Uma arquitetura de Data Lake baseado em Deep Learning para busca de imagens parasitárias de doenças socialmente determinadas.

João Gabriel Marques de Lima[1], **Danilo Fernandes**[1]
Fabiane da Silva Queiroz[1], **André L. L. Aquino**[1]

[1] Laboratório Orion – Universidade Federal de Alagoas (UFAL)
Av. Lourival Melo Mota, S/N - Tabuleiro do Martins, Maceió - AL, 57072-970, Brazil

{joao.gabriel,dfc,fabiane.queiroz,alla}@orion.ufal.br

Abstract. *This work compares data pipelines using image files in PNG format and Deep Lake for medical images (SHdataset) through performance benchmarks and a Deep Metric Learning (DML) case study to analyze performance and effectiveness trade-offs. Results indicate that Deep Lake, while faster in data iteration, required 59.2% more storage; although it produced a model of comparable quality, the image file-based approach yielded a feature space with marginally superior quantitative separability. We conclude that modern formats present a trade-off between management benefits and storage/optimization costs while largely preserving model effectiveness.*

Resumo. *Este trabalho compara pipelines de dados utilizando arquivos de imagem no formato PNG e o Deep Lake para imagens médicas (SHdataset) através de benchmarks de desempenho e de um estudo de caso com Deep Metric Learning (DML) para analisar os trade-offs de eficácia e performance. O Deep Lake, embora mais rápido na iteração de dados, exigiu 59,2% mais armazenamento; apesar de produzir um modelo de qualidade comparável, a abordagem com arquivos de imagem gerou um espaço de características com separabilidade quantitativa marginalmente superior. Conclui-se que formatos modernos apresentam um trade-off entre benefícios de gerenciamento e custos de armazenamento e otimização, mas preservam em grande parte a eficácia do modelo.*

1. Introdução

Doenças socialmente determinadas são assim denominadas porque sua ocorrência, distribuição e gravidade estão profundamente vinculadas a fatores socioeconômicos, como acesso a saneamento básico, educação, renda e condições de moradia. O avanço do deep learning tem se mostrado um poderoso aliado no enfrentamento dessas enfermidades, transformando a análise de imagens médicas e permitindo avanços significativos no diagnóstico e na pesquisa [Litjens et al. 2017].

A aplicação dessas técnicas em cenários de larga escala gera um volume massivo de dados (Big Data), cuja gestão eficiente se torna um desafio central [Hu et al. 2014]. A forma como esses datasets são estruturados, armazenados e acessados não é apenas um detalhe técnico, mas um pilar fundamental, influenciando a reprodutibilidade dos resultados.

Tradicionalmente, os pipelines de dados para visão computacional são construídos sobre uma premissa simples: um diretório contendo milhares ou milhões de arquivos de imagem individuais (ex: PNG, JPEG). Esta abordagem, embora direta, impõe gargalos operacionais. A sobrecarga de I/O (entrada/saída) para ler um grande número de arquivos pequenos retarda o treinamento, enquanto o versionamento do dataset torna-se uma tarefa manual, complexa e propensa a erros [Amershi et al. 2019]. Essas limitações evidenciam a necessidade de arquiteturas de dados mais sofisticadas.

Em resposta, surgiram arquiteturas como o Data Lakehouse [Armbrust et al. 2021], com implementações especializadas para IA como o Deep Lake [Hambardzumyan et al. 2022], que prometem um ecossistema de gerenciamento de dados mais robusto. Elas propõem uma mudança de paradigma: em vez de uma coleção de arquivos, o dataset é tratado como um artefato de dados único e transacional, otimizado para streaming rápido e com versionamento integrado. Essa abordagem promete não apenas ganhos de desempenho, mas também um ecossistema de gerenciamento de dados mais robusto e auditável. Contudo, a adoção de novas tecnologias exige uma análise prática de seus trade-offs, que vão além dos benchmarks de I/O e devem incluir o impacto na otimização do modelo.

Este trabalho realiza uma análise comparativa e prática entre um pipeline de dados tradicional, baseado em imagens armazenadas em disco, e um pipeline moderno utilizando a arquitetura Deep Lake. A comparação é conduzida sob duas óticas principais, utilizando o SHdataset [Oyibo et al. 2023] de imagens microscópicas: (1) benchmarks de desempenho, focados em espaço de armazenamento e velocidade de iteração; e (2) um estudo de caso aprofundado com Deep Metric Learning (DML), utilizando a abordagem de Triplet Loss [Schroff et al. 2015], que avalia e compara as características quantitativas e qualitativas dos modelos resultantes de cada pipeline.

2. Referencial Teórico

2.1. Base de Dados Utilizada

Este estudo utiliza a base de dados pública SHdataset [Oyibo et al. 2023], que contém 12.051 imagens microscópicas de ovos de *Schistosoma Haematobium*, doença parasitária de alto impacto em regiões com recursos limitados. As amostras foram coletadas como parte de um estudo de campo em Abuja, Nigéria. O estudo envolveu a amostra de urina de crianças em idade escolar que apresentavam hematúria.

O processamento de amostras seguiu o procedimento padrão, onde 10 mL de urina foram passados por uma membrana de filtro de 13mm de diâmetro com poros de 0,2 μm. Então, a membrana foi montada em lâmina de microscopia para aquisição de imagens, onde se foi utilizado o microscópio digital "Schistoscope".

2.2. Gerenciamento de Grandes Volumes de Imagens Médicas

Segundo [Yousef Ameen Esmail Ahmed et al. 2023] e [Deheyab et al. 2022], o processamento de imagens médicas tornou-se uma área fundamental na computação contemporânea, oferecendo suporte ao diagnóstico médico por meio de ferramentas para detecção automática. No entanto, os avanços tecnológicos em dispositivos de aquisição de imagens resultaram em um crescimento na quantidade de dados gerados. Este volume

crescente de dados, juntamente com a necessidade de análises em tempo real, é uma característica do big data. As imagens médicas se encaixam nesse paradigma, contribuindo para o aumento de dados de domínios distintos.

Além do volume, o big data apresenta características como a variedade (dados estruturados, semiestruturados e não estruturados) e a velocidade de geração. Em domínios como imagens médicas, os dados frequentemente incluem formatos complexos e não tabulares, como imagens e vídeos, que não se encaixam bem em estruturas relacionais tradicionais [Armbrust et al. 2021]. Os sistemas tradicionais de gerenciamento de dados, baseados principalmente em sistemas de gerenciamento de banco de dados relacionais (RDBMS), são inadequados para lidar com o volume e a variedade inerentes ao big data, oferecendo pouco suporte para dados não estruturados ou semiestruturados e não escalando de forma adequada com hardware comum para lidar com o volume crescente [Hu et al. 2014].

Como consequência, novas metodologias e arquiteturas de dados são necessárias para lidar com Big Data. No campo da medicina, a análise avançada e o machine learning, em particular o deep learning, têm se mostrado poderosos para tarefas de visão computacional, como a detecção de pneumonia a partir de raios-X, alcançando ou superando o nível de radiologistas experientes [Rajpurkar et al. 2017].

2.3. Arquitetura Deep Lake

Deep Lake [Hambardzumyan et al. 2022] é apresentado como um lakehouse especializado para deep learning, que preserva os benefícios dos data lakes tradicionais, porém apresenta uma diferença: armazena dados complexos (imagens, vídeos, dados tabulares) na forma de tensores. Essa abordagem permite o streaming rápido de dados pela rede diretamente para frameworks de deep learning (PyTorch, TensorFlow, JAX) ou para um motor de visualização no navegador, sem sacrificar a utilização da GPU. O formato de armazenamento de tensor do Deep Lake (TSF) é otimizado para lidar com arrays de forma dinâmica e acessar dados para treinamento de deep learning. Também inclui ferramentas específicas para deep learning que não são nativas em data lakes tradicionais, como Tensor Query Language (TQL) que suporta operações multidimensionais em tensores.

2.4. Deep Metric Learning

As arquiteturas de Deep Learning, como as Redes Neurais Convolucionais (CNNs), são eficazes para aprender automaticamente representações ricas e hierárquicas a partir de dados brutos, como imagens, eliminando a necessidade da engenharia manual de características [LeCun et al. 2015] e [Krizhevsky et al. 2012]. Em paralelo, a área de Metric Learning busca aprender uma função de distância otimizada diretamente dos dados, com o objetivo de aprimorar o desempenho de classificadores baseados em distância, como o k-Nearest Neighbor [Mensink et al. 2012].

A sinergia dessas duas áreas, conhecida como Deep Metric Learning (DML), utiliza o Deep Learning para extrair características e o Metric Learning para definir uma métrica de distância eficaz sobre essas representações. Essa abordagem é crucial para lidar com a complexidade de imagens médicas em cenários de Big Data [Hambardzumyan et al. 2022].

A função de perda Triplet Loss [Schroff et al. 2015], otimiza o espaço de características operando sobre trios de amostras, compostos por uma âncora, uma amostra positiva (da mesma classe) e uma negativa (de uma classe distinta). O objetivo do treinamento é ajustar as representações para que a distância entre a âncora e o par positivo não seja apenas menor que a distância para o par negativo, mas que a exceda por uma margem de separação predefinida. Ao forçar essa condição em todo o conjunto de dados, o modelo aprende a gerar embeddings altamente discriminativos, que agrupam semanticamente as classes e maximizam a distância entre elas.

3. Metodologia

3.1. SH Dataset 12K

Foram utilizadas 12.051 imagens microscópicas primárias e suas respectivas máscaras, ambas no formato PNG, conforme disponibilizadas pelo SH Dataset. As imagens, oriundas dos subconjuntos de treinamento e teste do conjunto original, foram tratadas como uma única coleção de pares imagem-máscara para os propósitos deste estudo. O subconjunto denominado diagnosis_test_dataset, descrito em [Oyibo et al. 2023], foi desconsiderado nos experimentos apresentados. Antes da ingestão na arquitetura Deep Lake, o volume total dos dados era de aproximadamente 26,2 GB. Os arquivos de anotação no formato JSON, incluídos na descrição original do dataset, não foram utilizados nas avaliações de desempenho relacionadas ao armazenamento ou à velocidade de leitura.

3.2. Hardware Utilizado

Os experimentos foram realizados em uma estação de trabalho configurada para otimizar as operações de entrada, saída e processamento de dados. O sistema contava com um processador AMD Ryzen 5 9600X, utilizado na execução dos scripts de benchmark e nas tarefas de manipulação dos dados. Para o carregamento e tratamento dos datasets, foram empregados 32 GB de memória RAM DDR5, operando a 6000 MHz, o que proporcionou alocação e acesso eficientes aos dados em memória, mesmo considerando os efeitos de cache do sistema operacional. As operações de leitura, tanto do dataset original em formato PNG quanto da estrutura Deep Lake, foram realizadas em uma unidade SSD NVMe de 1 TB, com velocidade de leitura sequencial de aproximadamente 6000 MB/s, a fim de minimizar possíveis gargalos relacionados à entrada e saída de dados.

3.3. Configuração do Ambiente

Todos os experimentos deste estudo foram conduzidos em um ambiente de software configurado sobre o sistema operacional Windows 11, por meio do Subsistema Windows para Linux (WSL), utilizando a distribuição Ubuntu 24.04.2 LTS. Todos os scripts e processos foram executados em um ambiente virtual Python, com a versão 3.11.12 da linguagem.

As principais bibliotecas Python empregadas neste estudo incluíram: Activeloop Deep Lake (versão 4.2.7), utilizada para a criação e manipulação do dataset otimizado; NumPy (versão 2.1.3), para operações numéricas; Pillow (versão 11.2.1), para carregamento das imagens em formato PNG; o framework PyTorch (versão 2.7.1) juntamente com as bibliotecas torchvision (versão 0.22.1) e pytorch-metric-learning (versão 2.8.1) para o treinamento do modelo de deep learning; a biblioteca Scikit-learn (versão 1.7.0) para as tarefas de avaliação; e Matplotlib (versão 3.10.3), para a geração dos gráficos comparativos. O desenvolvimento e a execução dos experimentos ocorreram predominantemente no ambiente interativo Jupyter Notebook.

3.4. Implementação da Arquitetura DeepLake

Para a avaliação comparativa, as 12.051 imagens microscópicas primárias e suas respectivas máscaras de segmentação, provenientes do SH Dataset, foram adicionadas em um novo dataset Deep Lake. O esquema do dataset foi definido para conter as seguintes colunas: id (inteiro de 32 bits), image (tipo imagem com compressão PNG), mask (tipo imagem com compressão PNG), split (texto indicando a partição 'train' ou 'test') e original_filename (texto), sendo a estrutura inicial finalizada com um commit de consolidação.

A ingestão dos dados foi automatizada por meio de um script Python, que percorreu recursivamente as subpastas train e test do SH Dataset original. Para cada par imagem-máscara, o script realizou o carregamento utilizando a biblioteca Pillow, seguido da conversão para arrays NumPy, com padronização das imagens primárias para o espaço de cores RGB. Esses dados, juntamente com seus metadados associados (id, nome do arquivo original e partição correspondente).

3.5. Obtenção de Métricas e Benchmarking

A avaliação de desempenho comparou o acesso a imagens armazenadas como arquivos no formato PNG com a arquitetura Deep Lake em duas etapas: otimização de armazenamento e velocidade de iteração. Para a análise de espaço, comparou-se o tamanho total em disco das imagens e máscaras codificadas em PNG com o diretório do dataset Deep Lake. Para a velocidade, foi realizado um benchmark de iteração completa sobre as 12.051 amostras, carregando-as sequencialmente com a biblioteca Pillow no caso das imagens em PNG, ou por acesso direto aos tensores no caso do Deep Lake.

Para o benchmark de velocidade, foram realizadas cinco iterações consecutivas para cada abordagem, com o cache de arquivos do sistema operacional sendo limpo antes de cada bloco de testes. A análise dos tempos distinguiu o desempenho de "partida fria"(a primeira iteração, medindo o acesso a disco) da "partida quente"(a média e o desvio padrão das quatro iterações seguintes, refletindo o acesso a dados em cache).

3.6. Extração de Características com Deep Metric Learning e k-NN

Para avaliar e comparar qualitativamente os modelos resultantes de cada pipeline de dados (PNG local e Deep Lake), foi conduzido um estudo de caso utilizando a abordagem de Deep Metric Learning (DML). O objetivo desta etapa não era a classificação direta, mas treinar um modelo para gerar um espaço de características onde amostras da mesma classe estivessem próximas e amostras de classes diferentes, distantes. O fluxograma de treinamento está ilustrado na Figura 1.

Utilizou-se um modelo ResNet-18 pré-treinado, modificado para gerar embeddings de 64 dimensões. O treinamento foi conduzido sob o paradigma de Deep Metric Learning (DML), empregando a função de perda TripletMarginLoss (margem de 0.2) com mineração de triplets "semi-hard". O modelo foi otimizado com o algoritmo Adam por 50 épocas (taxa de aprendizado de 0.0001, batch size de 32).

A avaliação da qualidade dos embeddings foi realizada de forma indireta: um classificador k-Nearest Neighbors (k=5) foi treinado nos embeddings de treino e sua performance (Acurácia, F1, AUC) foi medida no conjunto de teste. Adicionalmente, o algoritmo t-SNE (perplexidade de 30) foi usado para visualizar a separação das classes em um espaço 2D. O fluxograma das métricas de avaliação está ilustrado na Figura 2.

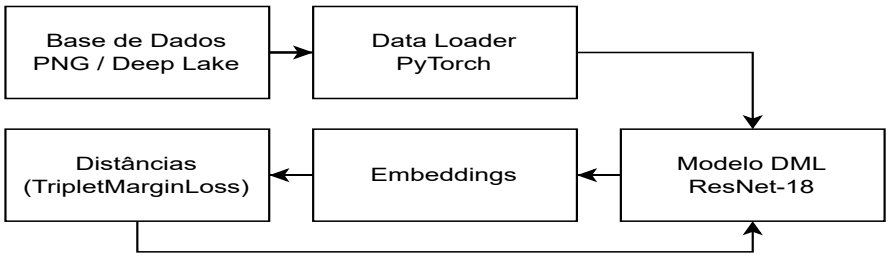

Figura 1. Arquitetura do ciclo de treinamento para o modelo de Deep Metric Learning (DML). Triplet Loss, calculada a partir dos embeddings de cada lote, é usada para otimizar os pesos do modelo via backpropagation.

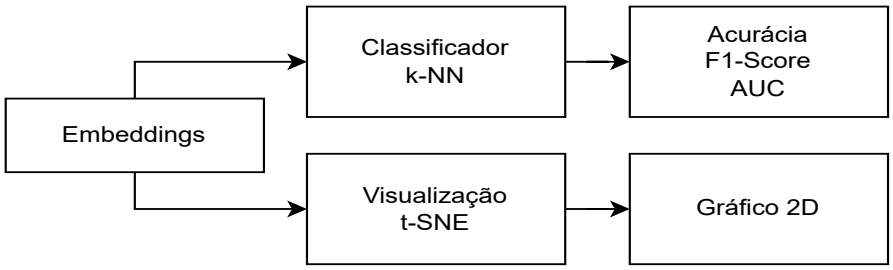

Figura 2. Fluxograma do processo de avaliação do modelo.

4. Resultados e Discussões

4.1. Armazenamento

A primeira métrica de desempenho analisada foi a otimização do espaço de armazenamento. A Figura 3 apresenta a comparação do tamanho total ocupado em disco pelos dados brutos e pelo dataset consolidado na arquitetura Deep Lake. O conjunto de dados original, composto por 24.102 arquivos no formato PNG, totalizou 26.870,88 MB. Em contrapartida, o dataset Deep Lake correspondente ocupou 42.789,47 MB, representando um aumento de aproximadamente 59,2% no espaço de armazenamento.

Este aumento é atribuído primariamente à sobrecarga estrutural e de metadados inerente ao formato Deep Lake. Diferente de um simples diretório de arquivos, a arquitetura gerencia os dados em “chunks” e armazena um extenso conjunto de informações adicionais para viabilizar suas funcionalidades, como o versionamento de dados, esquemas de tensores e índices para acesso otimizado. Como as imagens de origem já estavam em um formato comprimido (PNG), o potencial de otimização por uma nova compressão foi limitado, fazendo com que a sobrecarga da estrutura de gerenciamento se tornasse o fator dominante no tamanho final. Evidencia-se, portanto, um trade-off fundamental, no qual as capacidades avançadas de gerenciamento de dados do Deep Lake são obtidas ao custo de um maior consumo de espaço em disco.

4.2. Velocidade de Consulta

A segunda métrica de desempenho avaliada foi a velocidade de iteração sobre todo o conjunto de dados. A Figura 4 compara os tempos de execução, em milissegundos, para uma leitura completa dos 12.051 pares imagem-máscara, distinguindo entre ”partida

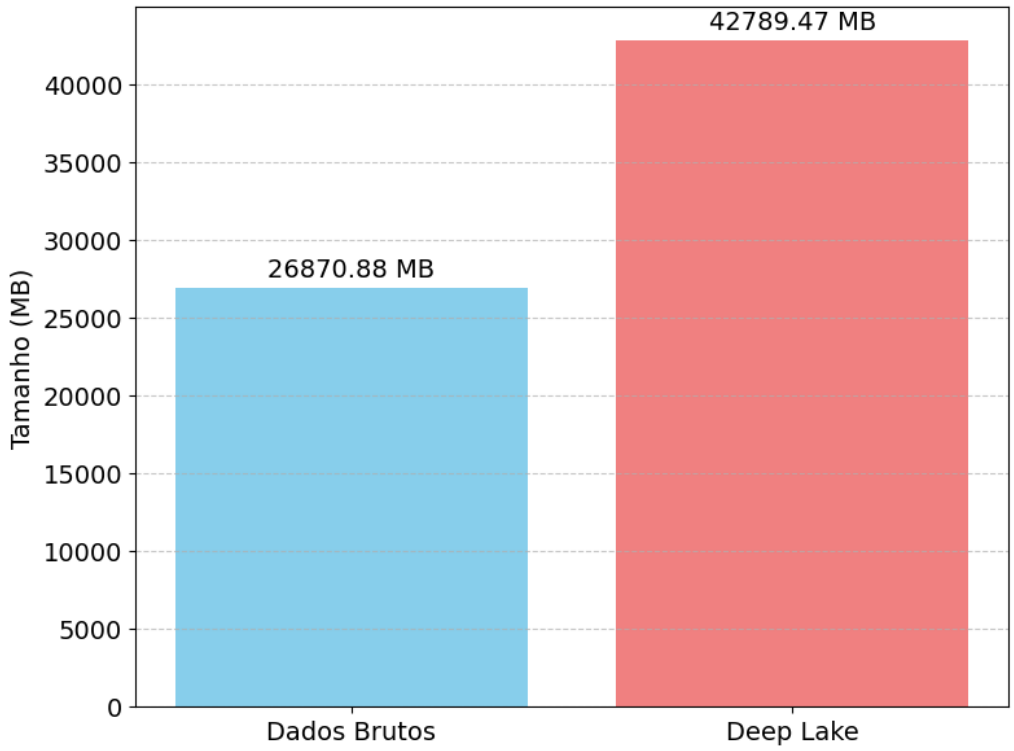

Figura 3. Comparação do espaço de armazenamento em disco.

fria"(primeira leitura, sem cache) e uma "partida quente"(média de leituras subsequentes, com cache).

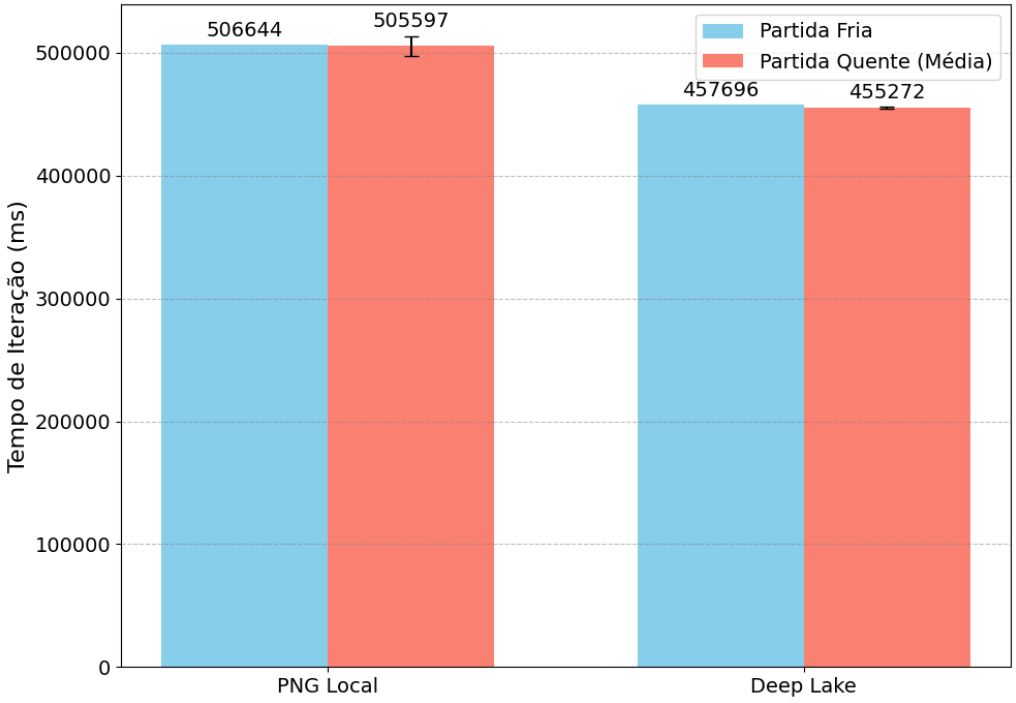

Figura 4. Velocidade de Consulta. Imagens locais no formato PNG (esquerda) e Deep Lake (direita).

Nos testes de partida fria, o Deep Lake demonstrou ser mais rápido, completando a tarefa em 457.696 ms, enquanto o acesso às imagens armazenadas localmente no formato PNG levou 506.644 ms. A vantagem de desempenho do Deep Lake foi mais evidente nas partidas quentes, com um tempo médio de 455.272 ms contra 505.597 ms do pipeline baseado em imagens locais. No entanto, a diferença na estabilidade do desempenho, visualizada pelas barras de desvio padrão, demonstra que o pipeline com imagens locais exibiu uma alta variabilidade, com um desvio padrão de 8.170 ms, enquanto o Deep Lake se mostrou extremamente consistente, com um desvio padrão de apenas 1.027 ms.

A superioridade no desempenho pode ser atribuída à forma como os dados são armazenados e acessados. Em vez de realizar milhares de operações de I/O (entrada/saída) para abrir e ler 24.102 imagens codificadas em PNG individualmente, o Deep Lake acessa os dados em chunks otimizados e contínuos dentro de uma estrutura de arquivos consolidada. Isso minimiza a latência e resulta em um carregamento rápido e previsível, explicando tanto a menor média de tempo quanto o desvio padrão inferior.

4.3. Estudo de Caso: Deep Lake e Deep Metric Learning

A avaliação quantitativa iniciou-se com uma análise direta da estrutura do espaço de características aprendido pelo Deep Metric Learning, conforme detalhado na Tabela 1. Foram calculadas as estatísticas das distâncias Euclidianas entre as amostras de teste para validar se o modelo cumpriu seu objetivo: agrupar amostras da mesma classe e separar as de classes distintas. O modelo treinado com imagens armazenadas localmente apresentou um espaço com maior separabilidade entre as classes.

Pipeline	Tipo de Distância	Média	Desvio Padrão
Deep Lake	Intra-classe	13.3013	11.5755
	Inter-classe	19.8972	16.0819
PNG Local	Intra-classe	14.9275	17.6376
	Inter-classe	37.4492	34.6348

Tabela 1. Média e desvio padrão das distâncias intra-classes e inter-classe.

O modelo treinado com o pipeline utilizando Deep Lake foi capaz de aprender um espaço de características eficaz para a tarefa. A qualidade dos embeddings foi validada indiretamente pela performance de um classificador k-NN, que alcançou uma Acurácia de 0.8335, um F1-Score de 0.5460 e um AUC de 0.7716. Esses resultados são similares aos obtidos com o pipeline de referência baseado em imagens no formato PNG armazenadas localmente, que atingiu uma acurácia de 0.8475, F1-Score de 0.5638 e AUC de 0.7725.

A análise visual do espaço de embeddings na Figura 5 reforça essa observação. O gráfico t-SNE para o modelo treinado com Deep Lake (direita) exibe uma tendência de separação entre as classes ”Positiva”e ”Negativa”, com uma estrutura qualitativa quase idêntica à do modelo de referência (esquerda).

A análise conjunta dos resultados demonstra que, embora a arquitetura Deep Lake seja viável, o pipeline tradicional baseado em imagens armazenadas localmente no formato PNG se mostrou marginalmente superior para esta carga de trabalho específica. A maior separabilidade das classes observada na análise de distâncias (Tabela 1) oferece uma explicação quantitativa para a pequena vantagem do pipeline com imagens locais

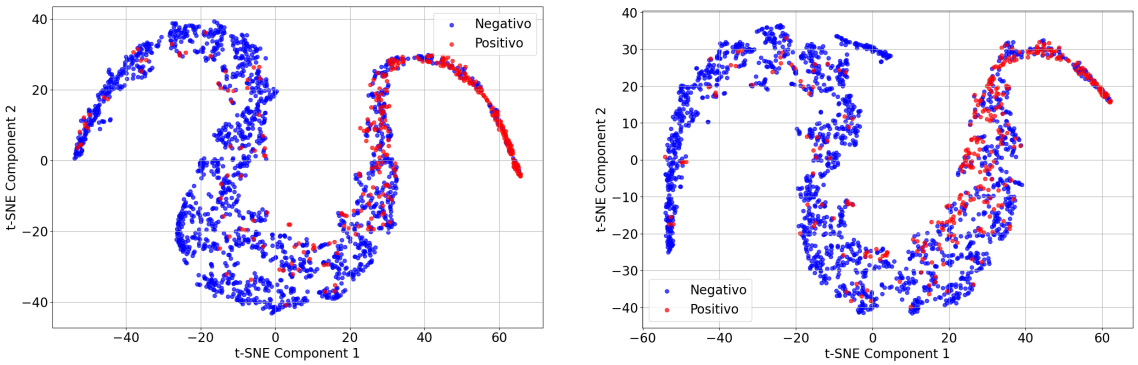

Figura 5. PNGs locais (esquerda) e Deep Lake (direita).

nas métricas do k-NN. Isso sugere que a maneira como os dados são processados e entregues ao modelo por cada pipeline pode ter consequências sutis no resultado final do treinamento.

5. Conclusão

Os resultados dos benchmarks de desempenho demonstraram que, enquanto a estrutura Deep Lake apresentou uma vantagem significativa em velocidade e estabilidade para tarefas de iteração pura sobre os dados, ela o fez ao custo de um aumento de quase 59,2% no espaço de armazenamento em disco, devido à sua sobrecarga estrutural e de metadados. Por outro lado, o estudo de caso com Deep Metric Learning revelou um resultado mais complexo: embora a qualidade final dos modelos seja qualitativamente comparável, uma análise mais aprofundada da estrutura do espaço de características demonstrou uma superioridade marginal, porém mensurável, do pipeline baseado em imagens armazenadas localmente no formato PNG. Este alcançou uma separabilidade entre classes quantitativamente maior, o que explica seu desempenho ligeiramente superior nas métricas de classificação.

A principal implicação deste trabalho é que a decisão sobre a arquitetura de dados deve ser guiada pelas prioridades específicas de cada projeto. Se a velocidade de leitura pura, o versionamento e a facilidade de gerenciamento de dados em larga escala são cruciais, o Deep Lake se apresenta como uma solução robusta, cujo custo em armazenamento pode ser justificável. Para projetos de escopo menor, a simplicidade e a eficiência de armazenamento das imagens codificadas em PNG permanecem uma alternativa não apenas viável, mas competitiva, como demonstrado pelos resultados de qualidade do modelo.

A adoção de formatos de dados modernos como o Deep Lake pode ser feita com a confiança de que as funcionalidades avançadas que são oferecidas preservam em grande parte a capacidade do modelo de aprender as características semânticas da tarefa, embora este estudo sugira que diferenças sutis na otimização final do espaço de características possam surgir. Como trabalhos futuros, sugere-se a replicação destes experimentos em ambientes de nuvem e com datasets de ordens de magnitude ainda maiores, onde os benefícios de uma arquitetura Deep Lake podem se tornar ainda mais pronunciados.

Referências

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., and Zimmermann, T. (2019). Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300. IEEE.

Armbrust, M., Ghodsi, A., Xin, R., and Zaharia, M. (2021). Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In *Proceedings of CIDR*, volume 8, page 28.

Deheyab, A. O. A., Alwan, M. H., khalid Abdul Rezzaqe, I., Mahmood, O. A., Hammadi, Y. I., Kareem, A. N., and Ibrahim, M. (2022). An overview of challenges in medical image processing. In *The 6th International Conference on Future Networks & Distributed Systems (ICFND '22)*, page 6, Tashkent, TAS, Uzbekistan. ACM.

Hambardzumyan, S., Tuli, A., Ghukasyan, L., Rahman, F., Topchyan, H., Isayan, D., McQuade, M., Harutyunyan, M., Hakobyan, T., Stranic, I., et al. (2022). Deep lake: A lakehouse for deep learning.

Hu, H., Wen, Y., Chua, T.-S., and Li, X. (2014). Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. *IEEE Access*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88.

Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. (2012). Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *European Conference on Computer Vision*, pages 488–501. Springer.

Oyibo, P., Meulah, B., Agbana, T., Bengtson, M., van Lieshout, L., Oyibo, W., Vdovine, G., and Diehl, J.-C. (2023). Schistosoma Haematobium Egg Image Dataset.

Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., et al. (2017). Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Yousef Ameen Esmail Ahmed et al., Biao Yue, Z. G. J. Y. (2023). An overview: Big data analysis by deep learning and image processing. *World Scientific Journals*, 21(07).