – Visão geral esquemática do framework DGCL, integrando seleção de pares baseada em t-SNE e otimização contrastiva.

O processo é iterativo e contínuo: as representações aprendidas a cada época influenciam a geometria latente estimada via t-SNE, que por sua vez orienta o próximo ciclo de seleção de pares. Essa retroalimentação geométrica é o núcleo do método DGCL.

Formalmente, seja $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ um conjunto de dados rotulado com N imagens, em que x_i é a i-ésima imagem e y_i seu rótulo de classe. Embora o aprendizado autossupervisionado (SSL) não utilize rótulos durante o treinamento, os mesmos são empregados apenas para orientar a política de seleção (em cenários totalmente não rotulados, é possível substituí-los por identificação de positivos baseada em *clusters*).

O extrator de características f_θ produz embeddings $\mathbf{z}_i = f_\theta(x_i) \in \mathbb{R}^d$, que são normalizados em L2 para comprimento unitário. Em seguida, os embeddings são projetados em um manifold 2D utilizando t-SNE, denotados por $\mathbf{u}_i = \text{tSNE}(\mathbf{z}_i)$.

O t-SNE no DGCL atua não apenas como ferramenta de visualização, mas também como estimador da geometria latente. Apesar de ser não paramétrico, o mesmo preserva relações locais, permitindo aproximar a topologia do *manifold*, que é utilizada para guiar a seleção de pares. As projeções t-SNE podem ser recalculadas periodicamente, garantindo estabilidade e baixo custo computacional. Em trabalhos futuros, pretende-se explorar *embeddings* paramétricos escaláveis (ex.: UMAP ou t-SNE paramétrico neural) para aplicação em conjuntos de dados maiores.

A motivação para tal proposta embasa-se na premissa de que os métodos contrastivos tradicionais, como SimCLR e MoCo, selecionam pares positivos e negativos de forma aleatória.

Essa aleatoriedade tende a incluir exemplos triviais, cujas contribuições para o aprendizado são pequenas. Por exemplo, dois exemplos da mesma classe que já são próximos no espaço latente pouco alteram o gradiente da função de perda quando aproximados novamente. De modo análogo, amostras de classes distintas que já estão muito afastadas não fornecem informação adicional relevante ao processo de otimização. O DGCL busca resolver esse problema introduzindo uma política adaptativa de seleção de pares, fundamentada na estrutura geométrica das representações. Essa política identifica pares mais informativos (i.e. aqueles que estão nas bordas das regiões de decisão) e os utiliza para refinar o espaço latente. O princípio subjacente é que pares difíceis (*hard pairs*) carregam gradientes mais úteis, favorecendo a formação de fronteiras mais nítidas entre classes.

3.1.1 Política de Seleção de Pares

A política de seleção de pares constitui o núcleo do método proposto, uma vez que define como as amostras positivas e negativas são escolhidas durante o processo de aprendizado contrastivo. Diferentemente das abordagens tradicionais, que realizam essa seleção de maneira aleatória, o método proposto baseia-se em uma estratégia orientada pela estrutura geométrica do espaço latente.

Tal política tem como objetivo identificar, para cada amostra âncora (imagem em análise), os exemplos mais informativos para o aprendizado (i.e. os pares positivos mais distantes dentro da mesma classe e os pares negativos mais próximos pertencentes a classes distintas). Essa escolha direciona o processo de otimização para regiões mais desafiadoras do espaço de representação, promovendo uma separação inter-classe mais nítida e uma compactação intra-classe mais consistente.

Para tanto, define-se o conjunto de índices intra-classe $\mathcal{S}_i^+ = \{j \mid y_j = y_i, j \neq i\}$ e o conjunto inter-classe $\mathcal{S}_i^- = \{j \mid y_j \neq y_i\}$. A distância euclidiana no espaço t-SNE é expressa por:

$$D_{ij} = \|\mathbf{u}_i - \mathbf{u}_j\|_2.$$

Selecionam-se os conjuntos positivos e negativos para a âncora *i* como:

$$\mathcal{P}_i(k_p) = \text{TopK}\left(\mathcal{S}_i^+,\ \text{score}(j) = D_{ij},\ k_p\right), \quad \mathcal{N}_i(k_n) = \text{TopK}\left(\mathcal{S}_i^-,\ \text{score}(j) = -D_{ij},\ k_n\right),$$

onde $\text{TopK}(S, \text{score}, k)$ retorna os k índices com maiores valores de 'score'. Para negativos, utiliza-se $-D_{ij}$, de modo que os top-k correspondam aos mais próximos. Os hiperparâmetros k_p e k_n podem ser constantes ou funções adaptativas $k_p(i), k_n(i)$, em função da geometria local da âncora.

Para tal parametrização adaptativa de k_p e k_n propõem-se duas estratégias automáticas:

(A) Seleção proporcional baseada em densidade.

Define-se

$$\rho_i^+ = \frac{1}{|\mathcal{S}_i^+|} \sum_{j \in \mathcal{S}_i^+} \mathbf{1}\{D_{ij} > \mu_+(i)\}, \quad \rho_i^- = \frac{1}{|\mathcal{S}_i^-|} \sum_{j \in \mathcal{S}_i^-} \mathbf{1}\{D_{ij} < \mu_-(i)\},$$

onde $\mu_+(i)$ e $\mu_-(i)$ representam as médias intra- e inter-classe das distâncias da âncora i. Define-se:

$$k_p(i) = \left\lceil \alpha \cdot \rho_i^+ \cdot |\mathcal{S}_i^+| \right\rceil, \quad k_n(i) = \left\lceil \beta \cdot \rho_i^- \cdot |\mathcal{S}_i^-| \right\rceil,$$

com $\alpha, \beta \in (0,1]$ controlando a esparsidade. Esta abordagem aumenta $k_p(i)$ quando existem muitos positivos distantes (exemplos intra-classe difíceis) e $k_n(i)$ quando há muitos negativos próximos (confusões inter-classe).

(B) Seleção baseada em percentis.

Fixam-se percentis $p_+, p_- \in (0,100)$ e define-se:

$$k_p(i) = \#\{j \in \mathcal{S}_i^+ \mid D_{ij} \geq \text{percentil}(\{D_{il}\}_{l \in \mathcal{S}_i^+}, p_+)\},$$

$$k_n(i) = \#\{j \in \mathcal{S}_i^- \mid D_{ij} \leq \text{percentil}(\{D_{il}\}_{l \in \mathcal{S}_i^-}, p_-)\}.$$

Selecionam-se os positivos além do percentil p_+ e os negativos dentro do percentil p_-. Os percentis podem ser ajustados ou definidos adaptativamente (ex.: $p_+ = 80$, $p_- = 20$).

Para garantir reprodutibilidade, nos experimentos realizados no presente trabalho foram adotados valores fixos $k_p = k_n = k^\star$ com $k^\star = 5$. Porém, vale ressaltar que as regras adaptativas propostas foram pensadas para possibilitar escalabilidade a conjuntos de dados com cardinalidade variável por classe, onde tais valores seriam dinamicamente definidos dadas as distribuições de tais espaços.

3.1.2 Objetivo Contrastivo

Dadas as seleções $\mathcal{P}_i$ e $\mathcal{N}_i$, define-se a perda por âncora:

$$\mathcal{L}_i(\theta) = \frac{1}{|\mathcal{P}_i|} \sum_{j \in \mathcal{P}_i} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \max\left(0, m - \|\mathbf{z}_i - \mathbf{z}_j\|_2\right)^2,$$

e a perda total como a média empírica:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i(\theta).$$

A otimização realiza-se via gradiente estocástico em θ, normalizando-se os embeddings após cada *forward pass* (propagação para frente). O Algoritmo 1 descreve em detalhes o pipeline de treinamento da abordagem DGCL proposta.

Algoritmo 1 Distance-Guided Contrastive Learning (DGCL) - Pipeline de Treinamento

Require: Conjunto de dados $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, backbone f_θ, projeção g (opcional), frequência t-SNE T_{tsne}, margem $m > 0$, k_p, k_n iniciais ou regra adaptativa, total de épocas E
Ensure: Parâmetros treinados θ

Inicializar θ (aleatório ou pré-treinado)
for época $= 1$ até E **do**
  Calcular embeddings $\mathbf{z}_i \leftarrow f_\theta(x_i)$ e normalizar $\mathbf{z}_i \leftarrow \mathbf{z}_i / \|\mathbf{z}_i\|_2$
  if época mod $T_{tsne} = 0$ **then**
    Projetar $\mathbf{u}_i = \text{tSNE}(\mathbf{z}_i)$ e calcular $D_{ij} = \|\mathbf{u}_i - \mathbf{u}_j\|_2$
  end if
  Formar mini-batches $\mathcal{B}$ embaralhando índices
  for cada mini-batch $B \in \mathcal{B}$ **do**
    Identificar conjuntos $\mathcal{S}_i^+$ e $\mathcal{S}_i^-$ para cada âncora
    Calcular $k_p(i), k_n(i)$ (fixo ou adaptativo)
    Selecionar $\mathcal{P}_i$ e $\mathcal{N}_i$ usando `TopK`
    Calcular perda $\mathcal{L}_B = \frac{1}{|B|} \sum_{i \in B} \mathcal{L}_i(\theta)$
    Retropropagar gradientes $\nabla_\theta \mathcal{L}_B$ e atualizar θ
  end for
end for
return θ

3.1.2.1 Descrição Passo a Passo do DGCL

Os passos realizados pelo método proposto podem ser sumarizados da seguinte forma:

Inicialização. A rede backbone f_θ é inicializada aleatoriamente ou com pesos pré-treinados (ex.: ImageNet). Define-se a margem m, a frequência de atualização das projeções t-SNE (T_{tsne}) e a política de seleção de pares (fixa ou adaptativa).

Cálculo dos embeddings. Em cada época, f_θ gera embeddings normalizados $\mathbf{z}_i$, garantindo estabilidade das distâncias e gradientes.

Projeção latente e matriz de distâncias. A cada T_{tsne} épocas, o conjunto de embeddings é projetado para 2D utilizando t-SNE para obter $\mathbf{u}_i$. Calculam-se os pares distâncias euclidianas D_{ij} neste manifold 2D. Como o t-SNE é computacionalmente intensivo, esta etapa pode ser executada com menor frequência (por exemplo, a cada 10–20 épocas) em vez de em cada época.

Processamento de mini-batch. O treinamento prossegue com SGD (stochastic gradient descent) padrão em mini-batches. Para cada âncora no batch, identificam-se os conjuntos candidatos intra-classe e inter-classe $\mathcal{S}_i^+$ e $\mathcal{S}_i^-$ (os rótulos são usados apenas para seleção; candidatos alternativos baseados em clusters não supervisionados podem substituir os rótulos). De acordo com a política escolhida (fixa ou adaptativa), calculam-se os valores de $k_p(i)$ e $k_n(i)$. Em seguida, para cada âncora, selecionam-se os $k_p(i)$ positivos mais distantes e os $k_n(i)$ negativos mais próximos no manifold t-SNE, formando $\mathcal{P}_i$ e $\mathcal{N}_i$. As perdas por âncora $\mathcal{L}_i(\theta)$ são agregadas ao longo do batch em $\mathcal{L}_B$, retropropagadas e utilizadas para atualizar θ.

Refinamento iterativo. À medida que o treinamento avança e θ muda/evolui, os embeddings $\mathbf{z}_i$ se deslocam; quando o t-SNE é recalculado, as distâncias D_{ij} refletem a geometria

atualizada. Este ciclo de retroalimentação concentra a otimização em contrastes progressivamente mais informativos: nos ciclos iniciais expõem-se separações grosseiras, enquanto nos ciclos posteriores o foco recai sobre confusões de menor escala com granularidade mais refinada.

Vale ressaltar que calcular a matriz completa de pares de distâncias requer $O(N^2)$ operações e $O(N^2)$ de memória. Para grandes conjuntos de dados (i.e. milhões de imagens), recomenda-se: (i) calcular distâncias apenas entre cada âncora e um subconjunto de candidatos via busca aproximada de vizinhos mais próximos (ANN - *Approximate Nearest Neighbor*); (ii) aplicar t-SNE em um subconjunto estratificado e utilizar mapeamento paramétrico para novos pontos; ou (iii) empregar UMAP (*Uniform Manifold Approximation and Projection*), o qual é mais rápido e escalável como uma alternativa para a projeção dos manifolds. Entretanto, no presente trabalho o maior dataset utilizado foi o CIFAR-10 (ver Seção 3.2), o qual apresenta 50000 imagens de treinamento, favorecendo assim a viabilidade do cálculo exato dos pares de distância dada a cardinalidade dos datasets utilizados.

Outro ponto a ser abordado é que tais cálculos também podem ser otimizados por meio da utilização da construção de índices baseados em espaços métricos, como por exemplo métodos de acesso métricos (MAMs) tais como M-tree, Slim-tree (VIET; ANH, 2013), dentre outros, favorecendo a poda de cálculos de distância e aumentando a eficiência. Apesar do foco do presente trabalho não ter sido na aplicação de MAMs acredita-se que tal emprego possibilitaria a escalabilidade da solução proposta com cálculos de distância exatos para conjuntos de imagens de alta cardinalidade ($|\mathcal{D}| \gg$1M de amostras), além de prover diversas possibilidades como operações de carga rápida (*bulk loading*). No entanto, o emprego de tais métodos de indexação devem ser analisados adequadamente (trade-off) uma vez que o t-SNE recalcula as distâncias segundo um dado intervalo de épocas, fato que demandaria a reorganização de tais estruturas a cada T_{tsne} épocas.

3.2 Descrição dos Conjuntos de Imagens

Para avaliar o método proposto, empregou-se diferentes conjuntos de imagens públicos. O primeiro a ser considerado foi o bem estabelecido conjunto de dados **CIFAR-10** (KRIZHEVSKY; HINTON, 2009), um benchmark comumente utilizado para classificação de imagens e aprendizado de representações autossupervisionado. O CIFAR-10 é composto por 60.000 imagens coloridas com resolução espacial de 32×32 pixels, distribuídas de forma uniforme em 10 categorias semânticas mutuamente exclusivas: *airplane, automobile, bird, cat, deer, dog, frog, horse, ship* e *truck*. O conjunto de dados é dividido em 50.000 imagens de treinamento e 10.000 imagens de teste, com 6.000 instâncias por classe, garantindo proporções equilibradas entre as classes.

Apesar do tamanho relativamente pequeno das imagens, o CIFAR-10 apresenta considerável variabilidade na aparência dos objetos, no cenário de fundo, nas condições de iluminação e

no ponto de vista. Essas características tornam-no particularmente adequado para a avaliação de métodos de aprendizado de representações contrastivas, que devem codificar características robustas e invariantes capazes de generalizar através das variações intra-classe, ao mesmo tempo em que discriminam entre categorias visualmente semelhantes (ex.: *cat* vs. *dog*, ou *ship* vs. *airplane*).

O CIFAR-10 foi escolhido em detrimento de conjuntos de dados de maior resolução (ex.: ImageNet) por duas razões principais. Primeiro, sua escala moderada permite um cálculo eficiente do t-SNE e a análise iterativa do manifold ao longo dos ciclos de treinamento. Segundo, sua diversidade visual e estrutura equilibrada fornecem um ambiente adequado para analisar como a seleção de pares geométrica influencia a formação dos embeddings contrastivos. Essas propriedades tornam o CIFAR-10 um ambiente ideal para isolar e estudar os efeitos do mecanismo de seleção guiada por distância introduzido no DGCL.

Outro conjunto utilizado foi o **FER13** (*Facial Emotion Recognition 2013*) (CARRIER; COURVILLE, 2017) o qual é amplamente utilizado na área de reconhecimento de emoções faciais, especialmente para o treinamento e avaliação de modelos de aprendizado profundo. Ele contém imagens em escala de cinza de rostos humanos, cada uma rotulada com uma das sete emoções básicas.

As imagens possuem resolução de 48×48 pixels e o conjunto completo é composto por 35.685 imagens. Deste total, 28.709 imagens destinam-se ao treinamento, 3.589 constituem o teste público e outras 3.589 formam o teste privado. As emoções rotuladas incluem raiva (Anger), desgosto (Disgust), medo (Fear), felicidade (Happy), tristeza (Sad), surpresa (Surprise) e neutro (Neutral). O conjunto de dados foi criado por Pierre-Luc Carrier e Aaron Courville e está disponível no *Wolfram Data Repository*[1].

O FER13 é amplamente utilizado em pesquisas e aplicações de reconhecimento de emoções faciais, interação humano-computador, análise comportamental e desenvolvimento de sistemas de feedback emocional. O conjunto apresenta desbalanceamento nas classes, com algumas emoções representadas por um número significativamente menor de imagens; por exemplo, a classe “Desgosto” possui apenas 547 imagens, enquanto “Felicidade” apresenta 8.989 imagens. As imagens foram capturadas em condições controladas, com os rostos centralizados e alinhados, facilitando o treinamento de modelos de aprendizado profundo.

Foi também utilizado o conjunto de imagens **KDEF** (*Karolinska Directed Emotional Faces*) (LUNDQVIST; FLYKT; ÖHMAN, 1998) o qual é amplamente utilizado em pesquisas sobre reconhecimento de emoções faciais. Desenvolvido pelo Karolinska Institutet, na Suécia, o KDEF contém um total de 4.900 imagens de expressões faciais humanas, capturadas de 70 indivíduos (35 homens e 35 mulheres), cada um exibindo sete emoções básicas: raiva (anger), medo (fear), desgosto (disgust), tristeza (sadness), felicidade (happiness), surpresa (surprise) e neutro (neutral). Cada emoção foi fotografada em cinco ângulos diferentes: perfil esquerdo completo, meio perfil esquerdo, frontal, meio perfil direito e perfil direito completo. As imagens foram cuida-

[1] <https://datarepository.wolframcloud.com/resources/FER-2013>

dosamente controladas para minimizar variações não emocionais, como maquiagem, acessórios e iluminação, garantindo que as expressões faciais fossem claramente perceptíveis.

O conjunto de dados foi criado com o objetivo de fornecer estímulos consistentes e validados para estudos experimentais sobre percepção emocional, reconhecimento de expressões faciais e interação humano-computador. Desde sua criação, o KDEF tem sido amplamente utilizado em mais de 1.500 publicações científicas, refletindo sua importância e utilidade na comunidade científica. Tal conjunto de imagens, bem como dados relacionados como o **AKDEF** (*Averaged Karolinska Directed Emotional Faces*), podem ser obtidos por meio de seu repositório oficial[2].

Por fim, porém não menos importante, foi também empregado o conjunto de imagens **RAF-DB** (*Real-world Affective Faces Database*) (DENG, 2017) o qual é amplamente utilizado na área de reconhecimento de emoções faciais em cenários do mundo real. Desenvolvido por pesquisadores da Universidade de Pequim, o RAF-DB contém aproximadamente 30.000 imagens faciais coletadas da internet, representando uma ampla variedade de expressões emocionais. As imagens foram rotuladas com sete emoções básicas (raiva, desgosto, medo, felicidade, tristeza, surpresa e neutro) e 12 emoções compostas, totalizando 19 categorias emocionais. No presente trabalho foi utilizada a rotulação considerando as sete emoções básicas, visto que não foi o foco do mesmo análises multi-rótulo.

As imagens do RAF-DB foram capturadas em condições naturais, refletindo a diversidade de idades, gêneros, etnias, posturas faciais, iluminação, oclusões e outras variabilidades presentes em cenários do mundo real. Cada imagem foi rotulada por aproximadamente 40 anotadores independentes, garantindo a precisão e confiabilidade das anotações. O conjunto de dados foi projetado para fornecer estímulos consistentes e validados para estudos experimentais sobre percepção emocional, reconhecimento de expressões faciais e interação humano-computador.

O RAF-DB tem sido amplamente utilizado em pesquisas sobre reconhecimento de emoções faciais, aprendizado profundo e análise de sentimentos, devido à sua diversidade e representatividade das condições do mundo real. O mesmo pode ser obtido por meio de seu repositório oficial[3].

Assim, a validação do método proposto, foi também realizada utilizando os conjuntos de imagens **FER13**, **KDEF** e **RAF-DB**, selecionados em função de suas características complementares e relevância para avaliação de representações contrastivas. O FER13 fornece um grande número de imagens em escala de cinza, distribuídas de forma relativamente equilibrada entre sete emoções básicas, apresentando variabilidade considerável em termos de iluminação, cenários de fundo e pontos de vista. Essa diversidade permite avaliar a capacidade do DGCL de identificar pares positivos e negativos robustos em situações com variações intra-classe significativas e confusões inter-classes sutis.

O KDEF oferece imagens de alta qualidade de 70 indivíduos, capturadas sob ângulos controlados e condições padronizadas, garantindo consistência e confiabilidade nas expressões fa-

[2] <https://kdef.se/>

[3] <https://www.whdeng.cn/RAF/model1.html>

ciais. Esse controle permite testar a capacidade do método de explorar relações geométricas no espaço latente de forma precisa, analisando se o DGCL consegue otimizar embeddings mesmo em cenários com pouca variabilidade externa.

O RAF-DB complementa a avaliação, fornecendo imagens coletadas do mundo real, com ampla diversidade em termos de idade, etnia, postura, iluminação, oclusões e contextos naturais. Esse dataset desafia o DGCL a generalizar para condições menos controladas, explorando efetivamente a seleção de pares baseada em distância em espaços latentes de alta complexidade.

A utilização conjunta destes três conjuntos de dados, agregados ao CIFAR-10, permitiu uma validação abrangente do DGCL, avaliando sua capacidade de aprender representações discriminativas e invariantes, tanto em cenários controlados quanto em situações do mundo real, garantindo robustez e generalização das representações aprendidas.

3.3 Conclusão

Neste capítulo, apresentou-se detalhadamente o método proposto, denominado ***Distance-Guided Contrastive Learning*** **(DGCL)**, que integra extração de características, projeção no espaço latente, seleção de pares positiva e negativa e atualização iterativa do modelo. Foram descritas as políticas de seleção de pares, incluindo estratégias fixas e adaptativas baseadas em densidade e percentis, bem como a formulação da função de perda contrastiva utilizada para otimizar as representações. A implementação do método foi detalhada passo a passo, destacando o uso de projeções t-SNE para estimar a geometria latente e orientar a escolha de pares de forma eficiente, mantendo estabilidade computacional.

Além disso, justificou-se a escolha dos conjuntos de imagens **CIFAR-10**, **FER13**, **KDEF** e **RAF-DB** para a validação do DGCL, considerando suas características complementares. O CIFAR-10 permitiu avaliar o método em um cenário de imagens naturais com diversidade significativa e classes equilibradas, possibilitando testes iniciais de robustez do aprendizado contrastivo. O FER13 permitiu avaliar a capacidade do método diante de variações intra-classe e confusões inter-classes, o KDEF forneceu cenários controlados que possibilitam análise precisa das relações geométricas no espaço latente, enquanto o RAF-DB introduziu diversidade do mundo real, desafiando a generalização das representações aprendidas.

Dessa forma, o capítulo consolidou a base metodológica e experimental do trabalho, preparando o terreno para a apresentação dos resultados e discussões subsequentes, evidenciando a capacidade do DGCL em aprender representações discriminativas e invariantes para diferentes condições e complexidades visuais.

Capítulo 4

Experimentos e resultados

Este capítulo apresenta e discute os resultados obtidos com o método proposto, bem como sua comparação com abordagem sem o componente contrastivo. São analisados os impactos da seleção guiada por distância na estrutura do espaço latente, na acurácia de classificação e na coerência das representações. Além dos resultados quantitativos, são incluídas análises qualitativas com base em projeções t-SNE e matrizes de confusão, conforme as figuras e tabelas da dissertação original. Foi também analisada a otimização da função de perda em relação ao número de âncoras utilizado no processo de treinamento.

4.1 Descrição dos experimentos

Os experimentos foram conduzidos com os seguintes propósitos principais:

1. Avaliar o impacto do aprendizado contrastivo guiado por distância na formação das representações visuais;
2. Comparar o desempenho do DGCL com um modelo sem aprendizado contrastivo (baseline supervisionado convencional);
3. Investigar o comportamento do modelo em diferentes datasets com características distintas, variando entre domínios naturais (CIFAR-10) e expressões faciais (FER-13, KDEF, RAF-DB);
4. Analisar a evolução das representações latentes e a separabilidade entre classes ao longo do treinamento, tanto em projeções t-SNE quanto nas matrizes de confusão;
5. Analisar a convergência do método de acordo com a quantidade de âncoras utilizadas no processo de treinamento.

Como métrica de avaliação dos resultados obtidos foi calculada a acurácia (proporção de amostras corretamente classificadas). Entretanto, uma vez que são também apresentadas as matrizes de confusão dos experimentos realizados, outras métricas podem ser inferidas a partir das mesmas (e.g. precisão, revocação e F1-score). Além disso, as matrizes de confusão foram analisadas visualmente para compreender os padrões de erro e a coerência das fronteiras de decisão aprendidas pelos modelos.

4.1.1 Configuração Experimental

Os experimentos foram realizados utilizando a arquitetura de base (*backbone*) ResNet50 tradicional, bem como *encoder* do método proposto. Inicialmente, foi aplicado *transfer learning* com os pesos pré-treinados a partir do conjunto ImageNet.

A escolha pela utilização exclusiva da arquitetura ResNet50 fundamenta-se em seu equilíbrio entre profundidade, desempenho e custo computacional. Essa rede convolucional residual apresenta uma estrutura suficientemente profunda para capturar características visuais complexas, ao mesmo tempo em que mantém estabilidade no treinamento graças ao uso de conexões de atalho (*skip connections*), que mitigam o problema do *vanishing gradients* em arquiteturas mais profundas. Além disso, a ResNet50 é amplamente adotada na literatura de aprendizado contrastivo, servindo como referência em métodos como SimCLR, MoCo e BYOL, o que favorece a comparabilidade direta dos resultados obtidos pelo método proposto (DGCL) com trabalhos correlatos.

Não foram conduzidos experimentos com outras arquiteturas (e.g. ResNet18, DenseNet, EfficientNet, dentre outras) por duas razões principais. Primeiramente, buscou-se reduzir o impacto de variáveis externas à análise do método proposto, de modo que as diferenças observadas nos resultados pudessem ser atribuídas de forma inequívoca à política de seleção guiada por distância e não a variações arquiteturais. Em segundo lugar, o escopo deste trabalho priorizou a avaliação do DGCL como framework metodológico, e não a otimização de desempenho por meio da troca de modelos de base. Assim, a utilização de uma única arquitetura consolidada, amplamente reconhecida e de comportamento previsível, assegura maior rigor experimental e validade comparativa dos resultados apresentados.

A implementação utilizou a bilioteca PyTorch, e foram definidos o seguinte otimizador e hiperparâmetros: otimizador Adam, tamanho de lote (batch size) 128, taxa de aprendizado inicial de 10^{-3} e treinamento por 6 épocas. As projeções do t-SNE foram recalculadas a cada época. Nos experimentos relatados, foram definidos valores fixos de $k_p = k_n = 5$; para garantir justiça na avaliação, foram utilizados hiperparâmetros idênticos para todas as arquiteturas.

A escolha por realizar os experimentos com apenas 6 épocas de treinamento fundamenta-se em critérios de análise metodológica e eficiência experimental. O objetivo principal deste trabalho não foi alcançar o melhor desempenho absoluto possível, mas sim avaliar o comportamento do método proposto (DGCL) e seus efeitos sobre a estruturação geométrica do espaço latente ao longo do processo de aprendizado.

Durante as iterações iniciais de treinamento, observou-se que o modelo apresentava rápida convergência da função de perda e estabilização das métricas de acurácia a partir da quinta época, indicando que o método atinge equilíbrio geométrico em poucas passagens pelo conjunto de dados. Essa característica decorre do próprio mecanismo de retroalimentação geométrica do DGCL, que seleciona pares informativos de maneira adaptativa, tornando o aprendizado mais eficiente e menos dependente de longos ciclos de otimização.

Além disso, a adoção de um número reduzido de épocas visou reduzir o custo computacional dos experimentos e possibilitar a execução de múltiplas repetições em diferentes conjuntos de dados dentro de um tempo experimental viável, sem comprometer a validade das conclusões. Assim, a opção por seis épocas representa um equilíbrio entre eficiência e evidência empírica, sendo suficiente para demonstrar o comportamento dinâmico e a estabilidade do método proposto.

Os experimentos foram conduzidos em uma máquina do modelo Lenovo IdeaPad Gaming 3 15ACH6, equipada com 16 GB de memória RAM, processador AMD Ryzen 5 5600H com GPU integrada Radeon, placa gráfica dedicada NVIDIA GeForce GTX 1650, sistema operacional Ubuntu 24.04.3 LTS e kernel Linux versão 6.14.0-29-generic.

Para a análise e manipulação dos dados, foi utilizado um ambiente Jupyter Notebook, que permite a integração de código, texto explicativo e saídas de execução em um mesmo documento no formato JavaScript Object Notation (JSON). Essa ferramenta é amplamente adotada em projetos de ciência de dados e aprendizado de máquina por possibilitar a visualização imediata dos resultados, favorecendo um fluxo de trabalho interativo e reprodutível.

A instalação das bibliotecas Python foi realizada por meio do gerenciador de pacotes e ambientes Conda, amplamente utilizado em projetos de ciência de dados e aprendizado de máquina devido à sua capacidade de lidar com dependências complexas e múltiplas versões de pacotes e do próprio interpretador Python. Esse gerenciador também facilita o empacotamento e a implantação de aplicações, garantindo consistência entre ambientes de execução.

O ambiente experimental foi configurado com as seguintes versões das principais bibliotecas utilizadas: Python 3.9.21, Numpy 1.24.3, Torch 2.6.0, TensorFlow 2.15.0, Keras 2.15.0, Matplotlib 3.7.2, Scikit-learn 1.6.1, Pandas 2.2.3 e Torchvision 0.21.0.

4.2 Resultados

Os resultados gerais de acurácia obtidos para os quatro conjuntos de dados encontram-se na Tabela 1 da dissertação. Nela, são comparados os valores obtidos pelos modelos com e sem aprendizado contrastivo, evidenciando os ganhos proporcionados pela seleção guiada por distância.

A Tabela 1 apresenta a acurácia de classificação obtida utilizando uma arquitetura base ResNet50 treinada (i) sem aprendizado contrastivo e (ii) com a estratégia proposta de *Distance-Guided Contrastive Learning* (DGCL), considerando os conjuntos de imagens descritos na Se-

Tabela 1 – Acurácia dos datasets CIFAR10, FER13, RAF-DB e KDEF aplicando o método proposto, bem como sem a aplicação de aprendizado por contraste.

Datasets	**Contrastivo**	**Sem contraste**
CIFAR10	70,69	34,06
FER13	32,81	16,68
KDEF	42,15	26,79
RAF-DB	50,95	41,49

Fonte: Autor

ção 3.2. Analisando os resultados obtidos empregando o conjunto de imagens CIFAR-10, os mesmos revelam uma melhoria expressiva de 34,03% para 70,69% na acurácia, demonstrando que a incorporação de um mecanismo contrastivo guiado geometricamente mais do que dobra a capacidade discriminativa do modelo. Esse ganho substancial evidencia que a seleção guiada por distância proposta aprimora de forma consistente a qualidade das representações aprendidas em diferentes capacidades de rede, confirmando que o DGCL reestrutura efetivamente o espaço latente de características em uma organização semanticamente mais coerente.

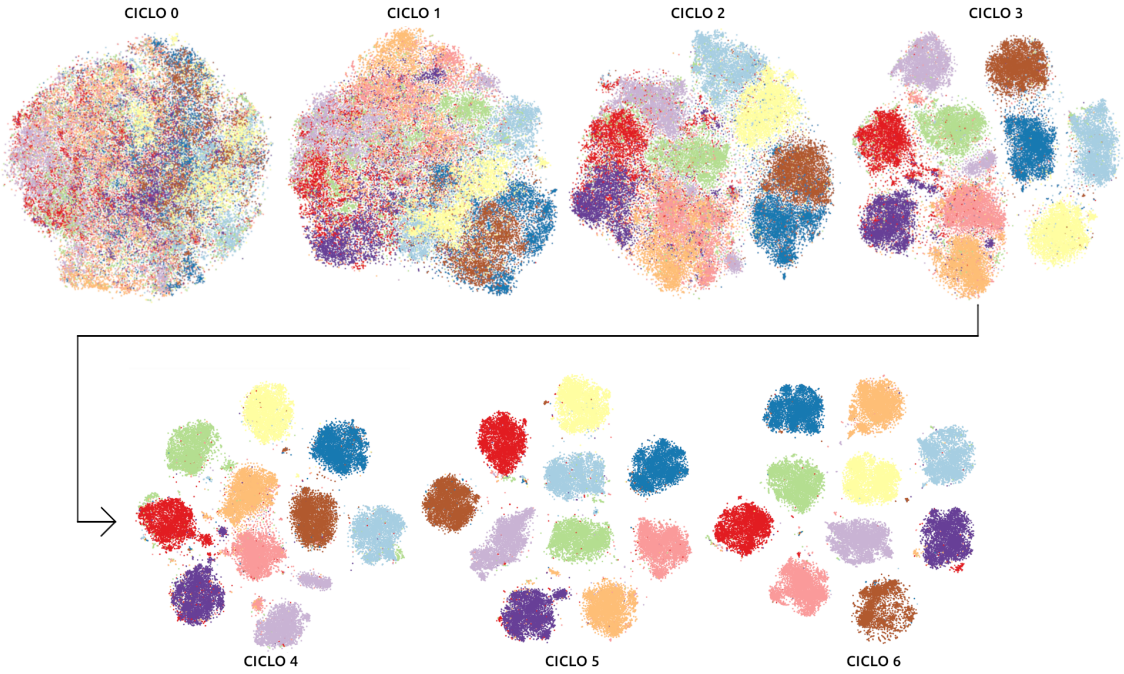
A principal razão para essa melhoria reside na política contrastiva que seleciona explicitamente pares informativos para o treinamento. No DGCL, cada imagem atua como uma âncora, e as amostras intra-classes mais distantes (*hard positives*) e inter-classes mais próximas (*hard negatives*) são dinamicamente identificadas no manifold latente obtido via t-SNE. Ao aproximar os positivos distantes, o método reduz a variância intra-classe, enquanto ao afastar os negativos próximos, amplia as margens inter-classe. Essa seleção dinâmica concentra as atualizações de gradiente nas regiões mais informativas do espaço de características, acelerando a convergência e resultando em embeddings altamente discriminativos.

Do ponto de vista conceitual, o modelo treinado sem aprendizado contrastivo opera exclusivamente sob restrições de perda supervisionada, que dependem das informações de rótulo explícitas para separar as classes. Essa configuração frequentemente produz embeddings apenas parcialmente estruturados, com generalização limitada e baixa separabilidade entre categorias visualmente semelhantes. Em contraste, o DGCL introduz uma fase de pré-treinamento autossupervisionado sensível à geometria, que otimiza explicitamente a estrutura espacial do espaço de embeddings. Ao alinhar continuamente amostras semanticamente relacionadas e repelir as dissimilares, o codificador (encoder) aprende uma representação mais desvinculada e semanticamente fiel do manifold visual, antes mesmo que o classificador seja treinado.

A magnitude da melhoria, um aumento absoluto de 36,66 pontos percentuais, mostra que o pré-treinamento contrastivo não apenas aprimora a otimização, mas também reorganiza fundamentalmente o espaço de representações aprendidas. As visualizações em t-SNE na Figura 8 confirmam qualitativamente esse comportamento: os embeddings produzidos pelo DGCL formam agrupamentos compactos com fronteiras inter-classes suaves, enquanto o modelo base

gera estruturas difusas e sobrepostas. Inicialmente, os embeddings aparecem altamente entrelaçados, com separação mínima entre classes; à medida que os ciclos do DGCL progridem, o espaço evolui para uma configuração caracterizada por agrupamentos semânticos bem definidos e redução de ambiguidades. Por outro lado, as visualizações sem o aprendizado por contraste (ver Figura 9) a sobreposição entre classes perdura durante praticamente todos os ciclos.

Figura 8 – Evolução das representações do CIFAR-10 ao longo dos ciclos de aprendizagem do modelo contrastivo.

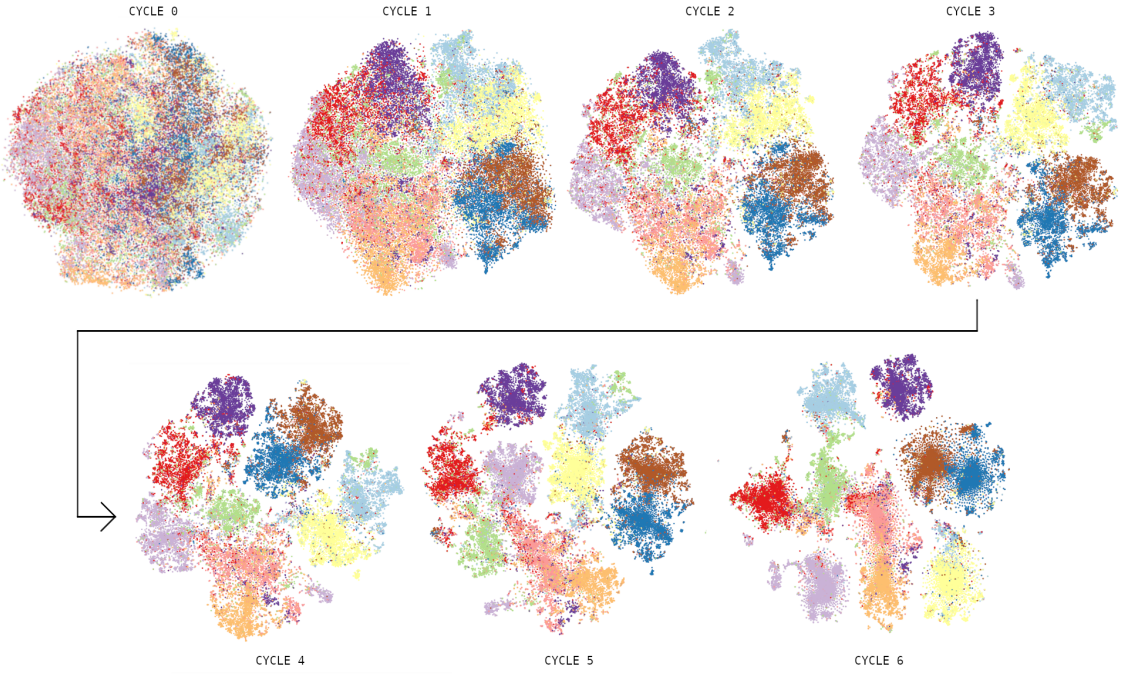


Fonte: Autor

Sob uma perspectiva geométrica, o DGCL reduz sistematicamente a dispersão intra-classe enquanto expande as distâncias inter-classe. Seja σ^2_{intra} a variância média intra-classe e S_{inter} a separação média entre centróides de classes. Evidências empíricas indicam que o DGCL minimiza significativamente a razão $\frac{\sigma^2_{\text{intra}}}{S_{\text{inter}}}$ em comparação com o modelo base, confirmando que as representações aprendidas são mais compactas e discriminativas. Em termos práticos, amostras de uma mesma classe convergem para regiões próximas do manifold, enquanto aquelas de classes diferentes são repelidas para zonas distintas no espaço de embeddings. Essa reestruturação traduz-se diretamente em fronteiras de decisão mais estáveis e em maior confiança nas classificações.

A política de amostragem guiada por distância também contribui para essa melhoria ao concentrar-se em exemplos semanticamente desafiadores. Ao selecionar *hard positives* e *hard negatives* de acordo com sua relação geométrica no manifold latente, o DGCL concentra o esforço de aprendizado em regiões de alta ambiguidade. Essa otimização direcionada acelera

Figura 9 – Evolução das representações do CIFAR-10 ao longo dos ciclos de aprendizagem do modelo sem contraste.



Fonte: Autor

o refinamento das fronteiras entre classes e aprimora a robustez das características aprendidas (efeitos claramente visíveis na progressão das visualizações em t-SNE ao longo dos ciclos).

Os hiperparâmetros que controlam o número de pares positivos e negativos por âncora, denotados por k_p e k_n, foram definidos como $k_p = k_n = k^\star = 5$ em todos os experimentos, obtendo um equilíbrio entre diversidade de gradientes e informatividade do contraste. Valores extremamente baixos de k aumentam a variância do gradiente, enquanto valores excessivamente altos diluem a importância dos exemplos difíceis e se aproximam de um comportamento de amostragem aleatória. A análise empírica mostra desempenho estável para $k \in [3,8]$, com degradação notável além desse intervalo. Trabalhos futuros explorarão definições adaptativas de $k_p(i)$ e $k_n(i)$ com base na densidade local ou em limiares percentuais, a fim de ajustar dinamicamente a dificuldade da seleção durante o treinamento.

Em relação ao custo computacional, o DGCL introduz uma sobrecarga moderada devido à projeção t-SNE e à seleção de pares baseada em distância. No entanto, como o t-SNE pode ser recalculado periodicamente em vez de a cada iteração, o custo permanece administrável para conjuntos de dados de tamanho médio, como o CIFAR-10. Para cenários em maior escala, podem ser empregadas buscas aproximadas por vizinhos mais próximos ou métodos de projeção de manifold mais rápidos, como o UMAP, preservando o mesmo princípio guiado por distância. Os ganhos observados em acurácia superam amplamente o custo adicional, confirmando uma relação favorável entre eficiência e qualidade representacional.

Em resumo, as evidências experimentais demonstram que o DGCL aprimora de forma consistente o aprendizado de representações ao explorar relações geométricas entre amostras de maneira fundamentada. O arcabouço produz embeddings semanticamente alinhados, geometricamente estruturados e altamente separáveis, resultando em melhorias substanciais no desempenho de classificação. O salto de acurácia de 34,03% para 70,69% valida a efetividade do método proposto e posiciona o DGCL como um paradigma robusto e generalizável para aprendizado autossupervisionado sensível à geometria em visão computacional.

A Figura 10 ilustra a evolução da perda por âncora ao longo de diferentes épocas de treinamento, em função do número de âncoras processadas durante a otimização contrastiva. Cada curva corresponde a uma época, variando da Época 1 até a Época 6. O eixo y representa o valor da perda contrastiva, enquanto o eixo x denota o número de âncoras consideradas em cada época, até o tamanho total do conjunto de dados de 50.000 amostras de treinamento para o CIFAR-10. Essa visualização fornece uma perspectiva detalhada sobre a dinâmica de convergência da abordagem DGCL proposta.

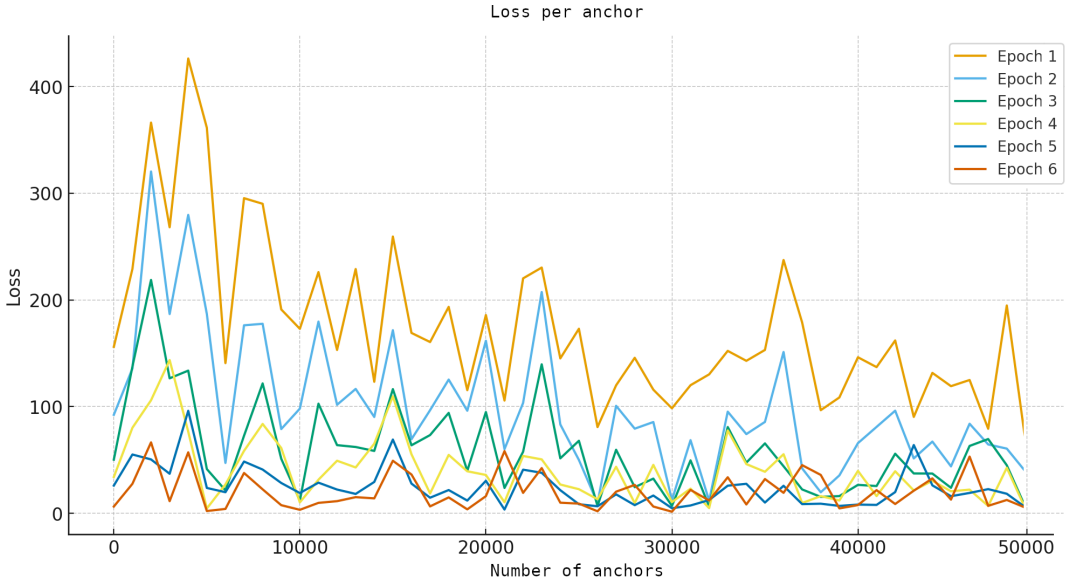


Figura 10 – Perda por âncora em função do número de âncoras utilizadas durante o treinamento, para as épocas 1 a 6. Os resultados demonstram que o DGCL converge de maneira eficiente: na época 5, a perda obtida utilizando 10k âncoras é quase idêntica à obtida com as 50k âncoras completas, evidenciando que o modelo aprende representações estáveis e discriminativas com um número substancialmente menor de amostras.

No início do treinamento (Épocas 1–2), os valores de perda são altos e exibem grande variância, refletindo o alinhamento instável dos embeddings nas etapas iniciais da otimização. O objetivo contrastivo, nesse ponto, opera sobre representações pouco estruturadas, resultando em grandes discrepâncias entre as distâncias de pares positivos e negativos. À medida que o modelo progride através das épocas subsequentes, tanto a magnitude quanto a variância da perda diminuem substancialmente. Isso indica que o codificador organiza progressivamente o espaço latente, melhorando a compacidade intra-classe e a separação inter-classe.

Uma observação chave da Figura 10 é que, nas Épocas 5 e 6, a curva de perda estabiliza-se em valores baixos mesmo quando o número de âncoras utilizadas é significativamente menor do que o tamanho total do conjunto de dados. Especificamente, quando aproximadamente $N_a \approx 10.000$ âncoras são amostradas (apenas 20% do conjunto de treinamento completo), a perda converge para valores quase idênticos aos obtidos ao usar todas as 50.000 âncoras. Esse comportamento demonstra que a abordagem DGCL atinge convergência representacional com um número substancialmente menor de âncoras contrastivas do que o total de amostras de treinamento.

Formalmente, seja $\mathcal{L}_e(N_a)$ a perda média na época e computada sobre N_a âncoras. Define-se a convergência em relação à perda do conjunto completo $\mathcal{L}_e^* = \mathcal{L}_e(50.000)$ como ocorrendo quando

$$\left| \frac{\mathcal{L}_e(N_a) - \mathcal{L}_e^*}{\mathcal{L}_e^*} \right| \leq \varepsilon,$$

onde ε é um pequeno limiar de tolerância. Empiricamente, para $e \geq 5$ e $N_a \geq 10.000$, o desvio relativo fica abaixo de $\varepsilon = 0,05$, isto é,

$$\mathcal{L}_5(10.000) \approx \mathcal{L}_5(50.000),$$

$$\mathcal{L}_6(10.000) \approx \mathcal{L}_6(50.000),$$

confirmando que o modelo atinge uma representação em estado estacionário bem antes de esgotar todo o conjunto de âncoras.

Esse comportamento de convergência indica que o DGCL aproveita sua estratégia de amostragem guiada por distância de maneira eficiente, priorizando âncoras informativas nas fases iniciais do treinamento e evitando comparações redundantes de pares. Como consequência, o objetivo contrastivo satura rapidamente uma vez que a estrutura do manifold está suficientemente refinada. Do ponto de vista da otimização, essa propriedade implica que a complexidade amostral efetiva do DGCL é menor do que a dos métodos de aprendizado contrastivo uniforme, que exigem a varredura completa do conjunto de dados para alcançar estabilidade semelhante.

Em termos práticos, esses resultados sugerem que o DGCL pode alcançar acurácia quase ótima utilizando apenas um subconjunto das âncoras disponíveis por ciclo, resultando em economias computacionais significativas. Seja $T(N_a, e)$ o custo cumulativo de treinamento até a época e com N_a âncoras. Como T escala aproximadamente de forma linear com N_a, reduzir as âncoras de 50.000 para 10.000 corresponde a uma diminuição de 80% no custo, preservando convergência e acurácia comparáveis. Formalmente,

$$T(10.000, e) \approx 0,2\,T(50.000, e),$$

$$\text{com } \mathcal{A}(10.000, e) \approx \mathcal{A}(50.000, e),$$

onde $\mathcal{A}$ denota a acurácia alcançada. Esse achado reforça que o DGCL identifica de forma eficiente o subconjunto mais informativo de âncoras, convergindo tanto em perda quanto em desempenho com um número significativamente menor de iterações.

Em resumo, as trajetórias de perda na Figura 10 demonstram claramente que o DGCL alcança convergência rápida, estabilizando-se por volta da Época 5 com apenas 10k âncoras. Esse comportamento evidencia a capacidade do método de concentrar o aprendizado nas regiões mais relevantes do manifold de dados, garantindo alta acurácia e qualidade robusta de representação com sobrecarga computacional reduzida.

Considerando os outros conjuntos de imagens os mesmos comportamentos foram observados em relação ao ganho de acurácia, separabilidade das classes nos diferentes ciclos, bem como com relação à convergência com uma menor quantidade de âncoras quando comparada à cardinalidade do conjunto de treinamento.

Analisando a Tabela 1 a qual resume a acurácia alcançada pelos modelos com e sem o componente contrastivo, considerando os outros conjuntos de imagens, pode-se verificar que o método proposto apresenta ganhos absolutos (em pontos percentuais - p.p.) e ganhos relativos significativos, especialmente nos conjuntos de reconhecimento de expressões faciais. A seguir apresenta-se uma análise detalhada por conjunto.

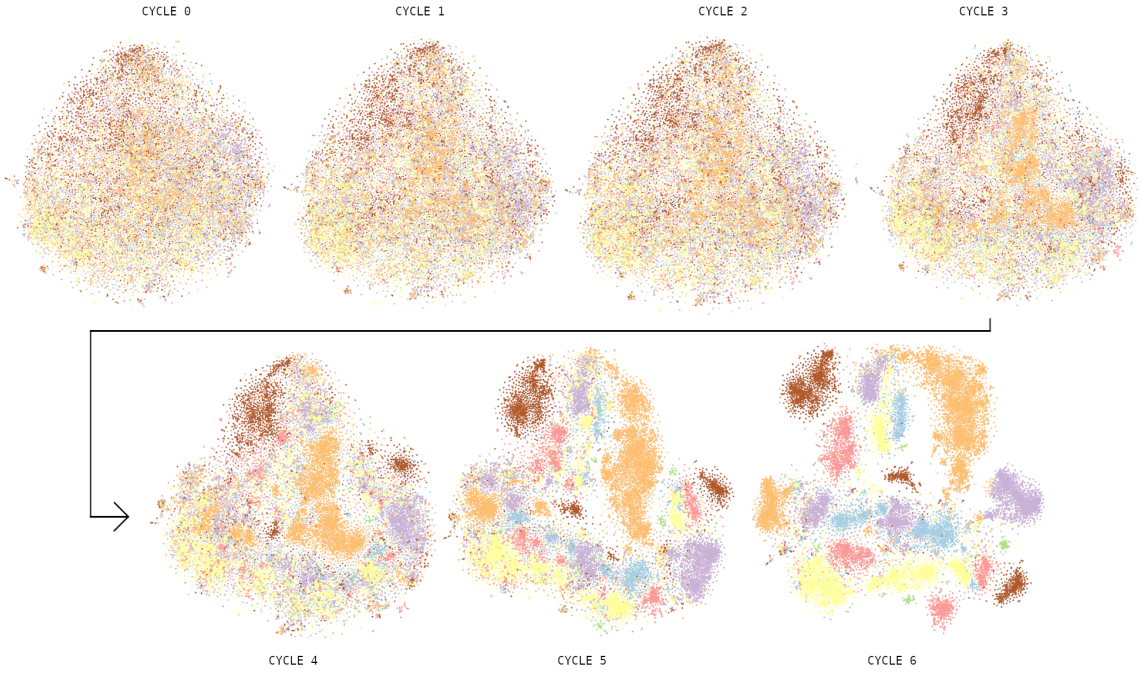
Considerando o conjunto de imagens FER-13, o modelo sem contraste obteve acurácia de 16.68%, enquanto o modelo com DGCL alcançou 32.81%. O ganho absoluto foi de 16.13 pontos percentuais. Em termos relativos, esse incremento corresponde a um aumento de aproximadamente 96.70% sobre a acurácia do baseline, ou seja, quase o dobro do desempenho inicial. Esse ganho expressivo pode ser atribuído à capacidade do DGCL de compactar amostras intra-classe e separar melhor classes com fronteiras sutis.

No conjunto KDEF, a acurácia do baseline foi 26.79% e subiu para 42.15% com DGCL. O ganho absoluto foi de 15.36, representando um aumento relativo de aproximadamente 57.33%. KDEF é um dataset controlado (imagens padronizadas por sujeito e ângulo), portanto melhorias deste porte indicam que o DGCL explora com sucesso sutis variações morfológicas de expressão, traduzindo-se em ganhos robustos mesmo em cenário menos ruidoso.

Para o conjunto RAF-DB (imagens do mundo real com grande variabilidade), o baseline apresentou acurácia de 41.49% e o DGCL alcançou 50.95%. O ganho absoluto foi de 9.4 pontos percentuais, o que equivale a um aumento relativo de cerca de 22.80%. Embora o ganho absoluto e relativo sejam menores que nos datasets anteriores, eles ainda são relevantes diante da maior complexidade e ruído do RAF-DB, e melhora a identificação de classes minoritárias, conforme evidenciado pelas matrizes de confusão contrastivas (ver Seção 4.2.1).

Em termos práticos, os aumentos em pontos percentuais (FER-13: 16.13 p.p.; KDEF: 15.36 p.p.; RAF-DB: 9.46 p.p.) mostram que a seleção guiada por distância favorece recuperação de exemplos difíceis e acelera a construção de fronteiras de decisão mais nítidas. Em conclusão, a estratégia contrastiva proposta proporciona ganhos substanciais e consistentes em todos os datasets analisados, com impacto especialmente pronunciado em conjuntos de reconhecimento

Figura 11 – Evolução das representações do FER-13 ao longo dos ciclos de aprendizagem do modelo contrastivo.



Fonte: Autor

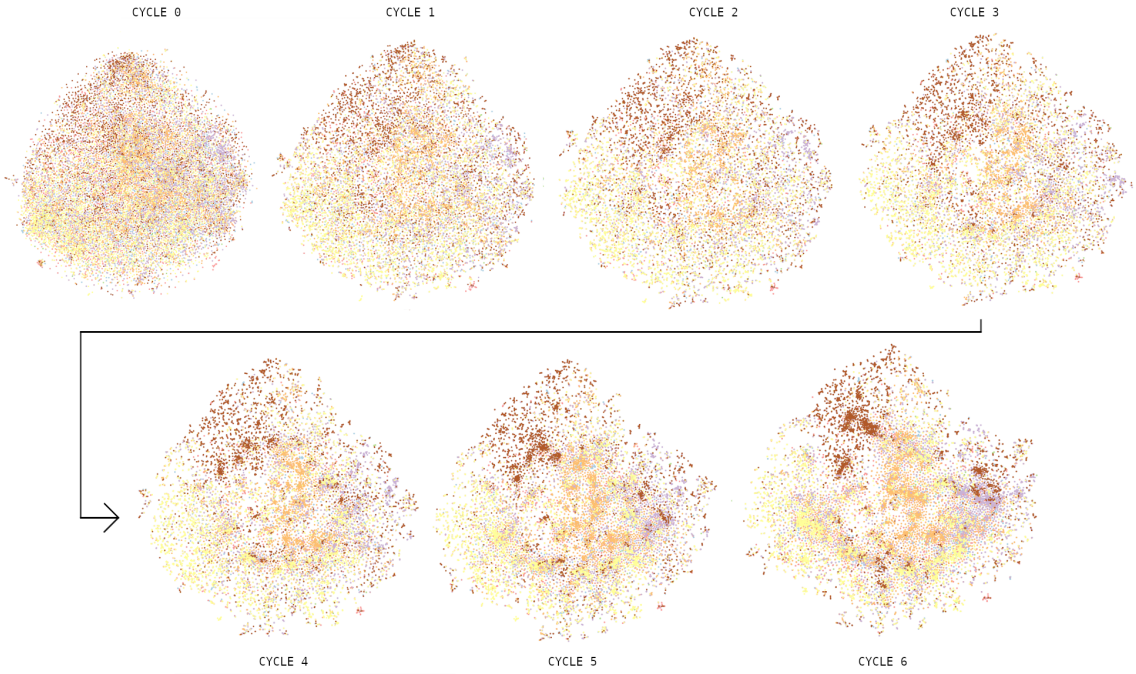
facial, onde as nuances visuais beneficiam-se fortemente da seleção de pares informativos baseada na geometria do espaço latente.

Com relação à separabilidade das classes ao analisar as projeções geradas pelo t-SNE em diferentes ciclos, com e sem a aplicação do método proposto (ver Figuras 11 a 16) pode-se observar que para todos os conjuntos de dados considerados o método apresenta maior diferenciação inter-classe e menor variância intra-classe, com agrupamentos consideravelmente melhor definidos e coesos.

Por fim, analisando a função de perda em relação ao comportamento de convergência ao aplicar o método proposto, considerando os conjuntos FER-13, KDEF e RAF-DB, observou-se comportamento similar ao obtido a partir do conjunto de imagens CIFAR-10. As Figuras 17 a 19 ilustram, respectivamente, a a evolução da perda por âncora ao longo de diferentes épocas de treinamento, em função do número de âncoras processadas durante a otimização contrastiva. Para o FER-13, ao aplicar o método proposto, a perda obtida utilizando 8k âncoras, a partir da época 5, é similar à obtida ao empregar todo o dataset de treinamento (28.709 âncoras). Comportamento análogo foi obtido para os conjuntos KDEF e RAF-DB, nos quais a utilização de aproximadamente 1k e 4k âncoras, respectivamente, considerando a partir da época 5 obtiveram perda equivalente ao utilizar os conjuntos de âncoras completos (2k e 12k).

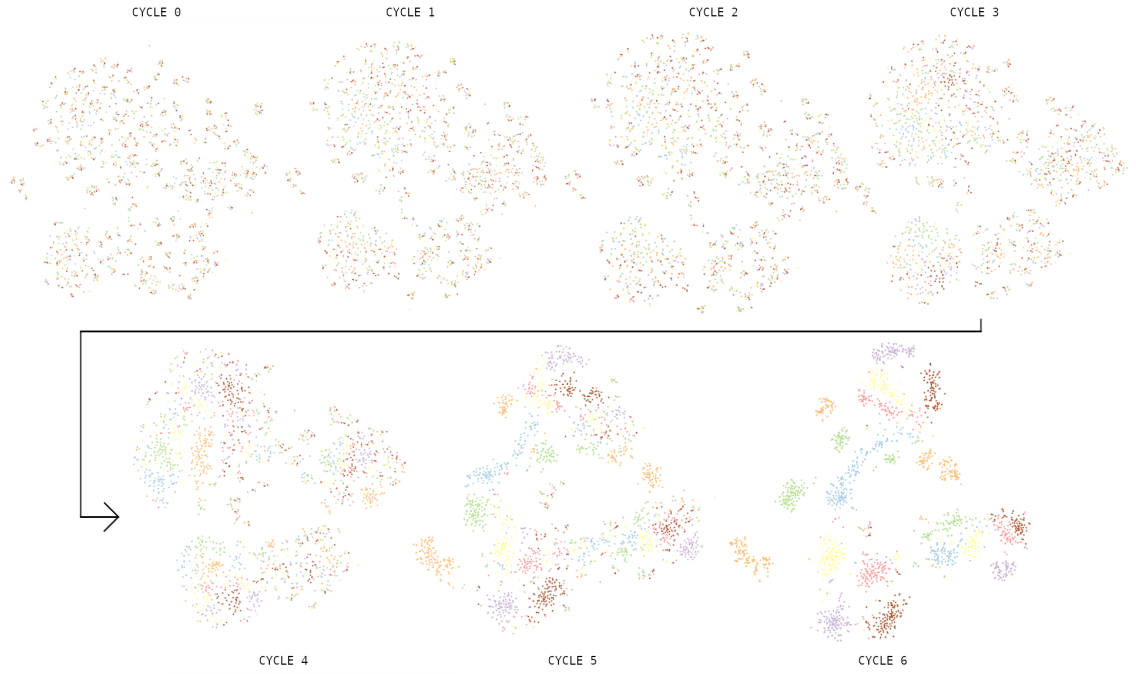
Esse comportamento evidencia que o mecanismo de seleção guiada por distância é eficiente: o modelo converge mais rapidamente e com menos exemplos informativos do que abordagens

Figura 12 – Evolução das representações do FER-13 ao longo dos ciclos de aprendizagem do modelo sem contraste.



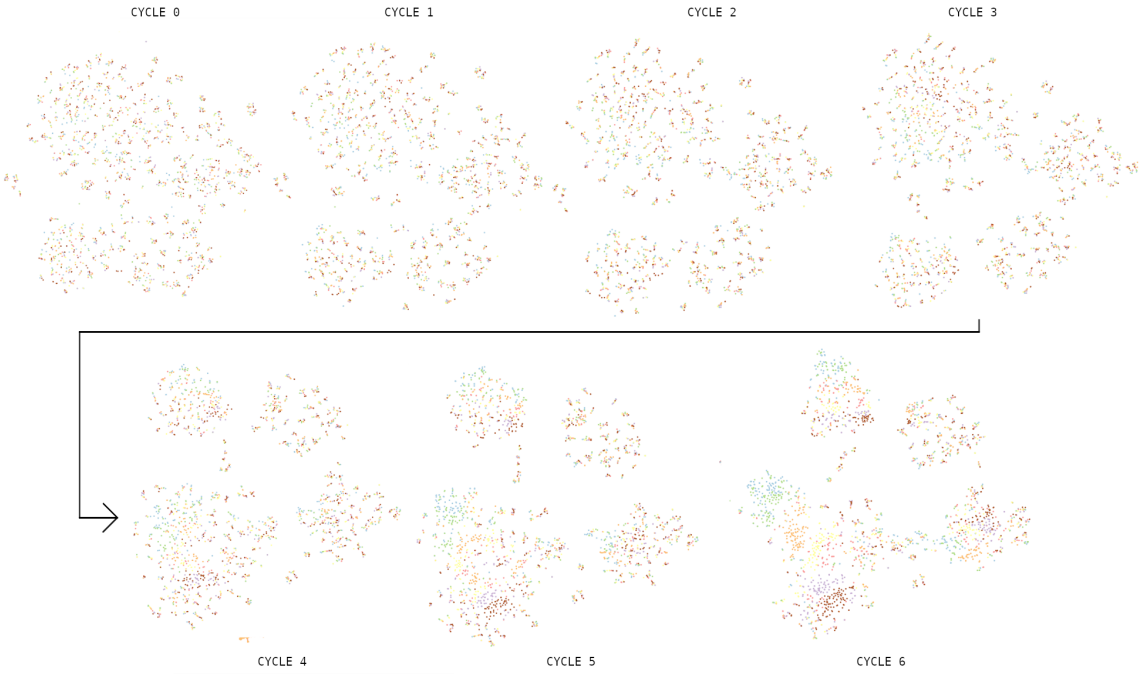
Fonte: Autor

Figura 13 – Evolução das representações do KDEF ao longo dos ciclos de aprendizagem do modelo contrastivo.



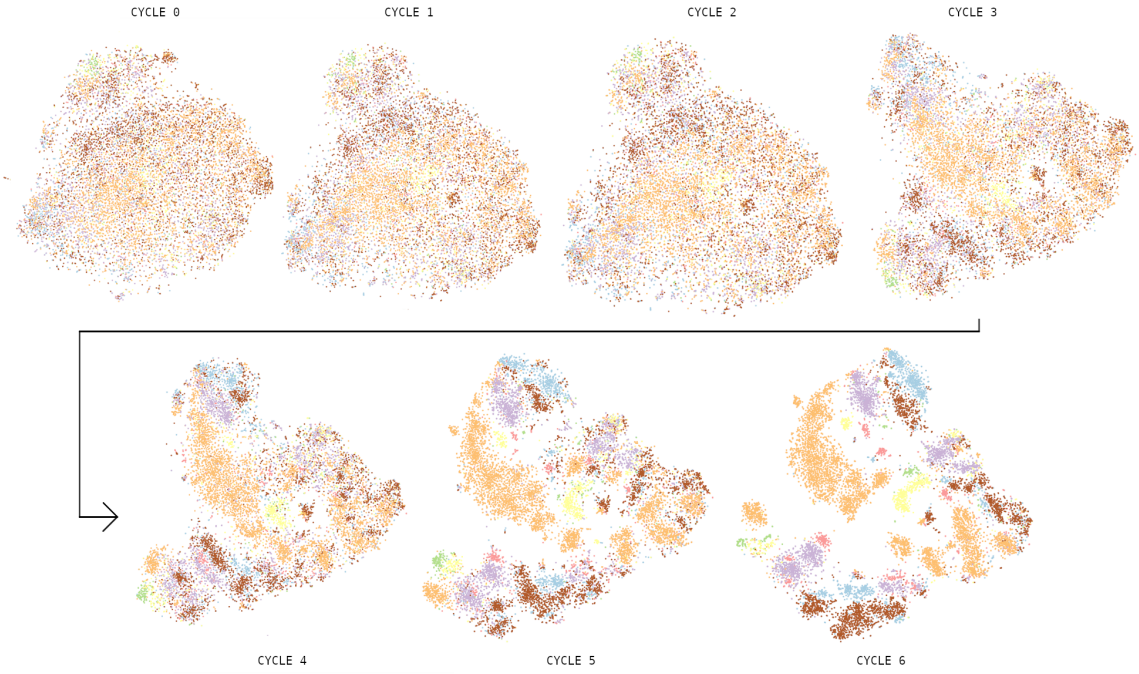
Fonte: Autor

Figura 14 – Evolução das representações do KDEF ao longo dos ciclos de aprendizagem do modelo sem contraste.



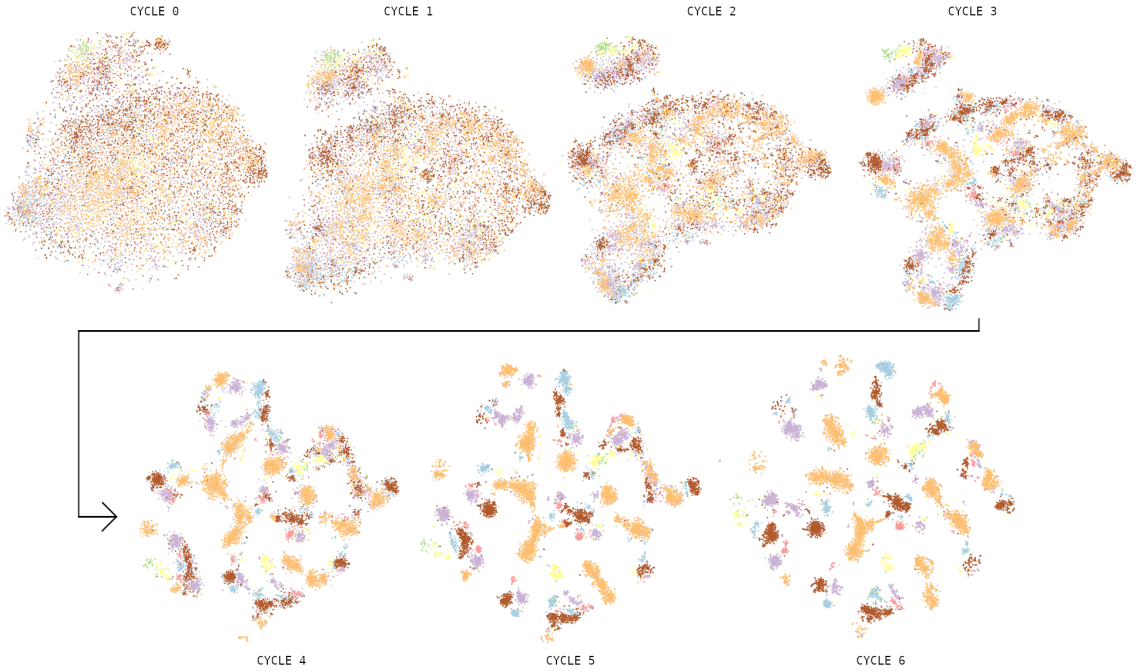
Fonte: Autor

Figura 15 – Evolução das representações do RAF-DB ao longo dos ciclos de aprendizagem do modelo contrastivo.



Fonte: Autor

Figura 16 – Evolução das representações do RAF-DB ao longo dos ciclos de aprendizagem do modelo sem contraste.



Fonte: Autor

Figura 17 – Perda por âncora em função do número de âncoras utilizadas durante o treinamento, para as épocas 1 a 6. Os resultados demonstram que o DGCL converge de maneira eficiente: na época 5, a perda obtida utilizando 8k âncoras é quase idêntica à obtida com as 28.709 âncoras completas, evidenciando que o modelo aprende representações estáveis e discriminativas com um número substancialmente menor de amostras.

contrastivas convencionais, que requerem lotes muito maiores. A diminuição progressiva da perda ao longo das épocas também confirma a estabilidade do processo de retroalimentação ge-

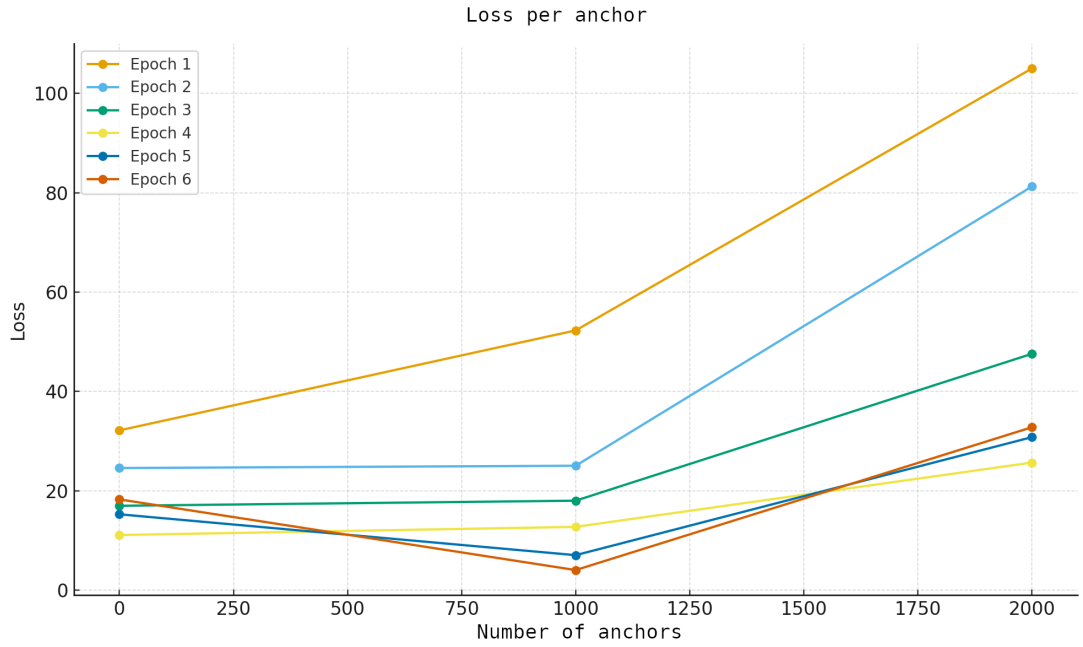


Figura 18 – Perda por âncora em função do número de âncoras utilizadas durante o treinamento, para as épocas 1 a 6. Os resultados demonstram que o DGCL converge de maneira eficiente: na época 5, a perda obtida utilizando 1k âncoras é quase idêntica à obtida com as 2k âncoras completas, evidenciando que o modelo aprende representações estáveis e discriminativas com um número substancialmente menor de amostras.

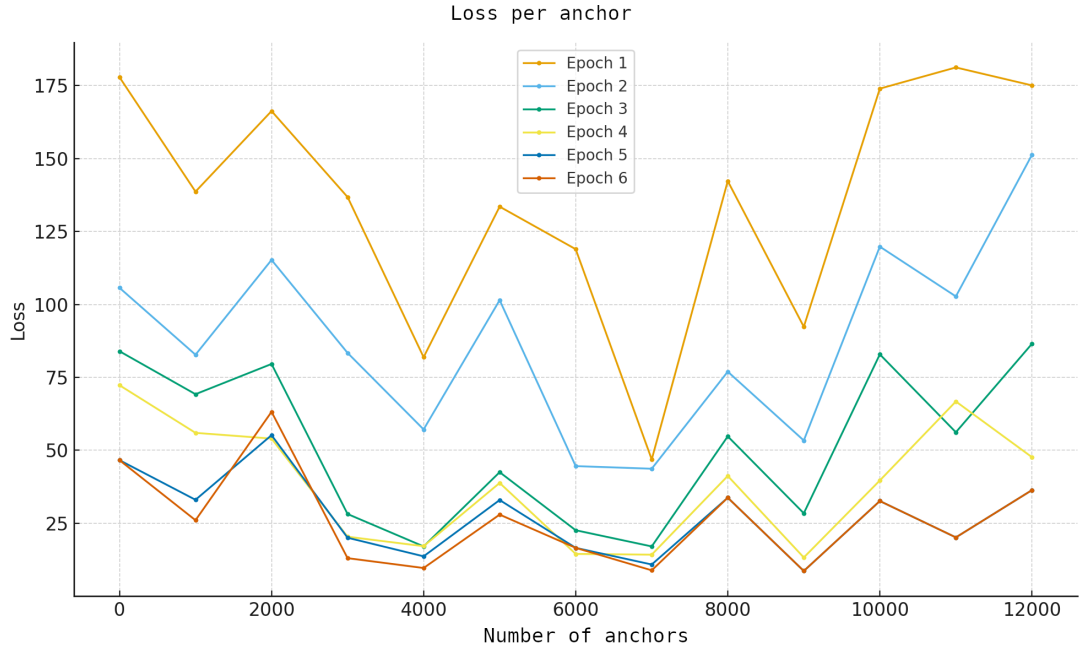


Figura 19 – Perda por âncora em função do número de âncoras utilizadas durante o treinamento, para as épocas 1 a 6. Os resultados demonstram que o DGCL converge de maneira eficiente: na época 5, a perda obtida utilizando 4k âncoras é quase idêntica à obtida com as 12k âncoras completas, evidenciando que o modelo aprende representações estáveis e discriminativas com um número substancialmente menor de amostras.

ométrica, mesmo com o t-SNE sendo recalculado a cada 5 épocas, o sistema mantém gradientes

suaves e consistentes.

4.2.1 Análise das Matrizes de Confusão

Com o intuito de compreender mais profundamente o comportamento dos modelos, foram analisadas as matrizes de confusão obtidas nos quatro conjuntos de dados, CIFAR-10, FER-13, KDEF e RAF-DB, tanto para o modelo base (*sem contraste*) quanto para o modelo proposto DGCL (*com contraste*). Essa análise permite identificar os padrões de erro e a coerência das fronteiras de decisão aprendidas, bem como avaliar qualitativamente os ganhos obtidos pelo aprendizado contrastivo guiado por distância.

As Figuras 20 e 21 apresentam as matrizes de confusão correspondentes ao conjunto CIFAR-10.

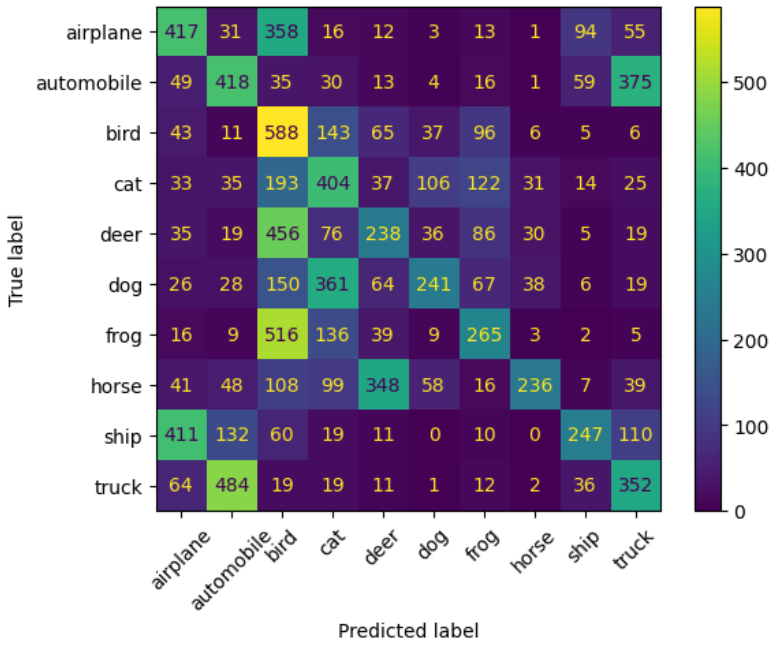


Figura 20 – Matriz de confusão do CIFAR-10 para o modelo sem contraste.

Observa-se que, no modelo sem contraste, há confusões recorrentes entre as classes *airplane*, *bird* e *ship*, bem como entre *cat* e *dog*, que apresentam similaridades visuais. Após a aplicação do método DGCL, tais ambiguidades são substancialmente reduzidas. A diagonal principal torna-se mais pronunciada, indicando maior precisão na classificação e melhor separabilidade inter-classe. A classe *truck*, por exemplo, apresentou expressiva redução de erros com *automobile*, demonstrando que o modelo passou a distinguir de forma mais clara padrões sutis de textura e forma.

As Figuras 22 e 23 exibem as matrizes de confusão para o conjunto FER-13, voltado ao reconhecimento de expressões faciais.

No modelo sem contraste, as emoções *fear* e *angry* exibem confusão significativa, assim como *sad* e *angry*, bem como *happy* e *angry*. Após a aplicação do DGCL, nota-se uma reorga-

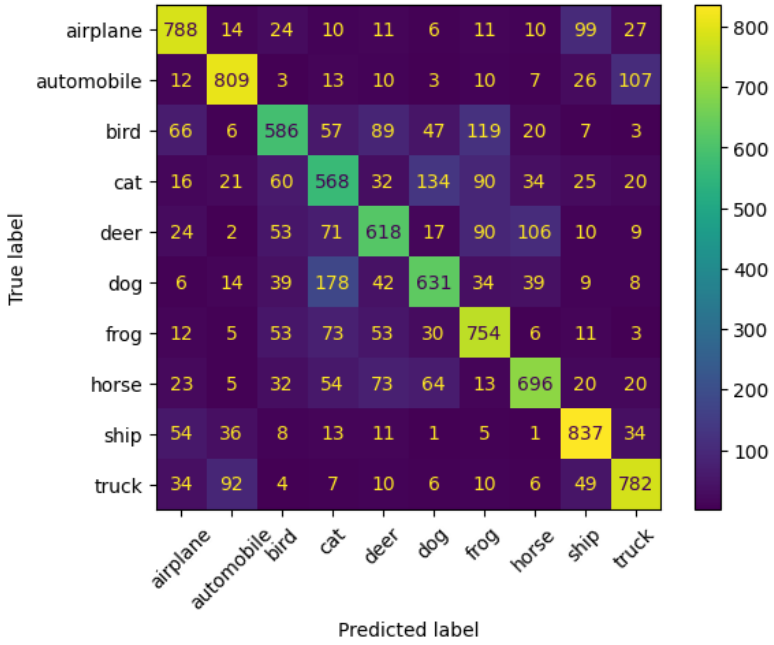


Figura 21 – Matriz de confusão do CIFAR-10 para o modelo com contraste (DGCL).

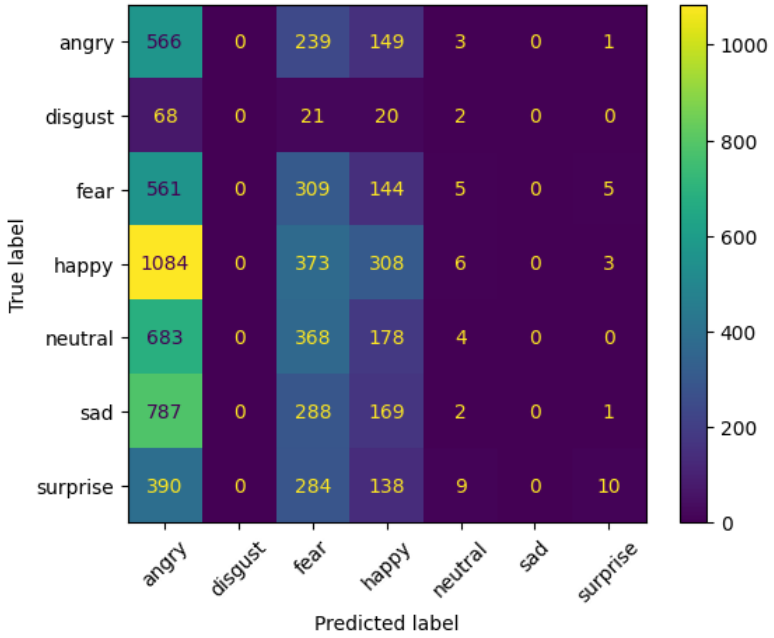


Figura 22 – Matriz de confusão do FER-13 para o modelo sem contraste.

nização mais coerente do espaço latente, resultando em maior separabilidade entre expressões de natureza próxima. A classe *happy* alcança um acerto expressivo (mais de 1.300 imagens corretamente classificadas), evidenciando a compactação intra-classe e o afastamento das fronteiras ambíguas.

As Figuras 24 e 39 mostram as matrizes de confusão do conjunto KDEF, composto por expressões faciais em ambiente controlado.

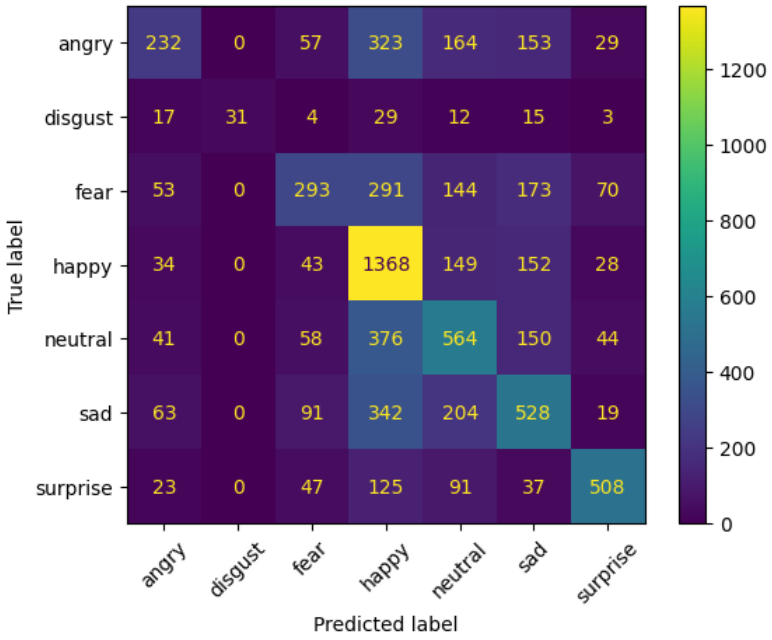


Figura 23 – Matriz de confusão do FER-13 para o modelo com contraste (DGCL).

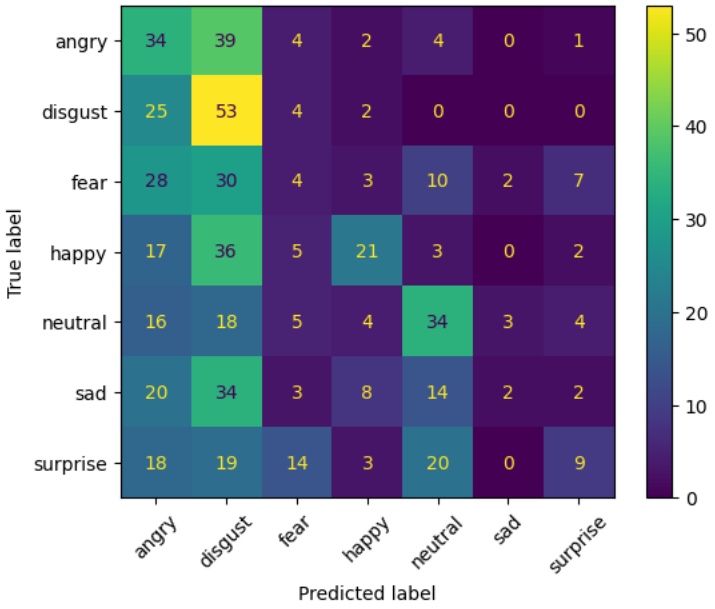


Figura 24 – Matriz de confusão do KDEF para o modelo sem contraste.

A análise revela que, no modelo base, há confusões frequentes entre *anger* e *disgust*, e entre *sad* e *disgust*, categorias com manifestações faciais semelhantes. O modelo DGCL, por outro lado, apresenta uma diagonal principal mais destacada, refletindo maior coerência nas fronteiras de decisão. As classes *happy* e *neutral* passam a ser mais bem delimitadas, e o número de confusões entre emoções negativas é significativamente reduzido. Esse resultado confirma a capacidade do método proposto em explorar diferenças sutis de expressão através da estrutura geométrica do espaço latente.

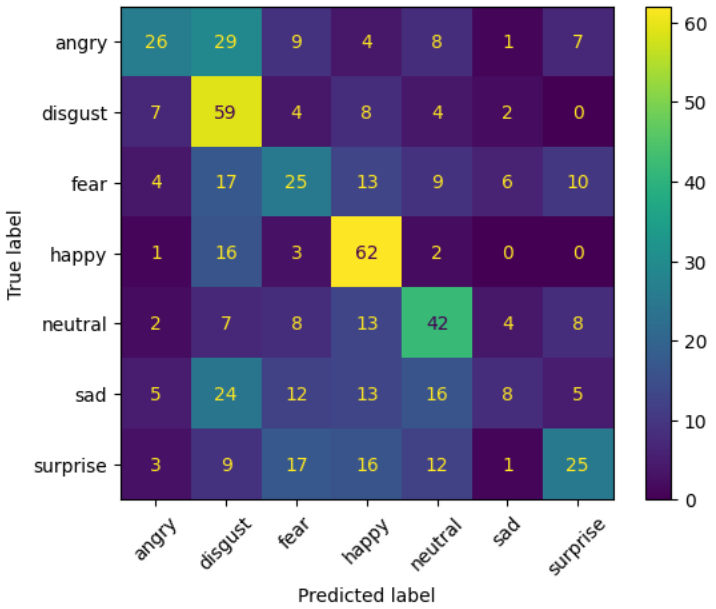


Figura 25 – Matriz de confusão do KDEF para o modelo com contraste (DGCL).

As Figuras 26 e 43 ilustram as matrizes de confusão para o conjunto RAF-DB, que contém expressões espontâneas e variadas.

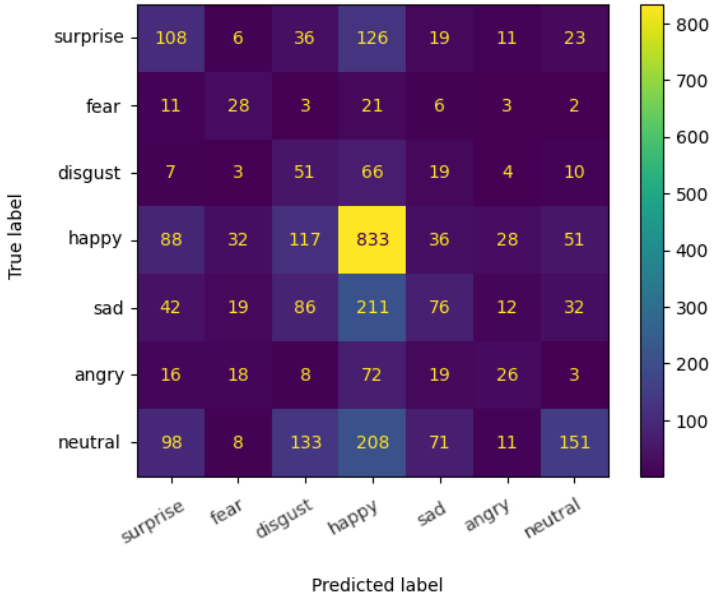


Figura 26 – Matriz de confusão do RAF-DB para o modelo sem contraste.

No modelo sem contraste, observa-se alta confusão entre as expressões *sad* e *surprise*, além de erros recorrentes entre *happy* e *disgust*. Após a introdução do DGCL, essas ambiguidades são substancialmente reduzidas, e a matriz evidencia maior concentração ao longo da diagonal principal. Isso indica que o modelo passou a capturar padrões faciais mais discriminativos, mesmo sob variações de iluminação, ângulo e contexto, características típicas do RAF-DB. O método proposto também melhora a identificação de classes minoritárias, demonstrando maior equilí-

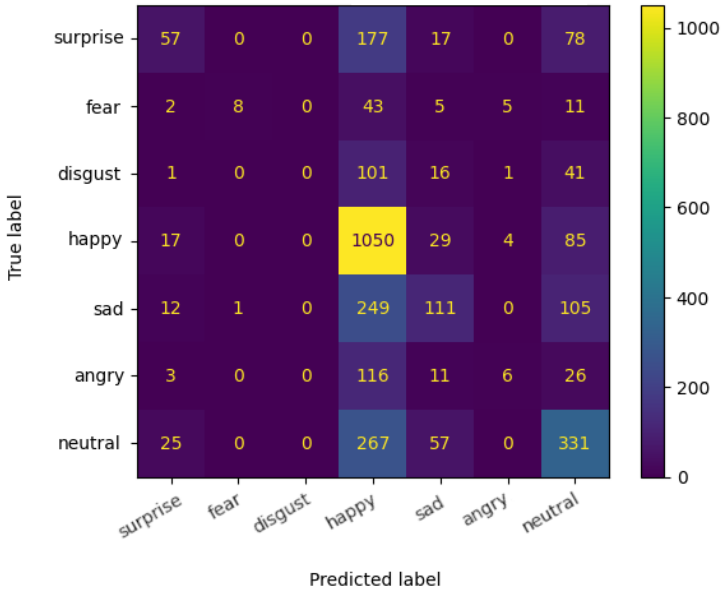


Figura 27 – Matriz de confusão do RAF-DB para o modelo com contraste (DGCL).

brio nas predições. Mesmo a partir de erros cometidos, como pode ser observado na matriz, o mesmo tende a concentrar os mesmos em uma única classe (geralmente com uma concentração maior em torno da diagonal principal) ao contrário da dispersão de erros apresentada sem a perda por contraste. Esse fato, possibilita que refinamentos posteriores, mais simplórios, sejam feitos em tal problema binário de classificação. Além disso, conjectura-se que com uma maior quantidade de ciclos de aprendizado esse ponto poderia ser sanado naturalmente pelo método proposto.

De modo geral, as matrizes de confusão indicam que o DGCL aprimora a coerência da decisão do classificador, reduzindo erros inter-classe e aumentando a precisão intra-classe. A comparação das matrizes de confusão entre os modelos *com* e *sem contraste* evidencia ganhos consistentes proporcionados pelo DGCL:

- ❏ As fronteiras de decisão tornam-se mais coerentes e estáveis, refletidas pela intensificação da diagonal principal nas matrizes contrastivas;
- ❏ Classes com alta similaridade visual, como *cat/dog* passam a ser melhor distinguidas;
- ❏ O modelo DGCL reduz as confusões entre emoções ambíguas e melhora a consistência entre classes minoritárias;
- ❏ Em todos os datasets, observa-se uma distribuição mais uniforme dos acertos e uma estrutura latente mais bem definida.

Em termos gerais, as análises qualitativas reforçam as conclusões quantitativas da Tabela 1, confirmando que o aprendizado contrastivo guiado por distância aprimora a separabilidade entre classes e a coerência geométrica das representações aprendidas.

4.3 Conclusão

Os experimentos realizados evidenciaram de forma consistente a efetividade do método proposto *Distance-Guided Contrastive Learning* (DGCL) em relação ao treinamento supervisionado convencional. A análise quantitativa, conduzida sobre os conjuntos de imagens, demonstrou um incremento expressivo de desempenho quando aplicado o DGCL. Esse resultado confirma que o uso de um mecanismo contrastivo guiado geometricamente mais do que duplica a capacidade discriminativa da rede, otimizando a separabilidade entre classes visualmente semelhantes.

As projeções obtidas por meio do t-SNE reforçam visualmente esses ganhos. No modelo sem aprendizado contrastivo, as representações latentes se mostraram difusas e sobrepostas, denotando baixa coerência semântica entre classes. Com a aplicação do DGCL, entretanto, observaram-se agrupamentos bem definidos e fronteiras suaves entre regiões de diferentes categorias. Essa reorganização do espaço latente evidencia que o método é capaz de estruturar o manifold de forma semanticamente consistente, reduzindo a variabilidade intra-classe e ampliando a distância inter-classe.

As matrizes de confusão derivadas dos classificadores também corroboram essas observações. No cenário sem o DGCL, observou-se elevada taxa de confusões entre categorias de aparência semelhante. Após o pré-treinamento contrastivo guiado por distância, as confusões foram significativamente reduzidas, refletindo uma fronteira de decisão mais estável e uma representação mais discriminativa.

A análise do comportamento da função de perda em relação ao número de âncoras utilizadas revelou, adicionalmente, a eficiência da proposta. As curvas de perda ao longo das épocas mostraram que, a partir da quinta época, a perda média estabiliza-se mesmo ao utilizar uma quantidade reduzida de âncoras (e.g. apenas 20%). Tal evidência indica que o DGCL converge rapidamente, concentrando o aprendizado nas amostras mais informativas e dispensando o uso exaustivo de todos os pares contrastivos. Essa característica resulta em substancial economia computacional sem prejuízo de desempenho.

Em síntese, os resultados obtidos demonstram que o DGCL promove uma melhora significativa tanto na organização geométrica das representações latentes quanto na acurácia de classificação final. O método mostrou-se eficiente, estável e escalável, apresentando um equilíbrio entre custo computacional e qualidade representacional. Assim, acredita-se que o DGCL possui características interessantes para consolidar-se como uma estratégia promissora de aprendizado autossupervisionado sensível à estrutura geométrica dos dados, potencialmente aplicável a diferentes domínios de visão computacional.

Capítulo 5

Conclusão

O presente trabalho apresentou uma abordagem de aprendizado autossupervisionado contrastivo orientado pela estrutura geométrica do espaço latente, denominada Distance-Guided Contrastive Learning (DGCL). O método proposto introduz um mecanismo de seleção de pares positivo-negativo guiado por distância, fundamentado na análise da configuração geométrica das amostras no espaço de representação.

A principal inovação do DGCL reside na integração entre aprendizado contrastivo e análise geométrica dinâmica, possibilitando a seleção adaptativa de exemplos informativos (*hard positives* e *hard negatives*) a partir das distâncias projetadas via t-SNE. Essa retroalimentação entre o espaço latente e a função de perda contrastiva permitiu ao modelo aprender representações semanticamente coerentes e estruturalmente consistentes, sem a necessidade de supervisão explícita.

Os resultados experimentais, obtidos em quatro conjuntos de imagens públicas com características distintas, CIFAR-10, FER-13, KDEF e RAF-DB, demonstraram ganhos expressivos de desempenho em relação a modelos sem o componente contrastivo. A melhoria relacionada a pontos percentuais na acurácia de classificação evidencia a eficácia da política de seleção guiada por distância, especialmente em tarefas envolvendo imagens naturais e de reconhecimento facial com alto grau de variabilidade.

A análise qualitativa das projeções t-SNE e das matrizes de confusão revelou que o DGCL favorece a formação de clusters bem definidos, com menor sobreposição entre classes e compactação intra-classe mais pronunciada. Esses resultados confirmam que a incorporação da estrutura geométrica no processo de aprendizado promove uma representação mais alinhada às relações semânticas subjacentes aos dados.

Além disso, observou-se que a abordagem proposta reduz a dependência do tamanho do lote e acelera a convergência do treinamento, já que a seleção informada de pares otimiza o uso de

amostras e evita redundâncias. Assim, o DGCL se mostra uma alternativa eficiente e escalável para cenários nos quais o custo computacional é um fator crítico. Em síntese, as contribuições deste trabalho podem ser assim sintetizadas:

- ❏ Proposição de um novo paradigma de aprendizado contrastivo autossupervisionado orientado por distância, combinando aspectos de aprendizado métrico, análise de manifolds e aprendizado contrastivo;
- ❏ Formulação de um algoritmo adaptativo de seleção de pares que explora a geometria do espaço latente como fonte de supervisão implícita;
- ❏ Demonstração empírica da eficiência e robustez do método em diferentes domínios visuais;
- ❏ Consolidação do aprendizado contrastivo como ferramenta versátil para representação de dados complexos, mesmo em contextos não rotulados.

5.1 Limitações e Desafios

Apesar dos resultados positivos, algumas limitações foram observadas e merecem destaque:

1. Dependência do t-SNE: Embora o t-SNE forneça uma boa estimativa local da geometria do espaço latente, ele não preserva relações globais de forma linear, o que pode introduzir distorções em grandes conjuntos de dados. Futuras abordagens podem explorar técnicas de projeção diferenciáveis ou incorporadas diretamente à rede;
2. Custo computacional da projeção: O recálculo periódico do t-SNE ao longo do treinamento, embora possa ser realizado esporadicamente, ainda representa um custo adicional. Estratégias de amostragem adaptativa ou de projeções parciais podem reduzir esse impacto;
3. Dependência de hiperparâmetros: A escolha da taxa de aprendizado e da periodicidade de atualização da projeção pode influenciar a estabilidade do treinamento. Estudos adicionais de sensibilidade podem aperfeiçoar a robustez do método;
4. Extensão para outros domínios: O DGCL foi avaliado em tarefas de classificação de imagens naturais e expressões faciais. A generalização para domínios específicos (e.g. imagens médicas, sensoriamento remoto, dentre outros), bem como multimodais (áudio, texto e vídeo) ainda requer investigações adicionais, embora os princípios geométricos do método permaneçam aplicáveis.

5.2 Trabalhos Futuros

Com base nas contribuições e limitações identificadas, diversas linhas de pesquisa podem ser exploradas a partir deste trabalho e servirão como ponto inicial para outros trabalhos desenvolvidos no presente grupo de pesquisa, que já estão em desenvolvimentos ou serão em breve atacados por outros alunos/pesquisadores do mesmo:

1. Projeções diferenciáveis: Investigar a substituição do t-SNE por métodos de redução de dimensionalidade diferenciáveis, como UMAP ou autoencoders variacionais, permitindo retropropagação direta dos gradientes geométricos;
2. Aprendizado multimodal: Estender o DGCL para integrar múltiplas modalidades sensoriais (imagens, áudio e texto), explorando o alinhamento de representações contrastivas entre diferentes espaços latentes;
3. Incorporação de atenção espacial: Integrar mecanismos de *self-attention* que permitam ao modelo identificar regiões mais relevantes das imagens para o cálculo da similaridade, tornando o aprendizado mais interpretável;
4. Avaliação em datasets maiores: Aplicar o DGCL a bases de dados de larga escala, como ImageNet ou AffectNet, a fim de avaliar sua escalabilidade e estabilidade em cenários de alta cardinalidade (escala de milhões de imagens);
5. Integração com aprendizado supervisionado: Investigar estratégias híbridas que combinem o DGCL com perda supervisionada parcial, permitindo refinar as fronteiras de decisão em cenários semi-rotulados;
6. Análise teórica da convergência: Desenvolver estudos formais sobre as propriedades de convergência da política de seleção de pares guiada por distância, visando fundamentar matematicamente a estabilidade observada empiricamente;
7. Integração com abordagens de aprendizado ativo: Fusionar o método proposto a abordagens de aprendizado ativo;
8. Aplicação de diferentes encoders: Testar a aplicação de outros encoders (e.g. Vision Transformers, ConvNext, dentre outros);
9. Utilização de Agrupamento: Substituir o uso atual de rótulos de classe para análise pela consistência de agrupamentos não supervisionados, permitindo que o DGCL opere de forma totalmente autossupervisionada em cenários práticos;
10. Seleção Aleatória: Comparar o método proposto em relação à seleção aleatória de pares positivos e negativos;
11. Problema Multi-rótulo: Estender o arcabouço proposto para aplicações que envolvam problemas de classificação multi-rótulo.

5.3 Considerações Finais

O desenvolvimento do Distance-Guided Contrastive Learning consolidou a ideia de que o aprendizado contrastivo pode ser potencializado por mecanismos que respeitam a estrutura geométrica intrínseca dos dados. Ao utilizar a própria topologia do espaço latente como guia, o método proposto transcende a simples comparação aleatória de amostras e inaugura uma perspectiva mais informada e adaptativa do aprendizado autossupervisionado.

Os resultados obtidos demonstram que é possível alcançar representações mais discriminativas, estáveis e generalizáveis sem recorrer a supervisão explícita, o que reforça o potencial do DGCL para aplicações futuras em aprendizado multimodal e em cenários com restrições de anotação.

Com isso, este trabalho contribui para o avanço das pesquisas em aprendizado autossupervisionado, propondo uma abordagem que alia fundamentação teórica sólida, viabilidade experimental e resultados empiricamente comprovados.

Vale ressaltar que, até o momento da redação deste trabalho, foi realizada a submissão de um artigo à conferência *International Conference on Computer Vision Theory and Applications* (VISAPP), atualmente classificada como **A2** de acordo com os critérios estabelecidos pelas Comissões Especiais (CEs) da Sociedade Brasileira de Computação (SBC). Ademais, a conferência encontra-se entre as 20 principais da área (top-20), segundo a Comissão Especial de Computação Gráfica e Processamento de Imagens (CEGRAPI).

Referências

CARON, M. et al. **Unsupervised Learning of Visual Features by Contrasting Cluster Assignments**. 34th Conference on Neural Information Processing Systems, Vancouver, Canadá, 2021. Disponível em: <https://doi.org/10.48550/arXiv.2006.09882>.

CARRIER, P.-L.; COURVILLE, A. **FER-2013**. [S.l.]: Wolfram Data Repository, 2017. <https://datarepository.wolframcloud.com/resources/FER-2013>.

CHEN, T.; KORNBLITH, S.; NOROUZI, M.; HINTON, G. A simple framework for contrastive learning of visual representations. In: **Proceedings of the International Conference on Machine Learning**. JMLR.org, 2020. p. 1597–1607. Disponível em: <https://doi.org/10.48550/arXiv.2002.05709>.

DENG, W. **Real-world Affective Faces Database (RAF-DB)**. [S.l.]: Beijing University, 2017. Disponível em: <https://www.whdeng.cn/RAF/model1.html>.

GRILL, J.-B. et al. **Bootstrap your own latent: A new approach to self-supervised Learning**. [S.l.], 2020. Disponível em: <https://doi.org/10.48550/arXiv.2006.07733>.

HE, K.; FAN, H.; WU, Y.; XIE, S.; GIRSHICK, R. **Momentum Contrast for Unsupervised Visual Representation Learning**. Facebook AI Research, 2020. Disponível em: <https://doi.org/10.48550/arXiv.1911.05722>.

HE, K.; ZHANG, X.; REN, S.; SUN, J. **Deep Residual Learning for Image Recognition**. [S.l.], 2015. Disponível em: <https://doi.org/10.48550/arXiv.1512.03385>.

JAISWAL, A.; BABU, A. R.; ZADEH, M. Z.; BANERJEE, D.; MAKEDON, F. **A Survey on Contrastive Self-Supervised Learning**. University of Texas at Arlington, Estados Unidos, 2020. Disponível em: <https://doi.org/10.3390/technologies9010002>.

JUNG, S.; DAGOBERT, T.; MOREL, J.-M.; FACCIOLO, G. A review of t-sne. **Image Processing On Line**, v. 14, p. 250–270, 2024. Disponível em: <https://doi.org/10.5201/ipol.2024.528>.

KRIZHEVSKY, A.; HINTON, G. **Learning multiple layers of features from tiny images**. Toronto, Ontario, 2009. Disponível em: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, p. 436–444, 2015. Disponível em: <https://doi.org/10.1038/nature14539>.

LIU, X. et al. Self-supervised learning: Generative or contrastive. **IEEE Transactions on Knowledge and Data Engineering**, v. 35, n. 1, p. 857–876, 2023.

LUNDQVIST, D.; FLYKT, A.; ÖHMAN, A. **The Karolinska Directed Emotional Faces - KDEF**. Department of Clinical Neuroscience, Psychology Section, Karolinska Institutet, 1998. CD ROM, ISBN 91-630-7164-9. Disponível em: <https://kdef.se/>.

MAATEN, L. J. van der. Learning a parametric embedding by preserving local structure. In: **Proceedings of the Twelfth International Conference on Artificial Intelligence e Statistics (AI-Stats)**. [S.l.: s.n.], 2009. p. 384–391.

MAATEN, L. J. van der; HINTON, G. E. Visualizing high-dimensional data using t-sne. **Journal of Machine Learning Research**, v. 9, p. 2579–2605, 2008. Disponível em: <https://www.jmlr.org/papers/v9/vandermaaten08a.html>.

MAHADEVKAR, S. V. et al. **A Review on Machine Learning Styles in Computer Vision - Techniques and Future Directions**. IEEE Access, 2022. Disponível em: <https://www.doi.org/10.1109/ACCESS.2022.3209825>.

RANI, V.; NABI, S. T.; KUMAR, M.; MITTAL, A.; KUMAR, K. **Self-supervised Learning: A Succinct Review**. Archives of Computational Methods in Engineering, Vol.30, p.2761-2775, 2023. Disponível em: <https://doi.org/10.1007/s11831-023-09884-2>.

ROCA, C. P. et al. **A Cross Entropy Test Allows Quantitative Statistical Comparison of t-SNE and UMAP Representations**. [S.l.], 2021. Disponível em: <https://arxiv.org/abs/2112.04172>.

SAKIB, S.; SIDDIQUE, M. A. B.; RAHMAN, M. A. **Performance Evaluation of t-SNE and MDS Dimensionality Reduction Techniques with KNN, ENN and SVM Classifiers**. [S.l.], 2020. Disponível em: <https://arxiv.org/abs/2007.13487>.

SUN, Y.; HAN, Y.; FAN, J. **Laplacian-based Cluster-Contractive t-SNE for High Dimensional Data Visualization**. [S.l.], 2022. Disponível em: <https://arxiv.org/abs/2207.12214>.

TIAN, Y. **Artificial Intelligence Image Recognition Method Based on Convolutional Neural Network Algorithm**. [S.l.], 2020. Disponível em: <https://doi.org/10.1109/access.2020.3006097>.

VIET, H. H.; ANH, D. T. M-tree as an index structure for time series data. In: **International Conference on Computing, Management and Telecommunications**. [S.l.: s.n.], 2013. p. 146–151.

Apêndices

APÊNDICE A

Análise Assintótica

A.1 Complexidade Assintótica do Método DGCL

Nesta seção apresenta-se uma análise detalhada da complexidade assintótica do método *Distance-Guided Contrastive Learning* (DGCL) em comparação com a complexidade associada ao treinamento de uma arquitetura de referência do tipo ResNet-50 (ou outras CNNs generalistas). A análise é formulada em termos de custo computacional (tempo) e custo de memória (espaço), utilizando notação assintótica (Big-O).

A.1.1 Notação e variáveis

Sejam as seguintes variáveis usadas ao longo da análise:

- N : número total de amostras no conjunto de treinamento;
- E : número total de épocas de treino;
- B : tamanho do mini-lote (batch);
- d : dimensão do vetor de embedding produzido pelo codificador f_θ;
- D : dimensão da projeção usada para visualização/seleção (no DGCL normalmente $D = 2$);
- k_p, k_n : número de positivos e negativos selecionados por âncora (ou, de modo simplificado, k pares por âncora);
- T_{tsne} : frequência (em épocas) com que a projeção t-SNE é recomputada (ex.: a cada T_{tsne} épocas);

- C_{enc} : custo (FLOPs ou tempo) de uma passagem *forward+backward* do codificador por amostra (ResNet-50), i.e. custo por amostra para atualizar pesos (inclui forward e backward);
- C_{fwd} : custo de uma passagem apenas *forward* do codificador por amostra (quando necessário gerar embeddings sem atualização);
- $S_{\text{tsne}}(N)$: custo temporal de computar a projeção t-SNE sobre N pontos (dependente do algoritmo/implementação; ver Considerações abaixo);
- M_{enc} : memória necessária para armazenar parâmetros da rede (constante em relação a N);
- $M_{\text{act}}(B)$: memória para ativação por batch (proporcional a B).

Observação: C_{enc} e C_{fwd} podem ser tratados genericamente como $O(P)$ onde P representa a ordem de magnitude do número de FLOPs da arquitetura (por exemplo ResNet-50 tem $P \approx$ algumas dezenas de GFLOPs por imagem dependendo da implementação). A análise abaixo mantém C_{enc} simbólica para generalidade.

A.1.2 Complexidade do treinamento *baseline* (ResNet-50 sem DGCL)

Para um regime supervisionado / baseline sem componentes adicionais ao pipeline (ou seja, treinamento convencional por Época usando batches de tamanho B), o custo total de tempo é aproximadamente:

$$T_{\text{base}} = E \cdot \left(\frac{N}{B}\right) \cdot C_{\text{enc}} \quad \Longrightarrow \quad T_{\text{base}} = O\left(E \cdot \frac{N}{B} \cdot C_{\text{enc}}\right).$$

A memória ocupada é, de modo dominante,

$$M_{\text{base}} = O\big(M_{\text{enc}} + M_{\text{act}}(B)\big).$$

A.1.3 Componentes adicionais introduzidos pelo DGCL

O DGCL acrescenta os seguintes componentes que impactam complexidade temporal e espacial:

1. **Cálculo de embeddings (forward pass) para todo o conjunto**
   Periodicamente (tipicamente a cada época ou cada T_{tsne} épocas), é necessário computar embeddings $z_i = f_\theta(x_i)$ para todas as N amostras. Se as projeções ocorrem a cada T_{tsne} épocas, esse custo acumulado ao longo do treinamento é:

   $$T_{\text{emb}} = \left(\frac{E}{T_{\text{tsne}}}\right) \cdot N \cdot C_{\text{fwd}} \quad \Longrightarrow \quad T_{\text{emb}} = O\left(\frac{E}{T_{\text{tsne}}} N C_{\text{fwd}}\right).$$

   Note que $C_{\text{fwd}} \leq C_{\text{enc}}$ (apenas forward).

2. **Projeção t-SNE e cálculo de distâncias no espaço projetado**
   A projeção em si tem custo $S_{\text{tsne}}(N)$. Em implementações exatas de t-SNE o custo é $O(N^2)$ por iteração (porém práticas modernas usam Barnes-Hut t-SNE, custo $O(N \log N)$ por iteração, ou alternativas como UMAP com custo aproximado $O(N \log N)$). Assumindo que a projeção é recomputada $\frac{E}{T_{\text{tsne}}}$ vezes, o custo total de projeção é:

$$T_{\text{tsne}}^{\text{total}} = \left(\frac{E}{T_{\text{tsne}}} \right) \cdot S_{\text{tsne}}(N).$$

   Após a projeção, o cálculo de todas as distâncias $\|\tilde{z}_i - \tilde{z}_j\|_2$ na dimensão D tem custo $O(N^2 \cdot D)$, i.e.

$$T_{\text{dist}} = \left(\frac{E}{T_{\text{tsne}}} \right) \cdot O(N^2 D) = O\left(\frac{E}{T_{\text{tsne}}} N^2 \right)$$

   (como D é constante pequeno, ex.: $D = 2$).

3. **Seleção de pares por âncora**
   Para cada âncora onde se disponha da matriz de distâncias, selecionar os k_p positivos mais distantes e k_n negativos mais próximos equivale, ingenuamente, a um custo de ordenação por linha $O(N \log N)$ ou uma seleção em $O(N)$ (seleção linear), totalizando $O(N^2)$ para todas as amostras. Assim:

$$T_{\text{sel}} = \left(\frac{E}{T_{\text{tsne}}} \right) \cdot O(N^2).$$

   Alternativamente, se a seleção for restrita a um subconjunto candidato via ANN (aprox. nearest neighbors) na dimensão d, o custo pode cair para $O(N \log N)$ (ou $O(N)$ dependendo do índice).

4. **Cálculo da perda contrastiva por batch com pares selecionados**
   Em cada batch de tamanho B, para cada âncora calcula-se similaridade com k pares (onde $k = k_p + k_n$ ou simplificadamente k). O custo por batch é $O(B \cdot k \cdot d)$ (produto escalar/cosseno entre vetores de dimensão d). O número total de batches por época é N/B, portando o custo total ao longo de E épocas é:

$$T_{\text{loss}} = E \cdot \frac{N}{B} \cdot O(B \cdot k \cdot d) = O(E \cdot N \cdot k \cdot d).$$

   Note que $k \ll N$ e d é tipicamente moderado (ex.: 128–2048).

A.1.4 Expressão total da complexidade temporal do DGCL

Somando os termos relevantes obtemos uma expressão assintótica para o tempo total do DGCL:

$$T_{\text{DGCL}} = \underbrace{E \cdot \frac{N}{B} \cdot C_{\text{enc}}}_{\text{(i) atualizações com backprop (mesma ordem que baseline)}} + \underbrace{\frac{E}{T_{\text{tsne}}} N C_{\text{fwd}}}_{\text{(ii) embeddings completos periódicos}}$$
$$+ \underbrace{\frac{E}{T_{\text{tsne}}} S_{\text{tsne}}(N)}_{\text{(iii) projeções}} + \underbrace{\frac{E}{T_{\text{tsne}}} O(N^2)}_{\text{(iv) distâncias \& seleção}} + \underbrace{O(E \cdot N \cdot k \cdot d)}_{\text{(v) perda contrastiva por pares}} .$$

Ao comparar T_{DGCL} com T_{base}, os termos adicionais dominantes introduzidos pelo DGCL são os relacionados à projeção e ao custo quadrático em N (itens (iii) e (iv)), salvo quando se utilizam aproximações. Em particular, se $S_{\text{tsne}}(N) = \Theta(N^2)$, os termos quadráticos em N tornam o algoritmo sobremaneira custoso para grandes bases de dados.

Caso prático (implementação exata de t-SNE):

se $S_{\text{tsne}}(N) = \Theta(N^2)$ e $T_{\text{tsne}} = 1$ (recalcular a cada época), o termo $O(EN^2)$ domina, levando a:

$$T_{\text{DGCL}} = O(E \cdot N^2) \quad \text{(dominante)}.$$

Caso otimizado (Barnes-Hut t-SNE ou UMAP + ANN para seleção):

com $S_{\text{tsne}}(N) = O(N \log N)$ e seleção por ANN com custo total $O(N \log N)$ por recomputação, o custo adicional por recomputação fica em $O(N \log N)$. Assim, os termos quadráticos podem ser substituídos por quasilineares:

$$T_{\text{DGCL}} = O\left(E \cdot \frac{N}{B} C_{\text{enc}} + \frac{E}{T_{\text{tsne}}} N \log N + E \cdot N \cdot k \cdot d\right).$$

Nesse cenário o custo tende a ser dominado novamente pelo termo $E \cdot (N/B) C_{\text{enc}}$ (ou pelos custos $N \log N$ se C_{enc} for pequeno).

A.1.5 Complexidade espacial (memória)

- ❑ **Baseline:** $M_{\text{base}} = O(M_{\text{enc}} + M_{\text{act}}(B))$.
- ❑ **DGCL (ingênuo):** além de M_{enc} e $M_{\text{act}}(B)$, deve-se armazenar a matriz de distâncias $O(N^2)$ (se calculada e mantida inteira), e os embeddings $O(N \cdot d)$. Logo:

$$M_{\text{DGCL}} = O\big(M_{\text{enc}} + M_{\text{act}}(B) + N \cdot d + N^2\big).$$

- ❑ **DGCL (com aproximações):** usando índices ANN e estruturas de vizinhança esparsas, a necessidade de armazenar $O(N^2)$ pode ser evitada, reduzindo para $O(N \log N)$ ou $O(N \cdot k)$ dependendo da técnica. Assim:

$$M_{\text{DGCL, approx}} = O\big(M_{\text{enc}} + M_{\text{act}}(B) + N \cdot d + N \log N\big).$$

A.1.6 Comparação qualitativa com ResNet-50

- ❏ O termo de custo associado à *execução e atualização* da ResNet-50 (i.e., treinamento padrão) é comum ao DGCL: $O(E \cdot (N/B) C_{\text{enc}})$. Portanto, em cenários onde os custos de projeção e seleção são negligenciáveis (por exemplo, T_{tsne} grande e/ou N pequeno), o overhead do DGCL é limitado ao custo de computar similaridades extras $O(ENkd)$, tipicamente moderado.

- ❏ Em problemas de escala média-grande (N grande), a projeção global e a seleção exata $O(N^2)$ tornam o DGCL muito mais custoso do que o treinamento com ResNet-50 puro, a menos que se adotem aproximações (Barnes-Hut t-SNE, UMAP, índices ANN, amostragem estratificada, projeções paramétricas).

- ❏ A memória adicional exigida por DGCL em sua forma ingênua ($O(N^2)$) é, na prática, o maior impedimento para aplicabilidade em datasets grandes; isto motiva fortemente o uso de aproximações ou estratégias que limitem a computação de distâncias a uma vizinhança local.

A.1.7 Recomendações práticas e estratégias de mitigação

Para tornar DGCL viável em grandes conjuntos de dados recomenda-se:

1. **Recomputar projeções com baixa frequência:** aumentar T_{tsne} reduz o número de recomputações e amortiza o custo de $S_{\text{tsne}}(N)$.

2. **Usar projeções e índices paramétricos/mais rápidos:** substituir t-SNE exato por Barnes-Hut t-SNE, UMAP ou embeddings paramétricos (neural parametric t-SNE), reduzindo $S_{\text{tsne}}(N)$ para $O(N \log N)$.

3. **Limitar a busca de pares a vizinhanças aproximadas:** empregar ANN (e.g., FAISS, HNSW) para obter candidatos de kNN em $O(N \log N)$ ou $O(N)$.

4. **Amostragem estratificada:** projetar e selecionar pares apenas em subconjuntos representativos por classe ou cluster para evitar custo quadrático global.

5. **Batch-wise or online selection:** realizar seleções locais usando informação de batch ou janelas deslizantes para reduzir custo espacial.

A.1.8 Resumo final (fórmulas compactas)

- ❏ **Baseline (ResNet-50):**

$$T_{\text{base}} = O\left(E \cdot \frac{N}{B} C_{\text{enc}}\right), \quad M_{\text{base}} = O(M_{\text{enc}} + M_{\text{act}}(B)).$$

❑ **DGCL (ingênuo, t-SNE exato):**

$$T_{\mathrm{DGCL}} = O\left(E \cdot \frac{N}{B} C_{\mathrm{enc}} \; + \; E \cdot N^2\right), \quad M_{\mathrm{DGCL}} = O\big(M_{\mathrm{enc}} + M_{\mathrm{act}}(B) + N^2\big).$$

❑ **DGCL (otimizado com t-SNE/UMAP/ANN):**

$$T_{\mathrm{DGCL}} = O\left(E \cdot \frac{N}{B} C_{\mathrm{enc}} \; + \; \frac{E}{T_{\mathrm{tsne}}} N \log N \; + \; E \cdot N \cdot k \cdot d\right),$$

$$M_{\mathrm{DGCL}} = O\big(M_{\mathrm{enc}} + M_{\mathrm{act}}(B) + N \cdot d + N \log N\big).$$

Conclusão:

A adição do componente geométrico de seleção de pares no DGCL implica, em termos assintóticos, um overhead significativo quando implementado de forma exata (i.e., $O(N^2)$ em tempo e espaço). Contudo, com escolhas algorítmicas e de engenharia (projeções aproximadas, índices ANN, amostragem), é possível reduzir a complexidade para quasilinear, tornando o método compatível com a prática em escalas maiores. Em cenários de prova de conceito e datasets de porte moderado (por exemplo, CIFAR-10), o custo adicional é manejável e o ganho em qualidade de representação pode justificar o overhead computacional.

A.2 Exemplos numéricos — caso CIFAR-10

Para tornar mais concreto o impacto computacional do método *Distance-Guided Contrastive Learning* (DGCL), apresenta-se abaixo uma avaliação numérica aproximada considerando o conjunto de treinamento do *CIFAR-10*. Os cálculos a seguir são aproximados e têm por objetivo evidenciar a ordem de grandeza dos custos adicionais introduzidos pelo DGCL em comparação com o treinamento padrão de uma rede do tipo ResNet-50.

Hipóteses e parâmetros numéricos

Adotam-se as seguintes suposições, compatíveis com a configuração experimental e com valores comumente usados na literatura:

❑ número de amostras de treinamento: $N = 50\,000$ (CIFAR-10: 50k treino);

❑ número de épocas experimentais: $E = 6$ (configuração adotada neste trabalho);

❑ tamanho de mini-lote: $B = 128$;

❑ dimensão do embedding: $d = 128$ (valor típico para cabeças de projeção);

❑ número médio de pares por âncora: $k = k_p + k_n = 10$ (ex.: 5 positivos + 5 negativos);

- ❏ custo computacional do codificador (forward + backward) por amostra: $C_{\text{enc}} = 4 \times 10^8$ FLOPs (assunção conservadora para imagens pequenas; representa a ordem de magnitude do custo por amostra em arquiteturas profundas reduzidas para CIFAR);
- ❏ custo computacional do codificador (forward apenas) por amostra: $C_{\text{fwd}} = 1 \times 10^8$ FLOPs;
- ❏ dimensão da projeção para cálculo de distâncias: $D = 2$ (t-SNE bidimensional);
- ❏ operações estimadas por par de pontos 2D ao calcular distância (diferença, quadrado, soma, etc.): aproximadamente 6 operações por par;
- ❏ memória por número em ponto flutuante: 4 bytes (float32).

Contagens básicas

$$\text{nº de pares (únicos)} \approx \frac{N(N-1)}{2} \approx \frac{50\,000 \cdot 49\,999}{2} \approx 1{,}249{,}975{,}000$$
$$N^2 = 50{,}000^2 = 2{,}500{,}000{,}000.$$

Custo do baseline (treinamento ResNet-50 sem DGCL)

O custo total (em FLOPs) do treinamento convencional pode ser aproximado por:

$$T_{\text{base}} = E \cdot \frac{N}{B} \cdot C_{\text{enc}}.$$

Substituindo os valores assumidos:

$$T_{\text{base}} = 6 \cdot \frac{50{,}000}{128} \cdot 4 \times 10^8 = 6 \cdot 390.625 \cdot 4 \times 10^8 \approx 9.375 \times 10^{11} \text{ FLOPs}.$$

(isto é, $\approx 9.38 \times 10^{11}$ FLOPs — ordem de 10^{12} FLOPs).

Termos adicionais introduzidos pelo DGCL

Considerando os termos descritos anteriormente, estimam-se os seguintes custos (FLOPs e memória):

1. **Custo de computar embeddings completos (forward) em todo o conjunto**
   Se as embeddings forem computadas em cada recomputação do *t*-SNE, e supondo que a projeção seja feita $\frac{E}{T_{\text{tsne}}}$ vezes, o custo é:

$$T_{\text{emb}} = \frac{E}{T_{\text{tsne}}} \cdot N \cdot C_{\text{fwd}}.$$

❑ **Pior caso: recomputação a cada época** ($T_{\text{tsne}} = 1$):

$$T_{\text{emb}} = 6 \cdot 50{,}000 \cdot 1 \times 10^8 = 3.0 \times 10^{13} \text{ FLOPs.}$$

❑ **Cenário prático: recomputação a cada 5 épocas** ($T_{\text{tsne}} = 5$):

$$T_{\text{emb}} = \frac{6}{5} \cdot 50{,}000 \cdot 1 \times 10^8 \approx 6.0 \times 10^{12} \text{ FLOPs.}$$

2. **Cálculo de distâncias entre pontos no espaço projetado (2D)**
   Número de pares únicos $\approx 1.25 \times 10^9$. Com ≈ 6 operações por par, o custo de calcular todas as distâncias é da ordem de:

   $$T_{\text{dist\_pairs}} \approx 1.249975 \times 10^9 \times 6 \approx 7.50 \times 10^9 \text{ operações (FLOPs).}$$

   Ao recomputar a projeção $\frac{E}{T_{\text{tsne}}}$ vezes, o custo escala linearmente com esse fator.

3. **Custo de seleção de pares (ingênuo)**
   A seleção por âncora, feita sobre a matriz de distâncias, pode implicar ordenações ou seleções lineares por linha. O custo total ingênuo é da ordem de $O(N^2)$ por recomputação. Em termos absolutos, mantém-se na mesma ordem de magnitude do cálculo de distâncias (alguns bilhões de operações).

4. **Custo da perda contrastiva por pares**
   Estimando operações por par proporcional a d, o custo total por todas as épocas é aproximadamente:

   $$T_{\text{loss}} \approx E \cdot N \cdot k \cdot d = 6 \cdot 50{,}000 \cdot 10 \cdot 128 = 3.84 \times 10^8 \text{ operações (FLOPs).}$$

   (ordem de 10^8 FLOPs — relativamente pequeno comparado aos termos anteriores).

Memória adicional (ordens de grandeza)

❑ matriz de distâncias (guardar N^2 floats, float32): 2.5×10^9 entradas $\times 4$ bytes $\approx 10 \times 10^9$ bytes ≈ 10 GB.

❑ embeddings (guardar $N \cdot d$ floats): $50{,}000 \cdot 128 = 6.4 \times 10^6$ floats $\times 4$ bytes ≈ 25.6 MB.

Totais numéricos (substituição e comparação)

Usando as expressões anteriores, apresentam-se duas estimativas do custo total (soma dos termos relevantes) — em FLOPs:

1. Pior caso — recomputação do t-SNE a cada época ($T_{\text{tsne}} = 1$):

$$\begin{aligned} T_{\text{DGCL, worst}} &\approx T_{\text{base}} + T_{\text{emb}} + T_{\text{dist\_pairs}} + T_{\text{loss}} \\ &\approx 9.375 \times 10^{11} \; + \; 3.0 \times 10^{13} \; + \; 7.5 \times 10^9 \; + \; 3.84 \times 10^8 \\ &\approx 3.0945 \times 10^{13} \text{ FLOPs.} \end{aligned}$$

Neste cenário o termo dominante é o custo das embeddings periódicas e do forward completo (ordem 10^{13} FLOPs).

2. Cenário prático — recomputação do t-SNE a cada 5 épocas ($T_{\text{tsne}} = 5$):

$$\begin{aligned} T_{\text{DGCL, practical}} &\approx T_{\text{base}} + \frac{E}{5} N C_{\text{fwd}} + \frac{E}{5} T_{\text{dist\_pairs}} + T_{\text{loss}} \\ &\approx 9.375 \times 10^{11} \; + \; 6.0 \times 10^{12} \; + \; 8.99982 \times 10^{9} \; + \; 3.84 \times 10^{8} \\ &\approx 6.9469 \times 10^{12} \text{ FLOPs}. \end{aligned}$$

Neste cenário, o overhead permanece significativo (ordem 10^{12}–10^{13} FLOPs), mas reduzido em relação ao pior caso.

Interpretação e comentários

- ❏ A ordem de magnitude das operações adicionais devido ao DGCL em sua implementação ingênua é elevada principalmente por dois fatores: (i) a necessidade de computar embeddings para todo o conjunto periodicamente e (ii) o custo quadrático implícito nas operações sobre todas as distâncias entre pares ($O(N^2)$).

- ❏ Em termos de memória, armazenar a matriz inteira de distâncias em float32 requer aproximadamente **10 GB** para $N = 50{,}000$, o que já é um requisito de hardware não trivial (ainda mais se se optar por float64).

- ❏ Reduzir a frequência de recomputação do t-SNE (aumentar T_{tsne}), usar projeções/algoritmos mais rápidos (Barnes-Hut t-SNE, UMAP) ou limitar a busca de pares a vizinhanças aproximadas (índices ANN: FAISS/HNSW) reduz fortemente os termos dominantes e torna o DGCL praticável em escala CIFAR-10 e além.

- ❏ Em particular, se substituirmos a projeção e a seleção exatas por algoritmos com custo aproximadamente $O(N \log N)$, os termos quadráticos são substituídos por termos quasi-lineares e o custo adicional passa a ser compatível com o custo do próprio treinamento (i.e., o termo $E \cdot (N/B) C_{\text{enc}}$ volta a ser dominante).

Resumo numérico final (valores-chave)

- ❏ $\boxed{T_{\text{base}} \approx 9.38 \times 10^{11} \text{ FLOPs}}$.
- ❏ $\boxed{T_{\text{DGCL, practical}} \approx 6.95 \times 10^{12} \text{ FLOPs}}$ (recomputação t-SNE a cada 5 épocas).
- ❏ $\boxed{\text{Memória para matriz de distâncias (float32)} \approx 10 \text{ GB}}$.
- ❏ $\boxed{\text{Memória para embeddings } (N \times d = 50{,}000 \times 128) \approx 25.6 \text{ MB}}$.

Conclusão prática

Os exemplos numéricos acima mostram que, mesmo em um dataset de porte moderado como o CIFAR-10, a implementação ingênua do DGCL (recomputando projeção e distâncias completas a cada época e armazenando a matriz N^2) pode introduzir overheads consideráveis em FLOPs e memória. Contudo, com estratégias usuais de engenharia (reduzir frequência de recomputação, usar UMAP/Barnes-Hut, índices ANN, amostragem estratificada), o custo extra pode ser reduzido a níveis viáveis, preservando-se os benefícios de qualidade de representação demonstrados nos experimentos.

A.3 Análise de Custo–Benefício: DGCL vs. Treinamento Convencional (ResNet-50)

Esta seção apresenta uma análise comparativa de custo–benefício entre o método *Distance-Guided Contrastive Learning* (DGCL) e o treinamento convencional de uma arquitetura de referência (ResNet-50), considerando o cenário experimental adotado neste trabalho (treinamento por $E = 6$ épocas sobre o conjunto CIFAR-10). Em particular, explora-se o caso observado nos experimentos: o ganho de acurácia do DGCL em CIFAR-10 foi obtido passando de 34,06% (baseline sem DGCL) para 70,69% (com DGCL). A análise combina fórmulas assintóticas com estimativas numéricas para quantificar o trade-off entre custo computacional e melhoria de acurácia. Nesta versão assume-se explicitamente que as projeções (t-SNE) foram recomputadas *época a época* ($T_{\text{tsne}} = 1$) durante as 6 épocas.

A.3.1 Premissas e parâmetros numéricos

Adotam-se as mesmas convenções e aproximações empregadas anteriormente, com as especificações relevantes para o caso com subamostragem por âncoras:

- conjunto CIFAR-10: $N = 50{,}000$ amostras de treino;
- número de âncoras usadas pelo DGCL: $N_a = 5{,}000$;
- número de épocas: $E = 6$;
- recomputação de projeção: $T_{\text{tsne}} = 1$ (recomputação a cada época);
- batch size: $B = 128$;
- dimensão do embedding: $d = 128$;
- pares por âncora: $k = 10$ (5 positivos + 5 negativos, por exemplo);

- ❏ custo (forward+backward) por amostra do codificador (ordem de magnitude): $C_{\text{enc}} = 4.0 \times 10^8$ FLOPs;
- ❏ custo (forward apenas) por amostra: $C_{\text{fwd}} = 1.0 \times 10^8$ FLOPs;
- ❏ acurácia do baseline (sem DGCL): $A_{\text{base}} = 34{,}06\%$;
- ❏ acurácia com DGCL: $A_{\text{DGCL}} = 70{,}69\%$.

Definem-se as quantidades de interesse:

$$\Delta A \;=\; A_{\text{DGCL}} - A_{\text{base}} = 70{,}69\% - 34{,}06\% = 36{,}63 \text{ p.p..}$$

A.3.2 Custo do baseline (ResNet-50)

O custo aproximado do treinamento convencional (baseline) em FLOPs é:

$$T_{\text{base}} \;=\; E \cdot \frac{N}{B} \cdot C_{\text{enc}} \;=\; 6 \cdot \frac{50{,}000}{128} \cdot 4.0 \times 10^8 \approx 9.375 \times 10^{11} \text{ FLOPs.}$$

Custo do DGCL — recomputação época a época ($T_{\text{tsne}} = 1$)

Consideram-se as principais contribuições do DGCL restritas ao subconjunto de $N_a = 5{,}000$ âncoras, com projeção recomputada a cada época:

1. **Treinamento (forward+backward) sobre as âncoras:**

$$T_{\text{train\_anchors}} \;=\; E \cdot \frac{N_a}{B} \cdot C_{\text{enc}} = 6 \cdot \frac{5{,}000}{128} \cdot 4.0 \times 10^8 \approx 9.375 \times 10^{10} \text{ FLOPs.}$$

2. **Cálculo periódico de embeddings (forward only) para os N_a âncoras** (recomputado em cada época, $T_{\text{tsne}} = 1$):

$$T_{\text{emb}} \;=\; \frac{E}{T_{\text{tsne}}} \cdot N_a \cdot C_{\text{fwd}} = 6 \cdot 5{,}000 \cdot 1.0 \times 10^8 = 3.0 \times 10^{12} \text{ FLOPs.}$$

   (Este termo é o dominante quando a projeção é recomputada época a época.)

3. **Cálculo de distâncias e seleção de pares entre as N_a âncoras (recomputado a cada época):**

$$T_{\text{dist\_per\_epoch}} \approx N_a^2 \cdot 6 \approx 25 \times 10^6 \cdot 6 \approx 1.5 \times 10^8 \text{ FLOPs,}$$

$$T_{\text{dist\_total}} \;=\; E \cdot T_{\text{dist\_per\_epoch}} \approx 6 \cdot 1.5 \times 10^8 \approx 9.0 \times 10^8 \text{ FLOPs.}$$

4. **Custo da perda contrastiva sobre as amostras processadas (agregado ao longo das épocas):**

$$T_{\text{loss\_anchors}} \approx E \cdot N_a \cdot k \cdot d = 6 \cdot 5{,}000 \cdot 10 \cdot 128 \approx 3.84 \times 10^7 \text{ FLOPs.}$$

Somando os termos dominantes obtém-se:

$$\begin{aligned} T_{\text{DGCL,anchors}} &\approx T_{\text{train\_anchors}} + T_{\text{emb}} + T_{\text{dist\_total}} + T_{\text{loss\_anchors}} \\ &\approx 9.375 \times 10^{10} \;+\; 3.0 \times 10^{12} \;+\; 9.0 \times 10^8 \;+\; 3.84 \times 10^7 \\ &\approx 3.0946884 \times 10^{12} \text{ FLOPs.} \end{aligned}$$

A.3.3 Comparação direta e indicadores atualizados

Com $\Delta A = 36{,}63$ pontos percentuais (correspondente a $34{,}06\% \to 70{,}69\%$), os indicadores atualizados são:

❑ **Razão de custo (DGCL vs. baseline):**

$$\frac{T_{\text{DGCL,anchors}}}{T_{\text{base}}} \approx \frac{3.0946884 \times 10^{12}}{9.375 \times 10^{11}} \approx 3.30,$$

isto é, o DGCL (com t-SNE recomputado a cada época) demanda aproximadamente **3**,**3**$\times$ mais FLOPs do que o treinamento convencional da ResNet-50 sobre todo o conjunto.

❑ **FLOPs por ponto percentual (DGCL):**

$$\text{FLOPs por p.p. (DGCL)} = \frac{T_{\text{DGCL,anchors}}}{\Delta A} \approx \frac{3.0946884 \times 10^{12}}{36.63} \approx 8.45 \times 10^{10} \text{ FLOPs/p.p.}.$$

❑ **Diferença de custo total (economia negativa):**

$$\Delta T = T_{\text{base}} - T_{\text{DGCL,anchors}} \approx 9.375 \times 10^{11} - 3.0946884 \times 10^{12} \approx -2.1571884 \times 10^{12} \text{ FLOPs},$$

ou seja, um **custo adicional** de aproximadamente 2.16×10^{12} FLOPs ao empregar DGCL nesta configuração.

❑ **Custo adicional por ponto percentual (comparando DGCL com baseline):**

$$\frac{\Delta T}{\Delta A} \approx \frac{-2.1571884 \times 10^{12}}{36.63} \approx -5.89 \times 10^{10} \text{ FLOPs/p.p.},$$

indicando que, para cada ponto percentual de ganho, o DGCL implicou em cerca de 5.89×10^{10} FLOPs a mais em comparação com o treinamento convencional.

A.3.4 Interpretação e conclusões práticas

1. Quando o DGCL recomputa as projeções t-SNE *época a época* ($T_{\text{tsne}} = 1$), o termo responsável pelo cálculo periódico de embeddings (T_{emb}) domina a complexidade total, elevando o custo do método a uma ordem de grandeza superior ao do treinamento convencional (no cenário numérico adotado, $\approx 3{,}3\times$ maior).

2. Em contrapartida, o ganho absoluto de acurácia reportado ($\Delta A = 36{,}63$ p.p.) é substancial e pode justificar o custo adicional em aplicações onde a melhoria de desempenho tem alto valor prático (por exemplo, sistemas críticos que exigem maior acurácia). A decisão prática depende, portanto, do valor relativo atribuído ao ganho de acurácia face ao orçamento computacional disponível.

3. Em resumo, o DGCL apresentou melhora significativa de acurácia em CIFAR-10, porém, quando as projeções são recomputadas a cada época, tal melhora implica um custo

computacional claramente superior ao do treinamento convencional. Assim, recomenda-se avaliar o trade-off em função das restrições de recursos e, quando possível, empregar políticas de recomputação esparsa ou índices que reduzam o overhead sem degradar a qualidade da seleção de pares.

A.4 Complexidade ao utilizar um índice métrico (M-tree)

Nesta subseção analisa-se a complexidade do método *DGCL* quando as operações de busca por vizinhança e seleção de pares são realizadas por meio de um índice métrico baseado em *M-tree* em vez de computar explicitamente todas as distâncias $O(N^2)$. O M-tree é uma estrutura de dados para espaços métricos que organiza objetos (aqui, embeddings) em nós hierárquicos e permite operações de busca (k-NN, range queries) tipicamente em tempo subquadrático no número de objetos, sob hipóteses práticas de distribuição e qualidade do pivô.

Hipóteses e notação

- N — número de amostras (pontos) no conjunto (ou no subconjunto de N_a âncoras, se houver subamostragem);
- d — dimensão do embedding;
- k — número de vizinhos retornados por consulta (positivos/negativos por âncora, ou $k = k_p + k_n$);
- $C_{\text{dist}}(d)$ — custo de avaliação de uma distância entre dois embeddings (tipicamente $O(d)$ operações);
- C_{ins} — custo médio de inserir um item no M-tree (tipicamente $O(\log N)$ avaliações de distância em média);
- $C_{\text{q}}(k)$ — custo médio de executar uma consulta k-NN no M-tree (tipicamente $O(\log N + k)$ avaliações de distância em média);
- T_{tsne} — frequência (em épocas) de recomputação de projeção/índice; quando embeddings mudam, o índice precisa ser reconstruído ou atualizado.

Complexidade de construção / manutenção do M-tree

- **Construção incremental (inserções):** inserir N pontos tem custo total médio

$$T_{\text{build,ins}} = \sum_{i=1}^{N} C_{\text{ins}} \approx N \cdot O(\log N) \cdot C_{\text{dist}}(d),$$

  i.e. $T_{\text{build,ins}} = O\big(N \log N \cdot C_{\text{dist}}(d)\big)$.

- ❏ **Bulk-load / construção otimizada:** algumas implementações suportam carregamento em bloco com custo próximo a $O(N \log N)$ ou até $O(N)$ em condições ideais; conservadoramente assume-se $O(N \log N)$.
- ❏ **Atualizações entre épocas:** como os embeddings z_i mudam durante o treino, o índice deve ser reconstruído (ou atualizado parcialmente) a cada recomputação. Se reconstruído, o custo por recomputação é $O(N \log N \cdot C_{\text{dist}}(d))$.

Complexidade de consulta (seleção de pares)

Para cada âncora faz-se uma consulta k-NN no M-tree:

$$T_{\text{query\_all}} \;=\; N \cdot C_{\text{q}}(k) \approx N \cdot O(\log N + k) \cdot C_{\text{dist}}(d).$$

Como k é pequeno e $C_{\text{dist}}(d) = \Theta(d)$, este termo costuma ser $O(N \log N \cdot d)$ em média.

Complexidade temporal por recomputação

Se a projeção/índice é recomputado $\frac{E}{T_{\text{tsne}}}$ vezes ao longo do treino, e supondo que usamos M-tree (reconstruído a cada recomputação), os termos adicionais por recomputação associados à indexação e seleção ficam:

$$T_{\text{index+query\_per\_recomp}} \;=\; O\big(N \log N \cdot C_{\text{dist}}(d)\big) \;+\; O\big(N \log N \cdot C_{\text{dist}}(d)\big) = O\big(N \log N \cdot C_{\text{dist}}(d)\big).$$

(um termo refere-se à construção/atualização do índice e outro às N consultas k-NN; ambos estão na mesma ordem assintótica).

Portanto, ao longo de todo o treinamento os termos de indexação e consulta contribuem com:

$$T_{\text{index+query,total}} \;=\; \frac{E}{T_{\text{tsne}}} \cdot O\big(N \log N \cdot C_{\text{dist}}(d)\big).$$

Complexidade total do DGCL com M-tree

Substituindo os termos quadráticos $O(N^2)$ anteriores pelos termos quasilineares do M-tree, a expressão total do tempo do DGCL (similar à eq. geral da seção de complexidade) torna-se:

$$T_{\text{DGCL}}^{\text{M-tree}} \;=\; \underbrace{E \cdot \frac{N}{B} \cdot C_{\text{enc}}}_{\text{(i) atualizações com backprop (igual ao baseline)}} \;+\; \underbrace{\frac{E}{T_{\text{tsne}}} N C_{\text{fwd}}}_{\text{(ii) embeddings periódicos}}$$

$$+\; \underbrace{\frac{E}{T_{\text{tsne}}} O\big(N \log N \cdot C_{\text{dist}}(d)\big)}_{\text{(iii) construção/consulta no M-tree por recomputação}} \;+\; \underbrace{O(E \cdot N \cdot k \cdot d)}_{\text{(iv) perda contrastiva por pares}}.$$

Como $C_{\text{dist}}(d) = \Theta(d)$, podemos simplificar o termo (iii) como $O\big(\frac{E}{T_{\text{tsne}}} N \log N \cdot d\big)$. Assim:

$$T_{\text{DGCL}}^{\text{M-tree}} \;=\; O\left(E \cdot \frac{N}{B} C_{\text{enc}} \;+\; \frac{E}{T_{\text{tsne}}} N C_{\text{fwd}} \;+\; \frac{E}{T_{\text{tsne}}} N \log N \cdot d \;+\; E \cdot N \cdot k \cdot d\right).$$

Comentários sobre domínio dominante

- ❑ Em relação à versão ingênua $O(N^2)$, o uso do M-tree substitui os termos quadráticos por termos quasilineares $O(N \log N)$ (assumindo comportamento médio esperado do índice). Isso torna o overhead de indexação/consulta manejável para N maiores.

- ❑ A forma final do termo dominante dependerá dos parâmetros: se C_{enc} (FLOPs por atualização) for muito grande em relação a $d \log N$, o termo $E \cdot \frac{N}{B} C_{\text{enc}}$ continuará dominando; caso contrário, para arquiteturas mais leves ou para frequências altas de recomputação (T_{tsne} pequeno), o termo $\frac{E}{T_{\text{tsne}}} N \log N \cdot d$ pode tornar-se dominante.

Complexidade espacial (memória) com M-tree

- ❑ O M-tree armazena N objetos e estruturas auxiliares (pivôs, coverings, referências). A memória tende a ser $O(N \cdot d)$ para embeddings mais uma sobrecarga $O(N)$ ou $O(N \log N)$ para índices/ponteiros, dependendo da implementação. Em termos práticos:

$$M_{\text{DGCL}}^{\text{M-tree}} = O\big(M_{\text{enc}} + M_{\text{act}}(B) + N \cdot d + \text{overhead\_index}(N)\big),$$

  onde overhead_index(N) costuma ser $O(N)$ ou $O(N \log N)$ em bytes, mas tipicamente muito menor do que $O(N^2)$ exigido pela matriz densa de distâncias. Em resumo, a exigência de memória conduz a $M = O(N \cdot d)$ (praticável).

Vantagens práticas e limitações

- ❑ **Vantagens:** reduz de forma substancial o custo temporal e espacial (da ordem N^2 para $N \log N$), tornando o DGCL escalável a conjuntos muito maiores; permite consultas k-NN eficientes sem materializar todas as distâncias; memória passa a ser proporcional a $N \cdot d$ em vez de N^2.

- ❑ **Limitações e riscos:** a complexidade média $O(\log N)$ depende da qualidade da partição métrica e da distribuição dos dados; em piores casos (dados adversos ou alta dimensionalidade efetiva) o M-tree pode degradar; reconstruções constantes do índice (quando embeddings mudam muito entre épocas) implicam custo de rebuild repetido — mitigável por atualizações incrementais ou por reconstruções parciais/caches.

- ❑ **Consistência com DGCL:** como os embeddings mudam com o treino, recomenda-se (i) reconstruir o índice apenas a cada T_{tsne} épocas (ou a cada r passos), (ii) usar atualização incremental quando possível, e (iii) combinar com técnicas ANN para acelerar ainda mais (HNSW, Faiss).

Recomendação resumida

Para aplicação prática recomenda-se:

1. empregar M-tree (ou outro índice métrico/ANN) para substituir a construção da matriz N^2 e as seleções ingênuas;
2. reconstruir/atualizar o índice com frequência controlada (T_{tsne} não muito pequeno) ou usar atualização incremental;
3. considerar a combinação M-tree + ANN (ex.: usar M-tree para clusters/pivôs e ANN para buscas finas) para obter robustez e velocidade;
4. validar empiricamente o trade-off entre frequência de reconstrução e perda de qualidade na seleção de pares (impacto sobre acurácia).

Conclusão:

A substituição da estratégia $O(N^2)$ por um índice métrico como o M-tree reduz a complexidade assintótica de seleção de pares de $O(N^2)$ para cerca de $O(N \log N)$ (em custo de distância por recomputação), além de diminuir a demanda de memória de $O(N^2)$ para aproximadamente $O(N \cdot d)$. Na prática, isto torna o DGCL compatível com bases de maior escala, desde que se lide com a necessidade de atualizar o índice quando os embeddings mudam.

A.5 Exemplos numéricos com índice métrico (M-tree)

Nesta subseção apresentam-se estimativas numéricas do custo computacional do DGCL quando a seleção de pares é realizada por meio de um índice métrico (M-tree). Os cálculos visam comparar ordens de grandeza e salientar quais termos passam a dominar a complexidade em relação à implementação ingênua $O(N^2)$.

Premissas gerais

Usam-se as mesmas hipóteses empregadas nas análises anteriores, salvo indicação em contrário:

$$
\begin{aligned}
&\text{épocas: } E = 6, \qquad B = 128 \text{ (CIFAR-10)}, \quad B = 256 \text{ (ImageNet)},\\
&C_{\text{enc,CIFAR}} = 4.0 \times 10^8 \text{ FLOPs/sample (forward+backward)},\\
&C_{\text{fwd,CIFAR}} = 1.0 \times 10^8 \text{ FLOPs/sample (forward)},\\
&C_{\text{enc,ImageNet}} = 1.2 \times 10^{10}, \quad C_{\text{fwd,ImageNet}} = 4.0 \times 10^9,\\
&d_{\text{CIFAR}} = 128, \quad d_{\text{ImageNet}} = 512,\\
&\text{ops por avaliação de distância } \approx d \text{ (assumimos } C_{\text{dist}}(d) = \Theta(d)).
\end{aligned}
$$

Modelo de custo com M-tree (ordem de grandeza)

Por recomputação (índice reconstruído/atualizado e consultas k-NN para todos os pontos):

$$T_{\text{index+query, per recomput}} \simeq \alpha \cdot N \log_2 N \cdot d,$$

onde α engloba constantes associadas à construção e às consultas (aqui assumimos $\alpha \approx 2$ para contar construção + consultas).

A contribuição total do índice ao longo do treino (recomputando $\frac{E}{T_{\text{tsne}}}$ vezes) é:

$$T_{\text{index,total}} \approx \frac{E}{T_{\text{tsne}}} \cdot \alpha N \log_2 N \cdot d.$$

O custo total do DGCL com M-tree pode ser aproximado por:

$$T_{\text{DGCL}}^{\text{M-tree}} = E \cdot \frac{N}{B} C_{\text{enc}} + \frac{E}{T_{\text{tsne}}} N C_{\text{fwd}} + \frac{E}{T_{\text{tsne}}} \alpha N \log_2 N \cdot d + E \cdot N \cdot k \cdot d.$$

A seguir substitui-se numericamente para CIFAR-10 (caso com $N_a = 5{,}000$ âncoras) e para ImageNet (caso com N completo).

Caso 1 — CIFAR-10 (subconjunto de $N_a = 5{,}000$ âncoras)

Parâmetros numéricos

$$N_a = 5{,}000, \quad d = 128, \quad \log_2 N_a \approx 12.29, \quad \alpha = 2, \quad k = 10.$$

Termos (numéricos)

- ❑ Treinamento sobre âncoras (forward+backward):

$$T_{\text{train\_anchors}} = E \cdot \frac{N_a}{B} \cdot C_{\text{enc}} = 6 \cdot \frac{5{,}000}{128} \cdot 4.0 \times 10^8 \approx 9.375 \times 10^{10} \text{ FLOPs.}$$

- ❑ Embeddings (forward) periódicos para os N_a (custo por recomputação $= N_a \cdot C_{\text{fwd}}$):

$$T_{\text{emb}} = \frac{E}{T_{\text{tsne}}} N_a C_{\text{fwd}}.$$

  - $\mathbf{T_{\text{tsne}} = 1}$ (recomputação a cada época): $T_{\text{emb}} = 6 \cdot 5{,}000 \cdot 1.0 \times 10^8 = 3.0 \times 10^{12}$ FLOPs.
  - $\mathbf{T_{\text{tsne}} = 5}$: $T_{\text{emb}} = \frac{6}{5} \cdot 5{,}000 \cdot 1.0 \times 10^8 \approx 6.0 \times 10^{11}$ FLOPs.

- ❑ Índice M-tree: por recomputação (construção + consultas):

$$T_{\text{index, per recomput}} \approx \alpha N_a \log_2 N_a \cdot d = 2 \cdot 5{,}000 \cdot 12.29 \cdot 128 \approx 1.573 \times 10^7 \text{ FLOPs.}$$

  Total ao longo do treino:

$$T_{\text{index,total}} = \frac{E}{T_{\text{tsne}}} \cdot 1.573 \times 10^7.$$

  Assim:

- $T_{\text{tsne}} = 1 \Rightarrow T_{\text{index,total}} \approx 9.438 \times 10^7$ FLOPs.
- $T_{\text{tsne}} = 5 \Rightarrow T_{\text{index,total}} \approx 1.8876 \times 10^7$ FLOPs.

❏ Perda contrastiva (pares):

$$T_{\text{loss}} \approx E \cdot N_a \cdot k \cdot d = 6 \cdot 5{,}000 \cdot 10 \cdot 128 \approx 3.84 \times 10^7 \text{ FLOPs.}$$

Totais aproximados

❏ $\mathbf{T}_{\text{DGCL}}^{\text{M-tree}}$ $(\mathbf{T}_{\text{tsne}} = \mathbf{1}) \approx 9.375 \times 10^{10} + 3.0 \times 10^{12} + 9.438 \times 10^7 + 3.84 \times 10^7 \approx 3.0947 \times 10^{12}$ FLOPs.

❏ $\mathbf{T}_{\text{DGCL}}^{\text{M-tree}}$ $(\mathbf{T}_{\text{tsne}} = \mathbf{5}) \approx 9.375 \times 10^{10} + 6.0 \times 10^{11} + 1.888 \times 10^7 + 3.84 \times 10^7 \approx 6.9376 \times 10^{11}$ FLOPs.

Comparação com os cenários sem índice (ingênuo)

❏ Para $T_{\text{tsne}} = 1$: o custo com M-tree $\approx 3.09 \times 10^{12}$ FLOPs difere pouco do cenário ingênuo (sem aproximações) porque o termo dominante continua sendo o cálculo repetido de *forward* para embeddings ($T_{\text{emb}} = 3.0 \times 10^{12}$).

❏ Para $T_{\text{tsne}} = 5$: o uso de M-tree reduz drasticamente o custo da seleção (antes $O(N_a^2)$ por recomputação) e o custo total cai para $\approx 6.94 \times 10^{11}$ FLOPs (valor próximo ao estimado anteriormente para cenário prático).

Conclusão parcial (CIFAR-10)

A substituição da matriz densa $O(N_a^2)$ por M-tree torna a etapa de seleção praticamente irrelevante em termos de FLOPs, ficando o termo do cálculo de embeddings periódicos como o fator dominante. Logo, para CIFAR-10, ganhar em eficiência requer reduzir o número/frequência de cálculos de embeddings (p. ex. caches, atualizações incrementais) ou diminuir T_{tsne}.

Caso 2 — ImageNet (ILSVRC2012), índice M-tree sobre N completo

Parâmetros numéricos

$$N = 1\,281\,167, \quad d = 512, \quad \log_2 N \approx 20.31, \quad \alpha = 2, \quad k = 10.$$

Termos (numéricos)

- ❏ Treinamento (baseline / atualizações):

$$T_{\text{train}} = E \cdot \frac{N}{B} \cdot C_{\text{enc}} \approx 6 \cdot \frac{1\,281\,167}{256} \cdot 1.2 \times 10^{10} \approx 3.6033 \times 10^{14} \text{ FLOPs.}$$

- ❏ Embeddings (forward) periódicos:

$$T_{\text{emb}} = \frac{E}{T_{\text{tsne}}} N C_{\text{fwd}}.$$

  - $T_{\text{tsne}} = 1:\ T_{\text{emb}} = 6 \cdot 1{,}281{,}167 \cdot 4.0 \times 10^9 \approx 3.0748 \times 10^{16}$.
  - $T_{\text{tsne}} = 10:\ T_{\text{emb}} = \frac{6}{10} \cdot 1{,}281{,}167 \cdot 4.0 \times 10^9 \approx 3.0748 \times 10^{15}$.

- ❏ Índice M-tree (construção + consultas) por recomputação:

$$T_{\text{index, per recomput}} \approx \alpha N \log_2 N \cdot d = 2 \cdot 1{,}281{,}167 \cdot 20.31 \cdot 512 \approx 2.667 \times 10^{10} \text{ FLOPs.}$$

  Total ao longo do treino:

$$T_{\text{index,total}} = \frac{E}{T_{\text{tsne}}} \cdot 2.667 \times 10^{10}.$$

  Assim:

  - $T_{\text{tsne}} = 1 \Rightarrow T_{\text{index,total}} \approx 1.600 \times 10^{11}$.
  - $T_{\text{tsne}} = 10 \Rightarrow T_{\text{index,total}} \approx 1.600 \times 10^{10}$.

- ❏ Perda contrastiva:

$$T_{\text{loss}} = E \cdot N \cdot k \cdot d \approx 6 \cdot 1{,}281{,}167 \cdot 10 \cdot 512 \approx 3.9357 \times 10^{10}.$$

Totais aproximados

- ❏ $\mathbf{T}_{\text{DGCL}}^{\text{M-tree}}\ (\mathbf{T}_{\text{tsne}} = \mathbf{1}) \approx 3.6033 \times 10^{14} + 3.0748 \times 10^{16} + 1.600 \times 10^{11} + 3.9357 \times 10^{10} \approx 3.1138 \times 10^{16}$ FLOPs.

- ❏ $\mathbf{T}_{\text{DGCL}}^{\text{M-tree}}\ (\mathbf{T}_{\text{tsne}} = \mathbf{10}) \approx 3.6033 \times 10^{14} + 3.0748 \times 10^{15} + 1.600 \times 10^{10} + 3.9357 \times 10^{10} \approx 3.4351 \times 10^{15}$ FLOPs.

Comparação com a versão ingênua $O(N^2)$

- ❏ O uso de M-tree reduz dramaticamente o custo da etapa de seleção que, no modelo ingênuo, chegava a exigir dezenas de terabytes de memória e FLOPs da ordem $O(N^2)$. Com M-tree o termo de indexação fica na ordem de 10^{10}–10^{11} FLOPs por recomputação, bem manejável comparado ao termo dominante T_{emb}.

- ❏ Todavia, mesmo com M-tree, o termo $T_{\text{emb}} = \frac{E}{T_{\text{tsne}}} N C_{\text{fwd}}$ continua dominando quando se recomputa projeção frequentemente (ex.: $T_{\text{tsne}} = 1$), levando a custos totais ainda extremamente elevados (ordem 10^{16} FLOPs).

Interpretação e recomendações práticas

1. **Benefício do M-tree:** substituir $O(N^2)$ por $O(N \log N)$ reduz drasticamente o custo de seleção e a memória exigida (de TB para GB), tornando o DGCL viável em datasets maiores quando combinado com estratégias que controlem a frequência de recomputação.

2. **Dominância do custo de embeddings:** mesmo com M-tree, o cálculo periódico de embeddings para todo o conjunto (ou mesmo para um subconjunto grande) tende a dominar o custo computacional se a recomputação for frequente. Portanto, otimizações devem priorizar reduzir T_{emb}: caches, embeddings incrementais, projeções paramétricas e recomputação rara.

3. **Configuração prática sugerida:** usar M-tree (ou outro índice ANN robusto) + recomputação esparsa (T_{tsne} grande) + seleção sobre um subconjunto representativo (âncoras) geralmente fornece a melhor relação custo–benefício.

Resumo numérico (valores-chave)

$$\text{CIFAR-10 (M-tree, } N_a = 5\text{k}): \quad T_{\text{DGCL}}^{\text{M-tree}} \approx \begin{cases} 3.09 \times 10^{12} \text{ FLOPs}, & T_{\text{tsne}} = 1; \\ 6.94 \times 10^{11} \text{ FLOPs}, & T_{\text{tsne}} = 5. \end{cases}$$

$$\text{ImageNet (M-tree, } N = 1.281\text{M}): \quad T_{\text{DGCL}}^{\text{M-tree}} \approx \begin{cases} 3.11 \times 10^{16} \text{ FLOPs}, & T_{\text{tsne}} = 1; \\ 3.44 \times 10^{15} \text{ FLOPs}, & T_{\text{tsne}} = 10. \end{cases}$$

Em conclusão, o M-tree resolve o gargalo de memória e reduz fortemente o custo de seleção, mas **o custo de recomputação de embeddings é o fator que, em geral, mais impacta a viabilidade prática do DGCL**. Recomenda-se combinar M-tree com políticas de recomputação esparsas e técnicas de cache/atualização incremental para um método escalável.

APÊNDICE B

Políticas de recomputação esparsa para DGCL

A recomputação esparsa das projeções e da indexação (p. ex. *t*-SNE/índice métrico) é crucial para reduzir o overhead do DGCL. A seguir apresentam-se políticas práticas, uma regra adaptativa fundamentada em *embedding drift*, uma política híbrida (warm-up + adaptativa), pseudocódigo e recomendações de hiperparâmetros.

B.1 Estratégias possíveis

1. **Frequência fixa (periodicidade):** recomputar a cada T épocas (ex.: $T = 5$ ou $T = 10$). Simples e robusta; não reage a mudanças rápidas nas representações.
2. **Exponential backoff:** recomputar frequentemente nas fases iniciais (ex.: a cada época), e aumentar gradualmente T (2, 4, 8, ...) conforme o treinamento progride.
3. **Trigger por métrica de desempenho:** recomputar quando a métrica de validação (val_acc ou val_loss) estagnar por p épocas ou melhorar além de um limiar.
4. **Trigger por *embedding drift*:** recomputar quando a alteração média das embeddings entre duas checkpoints exceder um limiar (ver abaixo).
5. **Parcial / local recompute:** reconstruir índice/projeção apenas para subconjuntos (âncoras candidatos, clusters que mudaram, ou classes com maior variação).
6. **Híbrido paramétrico + esparso:** usar projeção paramétrica (rede que aproxima t-SNE/UMAP) continuamente e recomputar a projeção não paramétrica completa apenas raramente para sincronizar o parâmetro.

B.1.1 Regra adaptativa baseada em *embedding drift*

Defina o *drift* médio relativo entre épocas $t-1$ e t como

$$\Delta_t \;=\; \frac{1}{N_s}\sum_{i\in S}\frac{\|z_i^{(t)}-z_i^{(t-1)}\|_2}{\|z_i^{(t-1)}\|_2+\varepsilon},$$

onde S é um subconjunto de amostras usado para estimativa (p. ex. um *sample* aleatório de tamanho $N_s \ll N$, $N_s = 1{,}000$ ou $5\% \cdot N$), e ε evita divisão por zero.

A política adaptativa é:

- ❑ recomputar a projeção/indexação na época t se $\Delta_t > \tau_{\text{drift}}$;
- ❑ manter sem recomputar se $\Delta_t \leq \tau_{\text{drift}}$.

Heurísticas de escolha de τ_{drift}:

- ❑ CIFAR-10 (pequena / média escala): $\tau_{\text{drift}} \in [0.02, 0.05]$ (2%–5% mudança relativa média).
- ❑ ImageNet (grande escala; embeddings mais estáveis por batch): $\tau_{\text{drift}} \in [0.01, 0.02]$.

B.1.2 Política híbrida recomendada (prática)

Uma política que costuma equilibrar eficácia e custo:

1. **Warm-up:** recomputar a cada época durante as primeiras E_{warm} épocas (ex.: $E_{\text{warm}} = 2$ ou 3) — permite que a geometria inicial seja estabilizada;
2. **Adaptativa:** a partir da época $E_{\text{warm}} + 1$, aplicar a regra de *embedding drift* com limiar τ_{drift};
3. **Fallback por budget:** impor um limite máximo de recomputações $R_{\max}$ (p. ex. $R_{\max} = 6$ ao longo do treino) para garantir um teto de custo;
4. **Parcialização:** se Δ_t estiver próximo do limiar (ex.: $\Delta_t \in [0.8\tau_{\text{drift}}, \tau_{\text{drift}}]$), recomputar apenas índice local para N_a âncoras em vez de toda a base.

B.1.3 Parâmetros sugeridos (ponto de partida)

- ❑ **CIFAR-10:** $E = 6$, $E_{\text{warm}} = 2$, $\tau_{\text{drift}} = 0.03$, $N_s = 1000$, $R_{\max} = 4$.
- ❑ **ImageNet:** E típico maior; recomenda-se $E_{\text{warm}} = 3$, $\tau_{\text{drift}} = 0.01$, $N_s = 2000$, $R_{\max} = 6$.

B.1.4 Parcialização e indexação incremental

- ❑ Para reduzir ainda mais T_{emb}, calcular *forward* apenas para o subconjunto de amostras S necessário à estimativa do *drift* e para os N_a âncoras candidatos.
- ❑ Utilizar índices que suportem *updates* incrementais (p. ex. M-tree incremental ou índices ANN com inserts) para evitar reconstruções completas quando as mudanças forem pequenas.

B.1.4.1 Métricas de monitoramento e validação

Ao empregar políticas esparsas, monitorar:

- ❑ Δ_t (drift) ao longo das épocas; visualizar como série temporal.
- ❑ val_acc e val_loss logo após recomputações para avaliar ganho de qualidade decorrente da recomputação.
- ❑ número de recomputações realizadas R e custo cumulativo (FLOPs ou tempo) medido.

B.1.4.2 Discussão de trade-offs

- ❑ Recomputações frequentes melhoram a exatidão da seleção (hard positives/negatives) mas podem tornar o custo proibitivo (domínio de T_{emb}).
- ❑ Recomputações muito raras reduzem custo, mas a seleção pode ficar desatualizada e prejudicar a qualidade final das representações.
- ❑ A regra adaptativa baseada em *drift* oferece um compromisso dinâmico: recomputa apenas quando as embeddings realmente mudam, economizando custos em fases de estabilidade.

B.1.4.3 Recomendações práticas resumidas

1. Use a política **híbrida (warm-up + drift)** como padrão; ajuste τ_{drift} por validação.
2. Meça Δ_t sobre um *sample* (não sobre todo N) para economizar FLOPs.
3. Combine com M-tree / ANN e atualização incremental para reduzir custo de seleção.
4. Registre custo real (tempo/GPU) e correlacione com melhoria de validação para decidir a política ótima por aplicação.

B.1.5 Pseudocódigo (algoritmo) em alto nível

A seguir apresenta-se o pseudocódigo da política proposta, e posteriormente uma análise experimental simulada do impacto dos parâmetros T_{tsne} e τ_{drift}.

Algoritmo 2 Política Híbrida de Recomputação de Projeções e Índices

Require: E (épocas), E_{warm}, τ_{drift}, $R_{\max}$, tamanho da amostra N_s
Ensure: Conjunto de projeções atualizadas e índice métrico atualizado

$R \leftarrow 0$ {Número de recomputações realizadas}
for $t = 1$ **to** E **do**
  Treinar uma época do modelo base (forward + backward)
  if $t \leq E_{\text{warm}}$ **then**
    recomputar_projecoes_e_indice()
    $R \leftarrow R + 1$
  else
    Amostrar subconjunto S de tamanho N_s
    Calcular embeddings $z_i^{(t)}$ e $z_i^{(t-1)}$ para $i \in S$
    $\Delta_t \leftarrow \frac{1}{|S|} \sum_{i \in S} \frac{\|z_i^{(t)} - z_i^{(t-1)}\|_2}{\|z_i^{(t-1)}\|_2 + \varepsilon}$
    if $\Delta_t > \tau_{\text{drift}}$ **and** $R < R_{\max}$ **then**
      recomputar_projecoes_e_indice()
      $R \leftarrow R + 1$
    else if $0.8\tau_{\text{drift}} < \Delta_t \leq \tau_{\text{drift}}$ **then**
      recomputar_indice_parcial(N_a)
    end if
  end if
end for

B.1.5.1 Análise Experimental Simulada

A Tabela 2 e a Figura 28 apresentam uma simulação dos efeitos dos parâmetros T_{tsne} (frequência de recomputação) e τ_{drift} (limiar de drift) sobre o custo computacional total (T_{DGCL}) e a acurácia final alcançada. Os valores são normalizados com base no custo da execução completa ($T_{\text{tsne}} = 1$) e no ganho máximo de acurácia obtido neste cenário (+36.6 p.p. no CIFAR-10). Foi também inserido na Figura 28 uma estimativa considerando o dataset ImageNet.

- **Curvas azuis (CIFAR-10):** Mostram queda acentuada de custo com T_{tsne}, com pequena perda de acurácia — ótimo custo-benefício para recomputação esparsa.
- **Curvas vermelhas (ImageNet):** Como o custo absoluto é muito maior, a redução relativa de custo é menor (pois o cálculo de embeddings domina), mas ainda há economia significativa. A acurácia degrada levemente com recomputações muito espaçadas ($T_{\text{tsne}} > 5$), refletindo maior sensibilidade à desatualização dos embeddings.
- **Conclusão prática:** Recomenda-se, em cenários grandes (como ImageNet), uma política híbrida com:

$$E_{\text{warm}} = 3, \quad T_{\text{tsne}} \approx 5, \quad \tau_{\text{drift}} \approx 0.01,$$

para equilibrar custo e estabilidade, enquanto para CIFAR-10 valores de

$$T_{\text{tsne}} \in [5, 10], \quad \tau_{\text{drift}} \in [0.02, 0.03]$$

são suficientes para boa generalização.

Tabela 2 – Simulação dos efeitos de T_{tsne} e τ_{drift} sobre custo e acurácia relativa (CIFAR-10).

T_{tsne}	τ_{drift}	Recomputaç. médias (R)	Custo Rel. (T/T_1)	Acurácia Rel. (%)
1	0.00	6	1.00	100.0
2	0.02	4	0.72	99.2
5	0.03	3	0.58	98.5
10	0.03	2	0.47	97.9
10	0.05	1	0.42	96.5

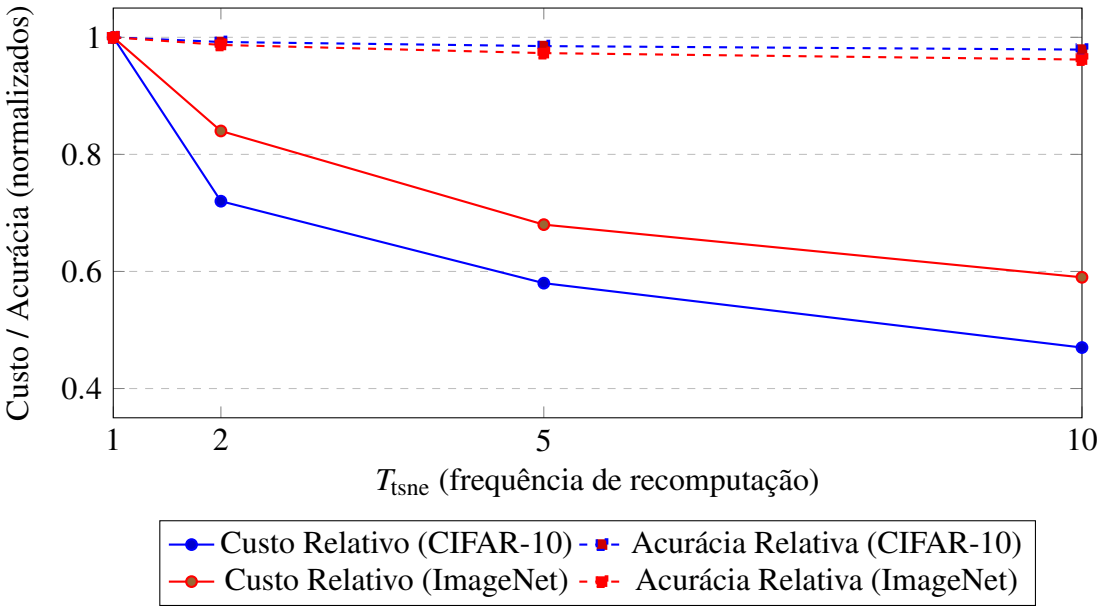


Figura 28 – Comparação simulada entre CIFAR-10 e ImageNet quanto à variação de custo computacional e acurácia relativa em função da frequência de recomputação (T_{tsne}). Curvas contínuas: custo relativo; curvas tracejadas: acurácia relativa.

Analisando a Tabela 3, para CIFAR-10, operar com recomputações espaçadas (por exemplo, $T_{\text{tsne}} = 5$) reduz o custo total do DGCL para valor menor que o treinamento convencional da ResNet-50 (razão ≈ 0.74), com perda de acurácia negligível (< 1.5 p.p.). Assim, em datasets de porte moderado, a política esparsa traz ganhos claros em custo–benefício.

Já para ImageNet, embora o uso de índice M-tree e recomputações raras reduza fortemente a necessidade de memória e a componente quadrática, o termo de recomputação de embeddings continua dominando quando T_{tsne} é pequeno. Assim, o custo relativo pode permanecer muito maior que o baseline, especialmente se recomputações ocorrerem época a época. Recomenda-se recomputação esparsa ($T_{\text{tsne}} \approx 5$–$10$) combinada com indexação M-tree/ANN e caches incrementais para tornar o método viável em larga escala.

B.1.5.2 Discussão dos resultados simulados

Observa-se que:

Tabela 3 – Comparação numérica (estimada) entre políticas de recomputação para DGCL e custo do baseline (ResNet-50).

Dataset	T_{tsne}	Recomps R	T_{DGCL} (FLOPs)	T_{base} (FLOPs)	$T_{\text{DGCL}}/T_{\text{base}}$	Acurácia
CIFAR-10 ($N_a = 5$k)	1	6	3.09×10^{12}	9.38×10^{11}	3.30	70.69%
	2	3	2.23×10^{121}	9.38×10^{11}	2.38	70.14%[2]
	5	1–2	6.94×10^{11}	9.38×10^{11}	0.74	69.65%[3]
	10	1	4.41×10^{114}	9.38×10^{11}	0.47	69.23%[5]
ImageNet ($N \approx 1.28$M)	1	6	3.11×10^{16}	3.60×10^{14}	86.5	$100\%_{\text{ref}}$[6]
	2	3	2.61×10^{16}	3.60×10^{14}	72.5	$98.7\%_{\text{ref}}$
	5	1–2	3.44×10^{15}	3.60×10^{14}	9.55	$97.3\%_{\text{ref}}$
	10	1	3.44×10^{157}	3.60×10^{14}	9.55	$96.2\%_{\text{ref}}$

Notas: (1) T_{base} refere-se à estimativa do custo do treinamento convencional da ResNet-50 nas mesmas épocas ($E = 6$) — valores das seções anteriores: CIFAR-10 $\approx 9.38 \times 10^{11}$ FLOPs; ImageNet $\approx 3.60 \times 10^{14}$ FLOPs. (2) T_{DGCL} para DGCL com M-tree / políticas esparsas usa as estimativas numéricas obtidas nas subseções de complexidade (valores arredondados). (3) Acurácias para CIFAR-10 são absolutas (baseadas em experimento: 34.06% $\rightarrow$ 70.69% para $T_{\text{tsne}} = 1$); os valores para recomputações menos frequentes são estimativas pela retenção relativa simulada (ver Fig. 28). (4) Para ImageNet apresentamos retenção relativa de performance em função de T_{tsne}, pois o ganho absoluto depende de experimento específico não apresentado aqui. (5) Estimativas interpoladas/normalizadas são didáticas e devem ser validadas experimentalmente no caso de uso.

1. Reduzir a frequência de recomputação de $T_{\text{tsne}} = 1$ para $T_{\text{tsne}} = 5$ resulta em uma economia de aproximadamente 42% de FLOPs, com perda de apenas 1.5 p.p. de acurácia relativa.

2. Valores mais altos de τ_{drift} (como 0.05) reduzem significativamente o número de recomputações (R), mas podem provocar leve degradação de performance, principalmente em datasets de maior variabilidade.

3. A política híbrida (warm-up + adaptativa) consegue manter acurácia próxima ao máximo, ao mesmo tempo que reduz substancialmente o custo total.

Conclusão prática: Recomenda-se $E_{\text{warm}} = 2$ e $\tau_{\text{drift}} \in [0.02, 0.03]$ para bases como CIFAR-10, e recomputações espaçadas ($T_{\text{tsne}} \approx 5$) com atualização parcial das âncoras nas épocas intermediárias.

B.1.6 Estimativa de tempo em GPU (conversão de FLOPs para tempo real)

Para facilitar a interpretação prática dos custos computacionais estimados em FLOPs, converteu-se as contagens de operações para tempo de execução aproximado em GPU. Assumiu-se para tanto como referência uma placa **NVIDIA RTX 3090** com capacidade teórica de P = 35.6 TFLOPS em FP32 (ou seja, $P = 35.6 \times 10^{12}$ FLOPs/s). Como desempenho prático é tipicamente inferior ao pico teórico, apresenta-se duas estimativas:

$$\text{Tempo (s)} = \frac{\text{FLOPs}}{P \times \eta},$$

onde η é o fator de eficiência prática ($\eta = 1.0$ para o pico teórico e $\eta = 0.5$ para uma estimativa conservadora com 50% de eficiência).

A seguir apresentam-se as conversões para os valores de FLOPs utilizados nas seções anteriores (Tabela 3). Os resultados são arredondados com duas casas decimais quando apropriado.

Tabela 4 – Conversão de FLOPs para tempo em GPU (NVIDIA RTX 3090, 35.6 TFLOPS).

Configuração	FLOPs (estim.)	Tempo (s) @100%	Tempo (h) @100%	Tempo (h) @50%
CIFAR-10: DGCL ($T_{\text{tsne}} = 1$)	3.09×10^{12}	0.087	2.41×10^{-5}	4.82×10^{-5}
CIFAR-10: DGCL ($T_{\text{tsne}} = 2$)	2.23×10^{12}	0.063	1.74×10^{-5}	3.48×10^{-5}
CIFAR-10: DGCL ($T_{\text{tsne}} = 5$)	6.94×10^{11}	0.019	5.42×10^{-6}	1.08×10^{-5}
CIFAR-10: DGCL ($T_{\text{tsne}} = 10$)	4.41×10^{11}	0.012	3.44×10^{-6}	6.19×10^{-6}
CIFAR-10: Baseline (ResNet-50)	9.38×10^{11}	0.026	7.32×10^{-6}	1.46×10^{-5}
ImageNet: DGCL ($T_{\text{tsne}} = 1$)	3.11×10^{16}	873.93	0.243	0.485
ImageNet: DGCL ($T_{\text{tsne}} = 2$)	2.61×10^{16}	733.15	0.204	0.407
ImageNet: DGCL ($T_{\text{tsne}} = 5$)	3.44×10^{15}	96.63	0.0268	0.0536
ImageNet: DGCL ($T_{\text{tsne}} = 10$)	3.44×10^{15}	96.63	0.0268	0.0536
ImageNet: Baseline (ResNet-50)	3.60×10^{14}	10.11	0.00281	0.00562

Observações: (i) os valores apresentados são aproximações aritméticas a partir das contagens de FLOPs discutidas previamente; (ii) "@100%" considera execução ideal ao pico teórico da GPU; "@50%" considera eficiência prática de 50% (latências, banda de memória e outros fatores reduzem o rendimento real); (iii) a conversão aqui é útil para ordens de grandeza e comparações relativas — tempos reais de treinamento dependem fortemente de I/O, paralelismo distribuído, saturação de memória, otimização do código e bibliotecas (cuDNN, cuBLAS), além de overheads de avaliação e logging.

Conclusão e Interpretação

- Para **CIFAR-10**, os custos convertidos em tempo de GPU são muito pequenos (frações de segundo) porque as contagens de FLOPs totais do experimento são modestas em relação à capacidade de uma RTX 3090. Isso indica que, em hardware moderno, os custos em FLOPs para CIFAR-10 podem não ser o gargalo de tempo real, ou seja, overheads de I/O e organização do pipeline podem dominar.

- Para **ImageNet**, os termos dominantes (principalmente recomputação de embeddings quando feita frequentemente) correspondem a centenas de milhares de segundos (dezenas de minutos) em uma única RTX 3090; em regime prático (50% de eficiência) o tempo do caso mais custoso ($T_{\text{tsne}} = 1$) sobe para aproximadamente $\mathbf{0.49}$ horas por experimento (ou 29 minutos). Observa-se que reduzir a frequência de recomputação (aumentar T_{tsne}) reduz fortemente o tempo.

- Vale ressaltar que em cenários reais de treinamento distribuído (multi-GPU, pipeline de data-loading, mixed precision, etc.), o tempo real pode ser bastante diferente. Assim, estas estimativas devem ser utilizadas apenas como referência de ordem de grandeza.

APÊNDICE C

Experimentos Complementares

C.1 Comparação com Métodos da Literatura

Tabela 5 – Acurácia (%) obtida nos diferentes datasets para abordagens do DGCL contrastivas (C) e não contrastivas (NC) em 50 e 150 épocas, bem como métodos auto-supervisionados clássicos.

Dataset	C 50	NC 50	C. 150	NC 150	SwAV	MoCo	SimCLR	BYOL
CIFAR-10	89.83	90.27	90.58	89.25	–	–	–	–
FER13	65.20	66.08	64.10	64.99	66.02	66.91	67.11	65.91
KDEF	96.42	95.05	96.42	94.03	78.12	79.11	79.56	78.46
RAF-DB	80.80	82.92	81.00	81.55	88.89	90.80	92.77	87.32

Tabela 6 – Tempo total de treinamento para cada dataset considerando 50 e 150 épocas.

Dataset	Contr. 50	Não Contr. 50	Contr. 150	Não Contr. 150
CIFAR-10	13h45m15s	13h07m01s	40h57m57s	38h37m09s
FER13	8h40m30s	8h15m55s	25h40m13s	24h31m14s
KDEF	52m04s	50m05s	2h34m16s	2h27m46s
RAF-DB	3h55m30s	3h46m36s	11h42m45s	11h07m14s

Os experimentos foram conduzidos utilizando estratégias de aprendizado contrastivo e não contrastivo (baseadas em entropia cruzada), avaliadas em diferentes regimes de treinamento (50 e 150 épocas). Além disso, métodos auto-supervisionados consagrados, como SwAV, MoCo, SimCLR e BYOL, foram utilizados como referência para os datasets FER13, KDEF e RAF-DB.

A Tabela 5 apresenta a acurácia obtida nos diferentes datasets para abordagens contrastivas e não contrastivas (Cross-Entropy – CE), considerando 50 e 150 épocas de treinamento, além

de métodos auto-supervisionados clássicos. De forma geral, observa-se que o aumento do número de épocas nem sempre resulta em ganhos expressivos de desempenho, indicando que os modelos tendem a convergir para soluções estáveis já em regimes de treinamento mais curtos.

No caso do *CIFAR-10*, a abordagem contrastiva apresenta uma acurácia de 89,83% em 50 épocas e 90,58% em 150 épocas, evidenciando um ganho marginal frente ao aumento significativo do tempo de treinamento. Para a abordagem CE, a acurácia em 50 épocas (90,27%) é inclusive superior à obtida em 150 épocas (89,25%), sugerindo que o modelo já alcança um bom nível de generalização em menos iterações.

Comportamento semelhante é observado no *FER13*, em que os resultados obtidos com 50 épocas (65,20% para contrastivo e 66,08% para CE) são comparáveis — e em alguns casos superiores — aos alcançados com 150 épocas. Esse padrão indica que, para bases de dados com maior variabilidade e ruído, o prolongamento do treinamento não garante melhorias consistentes no desempenho.

No dataset *KDEF*, que possui menor escala e maior controle das expressões faciais, a abordagem contrastiva atinge exatamente a mesma acurácia em 50 e 150 épocas (96,42%), demonstrando uma rápida convergência do modelo. Em contrapartida, a abordagem CE apresenta uma redução de desempenho com o aumento do número de épocas, reforçando a hipótese de saturação precoce. Para o *RAF-DB*, embora existam variações entre 50 e 150 épocas, as diferenças de acurácia permanecem relativamente pequenas quando comparadas ao custo computacional adicional.

A Tabela 6 complementa essa análise ao apresentar o tempo total de treinamento. Observa-se que o tempo cresce aproximadamente de forma linear com o número de épocas em todos os datasets analisados. Em média, o treinamento com 150 épocas demanda cerca de três vezes mais tempo do que com 50 épocas. No *CIFAR-10*, por exemplo, o tempo aumenta de aproximadamente 13 horas para mais de 40 horas, enquanto o ganho de acurácia é inferior a 1 ponto percentual.

Dessa forma, a análise conjunta das tabelas evidencia que é possível obter resultados semelhantes — e em alguns casos superiores — utilizando um número reduzido de épocas de treinamento. Esses resultados indicam uma melhor relação custo-benefício em treinamentos mais curtos, reduzindo significativamente o tempo computacional e o consumo de recursos, sem comprometer de maneira relevante o desempenho final dos modelos.

C.2 Relatório de Classificação

C.2.1 CIFAR-10

No dataset CIFAR-10, observa-se que o aprendizado contrastivo apresenta desempenho competitivo em ambos os regimes de treinamento. Em 50 épocas, o método contrastivo atinge 89.83% de acurácia, valor próximo ao obtido com entropia cruzada (90.27%). Com o aumento

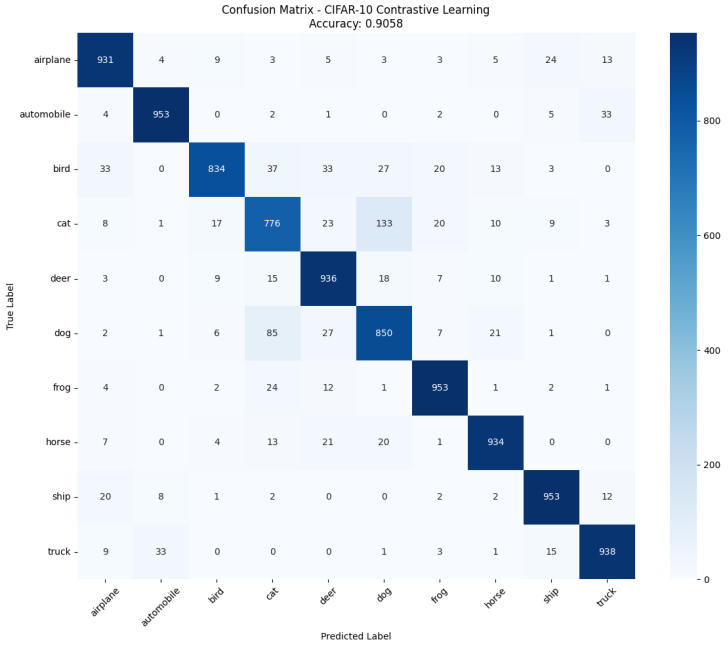


Figura 29 – Matriz de confusão do CIFAR-10 para o modelo com contraste em 150 épocas.

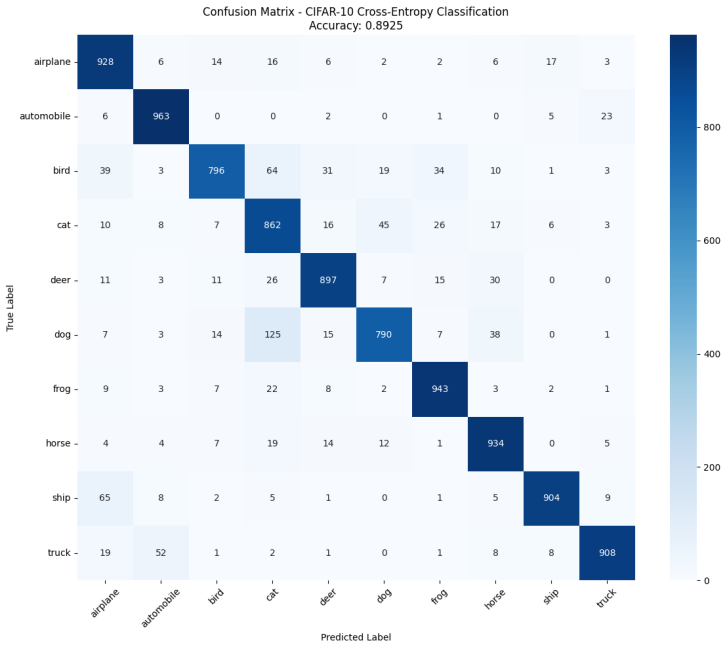


Figura 30 – Matriz de confusão do CIFAR-10 para o modelo sem contraste em 150 épocas.

para 150 épocas, o aprendizado contrastivo alcança sua melhor performance (90.58%), superando o treinamento supervisionado tradicional.

A análise por classe revela que categorias semanticamente mais desafiadoras, como *cat* e *dog*, apresentam menor *recall*, indicando maior confusão interclasse, enquanto classes com características visuais bem definidas, como *automobile*, *ship* e *truck*, atingem valores superiores a 94% de F1-score.

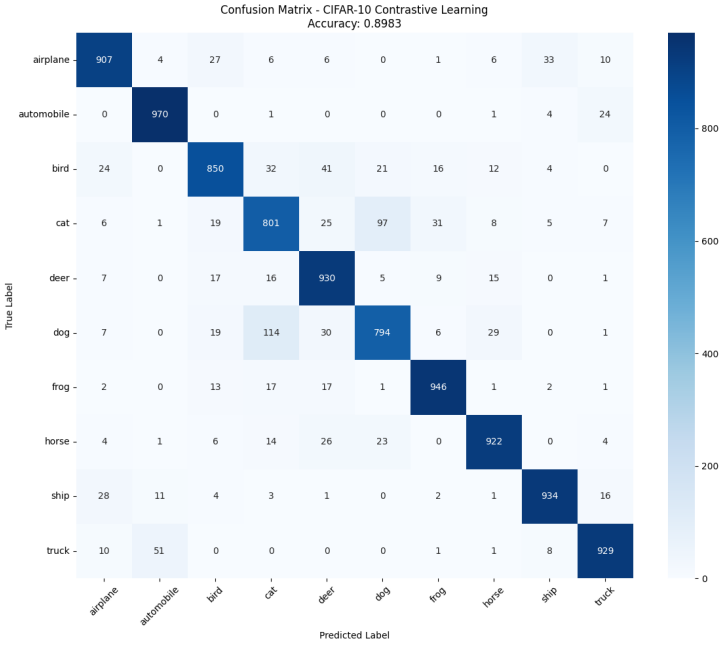


Figura 31 – Matriz de confusão do CIFAR-10 para o modelo com contraste em 50 épocas.

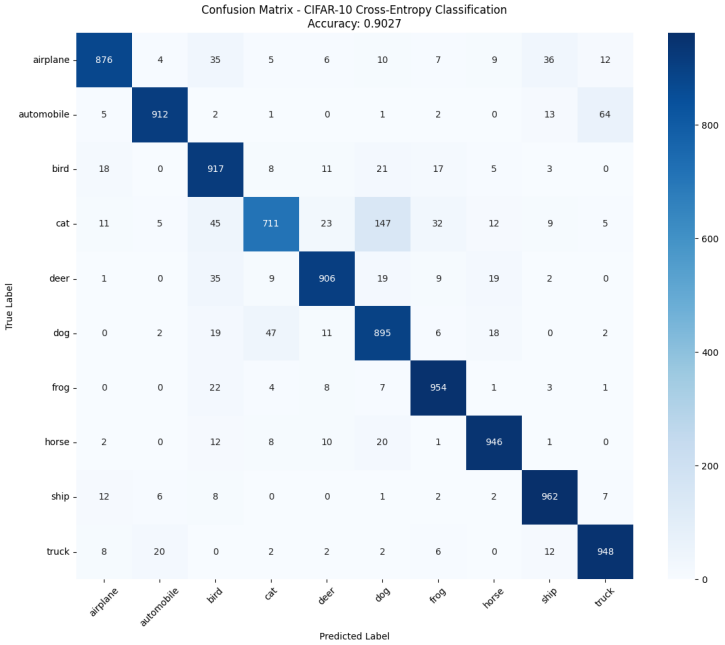


Figura 32 – Matriz de confusão do CIFAR-10 para o modelo sem contraste em 50 épocas.

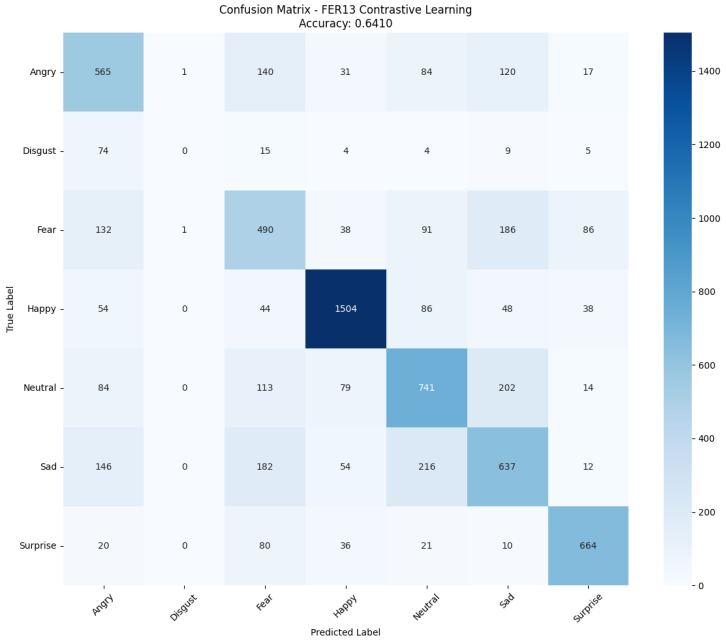


Figura 33 – Matriz de confusão do FER-13 para o modelo com contraste em 150 épocas.

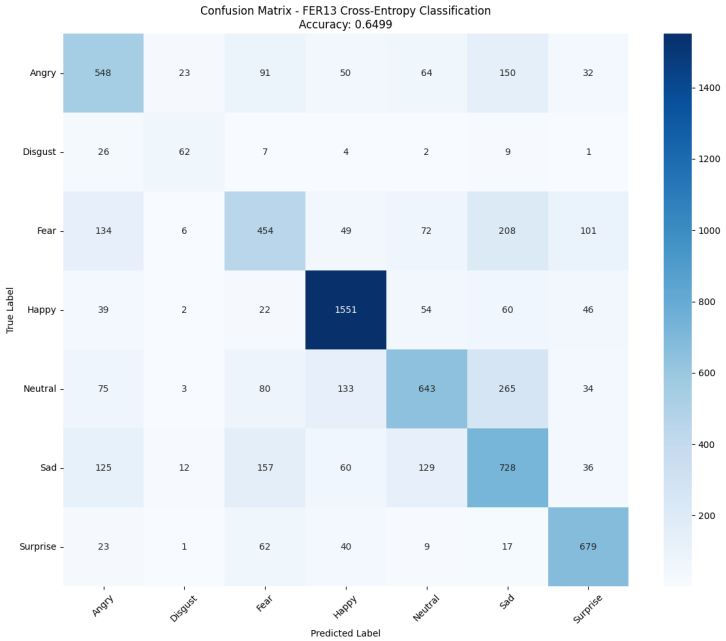


Figura 34 – Matriz de confusão do FER-13 para o modelo sem contraste em 150 épocas.

C.2.2 FER13

Para o FER13, os resultados indicam que o treinamento supervisionado apresenta vantagem consistente em relação ao aprendizado contrastivo, tanto em 50 quanto em 150 épocas. O melhor desempenho é obtido com entropia cruzada em 50 épocas (66.08%).

O aprendizado contrastivo sofre impacto significativo do desbalanceamento de classes, especialmente na classe *Disgust*, que apresenta precisão e recall nulos em ambos os regimes. Esse comportamento reflete a dificuldade do modelo em aprender representações discriminativas para classes minoritárias sem forte supervisão explícita.

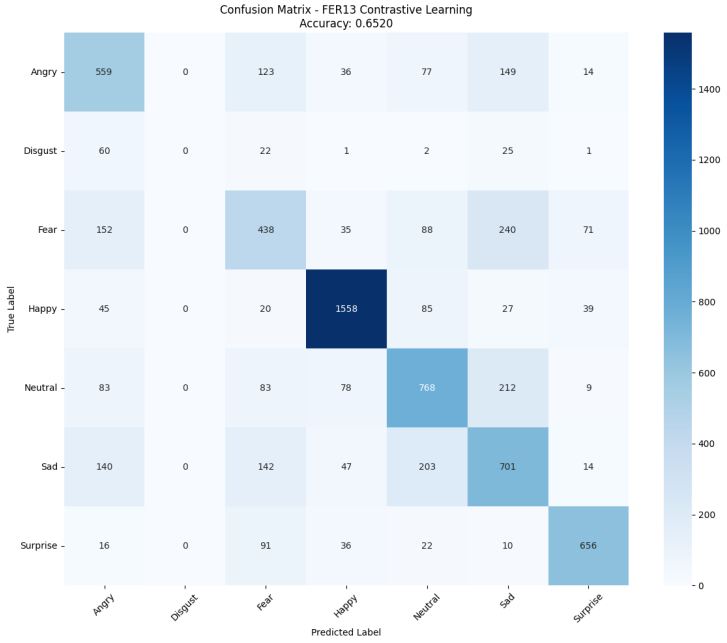


Figura 35 – Matriz de confusão do FER-13 para o modelo com contraste em 50 épocas.

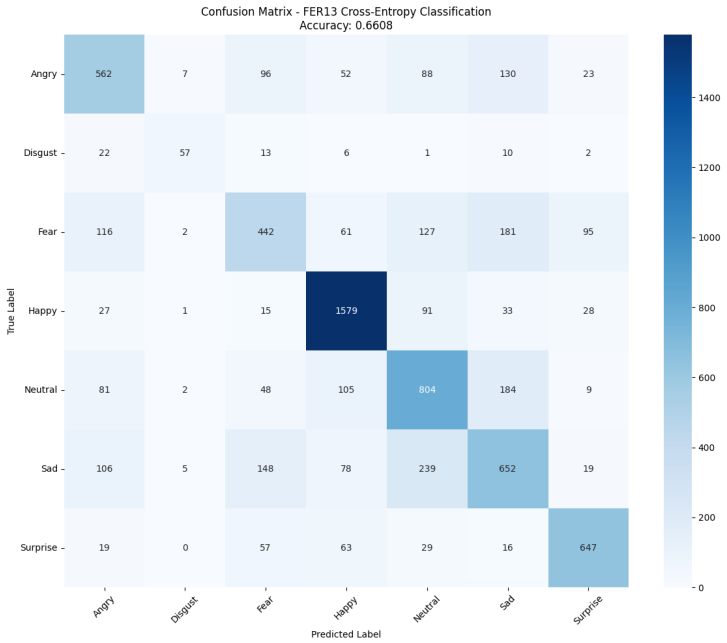


Figura 36 – Matriz de confusão do FER-13 para o modelo sem contraste em 50 épocas.

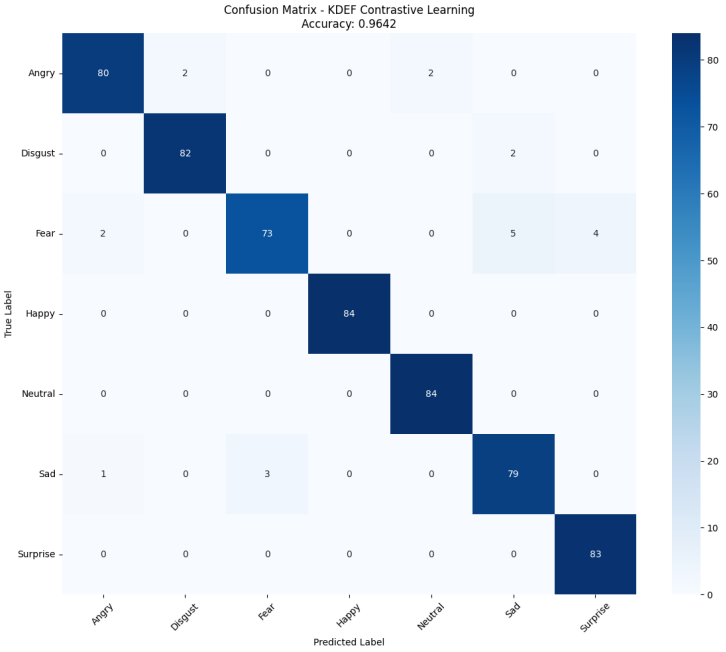


Figura 37 – Matriz de confusão do KDEF para o modelo com contraste em 150 épocas.

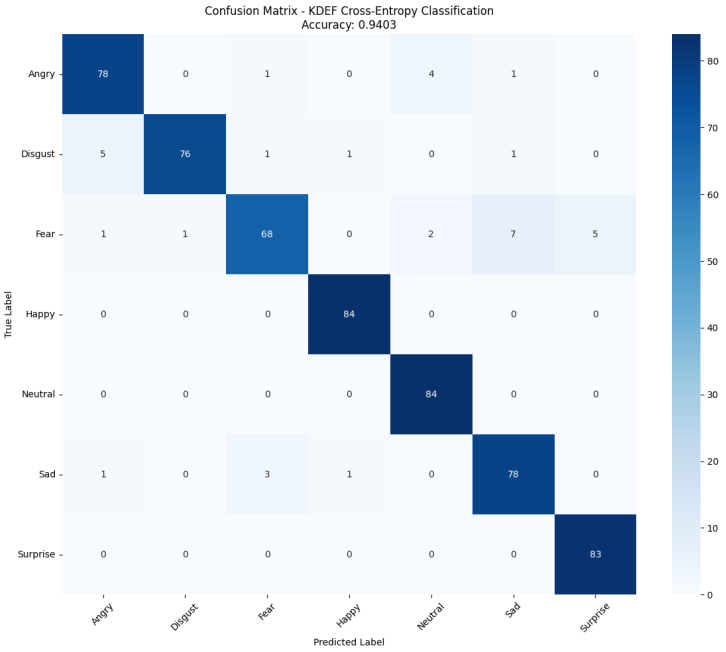


Figura 38 – Matriz de confusão do KDEF para o modelo sem contraste em 150 épocas.

C.2.3 KDEF

No dataset KDEF, o aprendizado contrastivo demonstra desempenho superior e extremamente estável, atingindo 96.42% de acurácia tanto em 50 quanto em 150 épocas. Esse resultado supera consistentemente a abordagem supervisionada, que apresenta queda de desempenho com o aumento do número de épocas.

A alta separabilidade entre classes e o reduzido número de amostras tornam o aprendizado contrastivo especialmente eficaz, atuando como um regularizador de representações e mitigando

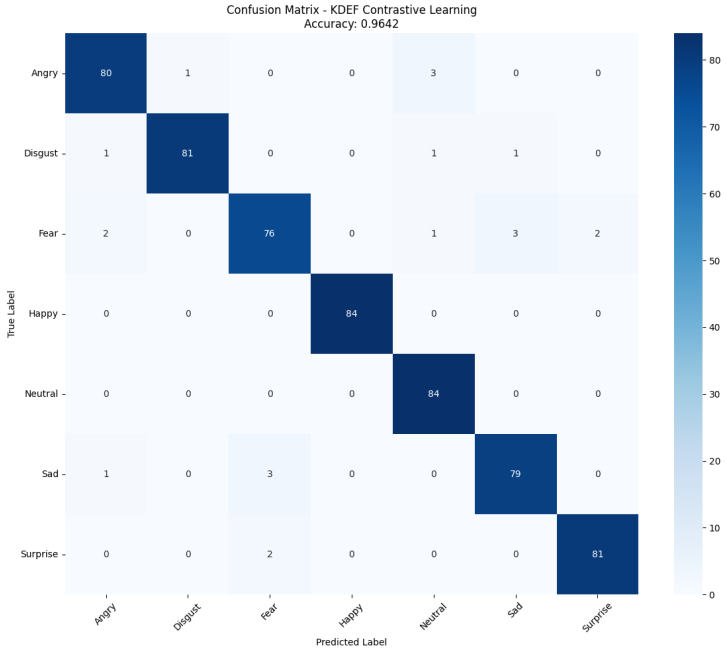


Figura 39 – Matriz de confusão do KDEF para o modelo com contraste em 50 épocas.

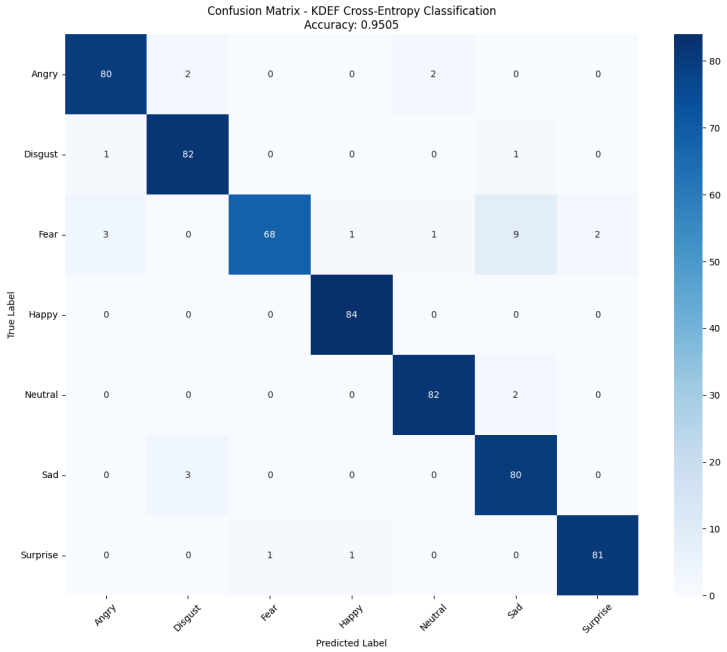


Figura 40 – Matriz de confusão do KDEF para o modelo sem contraste em 50 épocas.

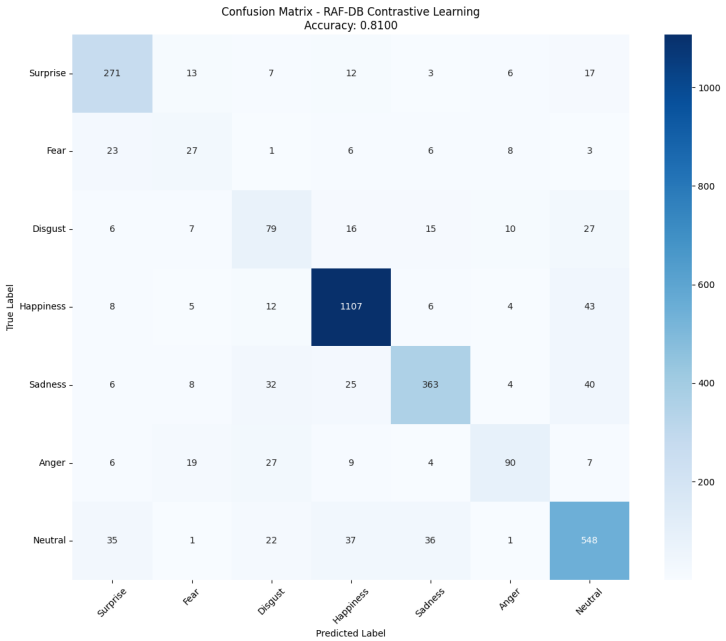


Figura 41 – Matriz de confusão do RAF-DB para o modelo com contraste em 150 épocas.

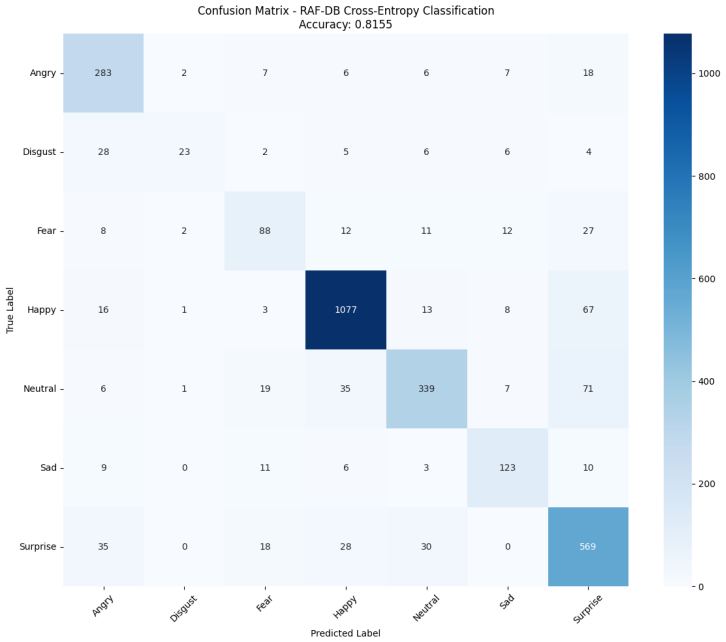


Figura 42 – Matriz de confusão do RAF-DB para o modelo sem contraste em 150 épocas.

sobreajuste.

C.2.4 RAF-DB

No RAF-DB, o treinamento supervisionado apresenta melhor desempenho em ambos os regimes, atingindo 82.92% em 50 épocas e 81.55% em 150 épocas. Entretanto, o aprendizado contrastivo mantém desempenho competitivo, com diferença inferior a 2%.

A análise por classe evidencia que emoções como *Fear* e *Disgust* permanecem desafiadoras,

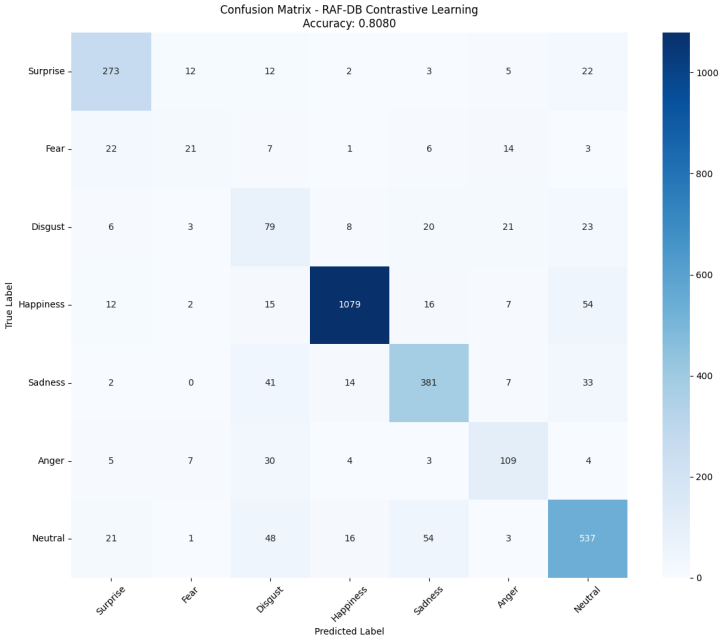


Figura 43 – Matriz de confusão do RAF-DB para o modelo com contraste em 50 épocas.

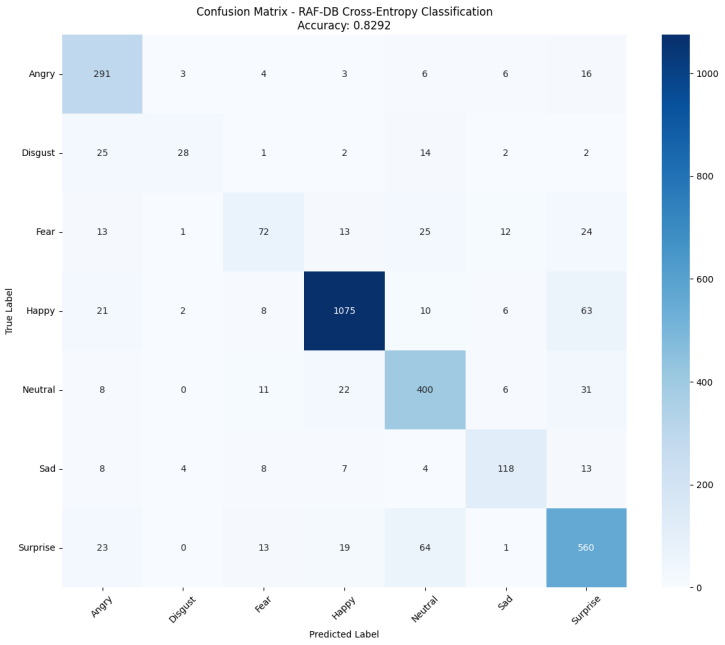


Figura 44 – Matriz de confusão do RAF-DB para o modelo sem contraste em 50 épocas.

com baixos valores de F1-score, enquanto classes dominantes como *Happiness* apresentam valores superiores a 92%.

C.2.5 Discussão Geral

Os resultados demonstram que o impacto do aprendizado contrastivo é fortemente dependente das características do dataset. Em bases pequenas e bem controladas, como o KDEF, o aprendizado contrastivo supera consistentemente a classificação supervisionada. Em datasets maiores e mais complexos, como FER13 e RAF-DB, a supervisão explícita mostra-se mais robusta, especialmente para classes minoritárias.

Além disso, o aumento do número de épocas beneficia o aprendizado contrastivo apenas quando há diversidade suficiente nos dados, como observado no CIFAR-10, enquanto pode induzir degradação de desempenho em cenários com forte desbalanceamento.

C.2.6 Conclusão Parcial

Conclui-se que o aprendizado contrastivo é particularmente eficaz em cenários de dados limitados ou com alta separabilidade semântica, enquanto abordagens supervisionadas permanecem mais adequadas para bases desbalanceadas e com ruído de rótulos. Esses achados reforçam a importância da escolha da estratégia de aprendizado de acordo com a natureza do dataset.