



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS DA EDUCAÇÃO
DEPARTAMENTO DE CIÊNCIA DA INFORMAÇÃO
CURSO DE GRADUAÇÃO EM CIÊNCIA DA INFORMAÇÃO

Matheus Faustino

Pipeline para acesso efêmero a dados sensíveis na Tele dermatologia

Florianópolis
2026

Matheus Faustino

Pipeline para acesso efêmero a dados sensíveis na Teledermatologia

Trabalho de Conclusão de Curso do Curso de Graduação em Ciência da Informação do Centro de Ciências da Educação da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Ciência da Informação.

Orientador: Prof. Dr. Douglas Dyllon Jeronimo de Macedo

Florianópolis

2026

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Faustino, Matheus

Pipeline para acesso efêmero a dados sensíveis na
Teledermatologia / Matheus Faustino ; orientador, Douglas
Dyllon Jeronimo de Macedo, 2026.
76 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro de Ciências
da Educação, Graduação em Ciência da Informação,
Florianópolis, 2026.

Inclui referências.

1. Ciência da Informação. 2. Teledermatologia. 3.
Pipeline de Dados. 4. Acesso Efêmero. 5. Inteligência
Artificial. I. Macedo, Douglas Dyllon Jeronimo de . II.
Universidade Federal de Santa Catarina. Graduação em
Ciência da Informação. III. Título.

Matheus Faustino

Pipeline para acesso efêmero a dados sensíveis na Teledermatologia

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Ciência da Informação e aprovado em sua forma final pelo Curso de Graduação em Ciência da Informação da Universidade Federal de Santa Catarina.

Florianópolis, 26 de junho de 2026.



Documento assinado digitalmente
Ricardo Alexandre Reinaldo de Moraes
Data: 06/07/2026 09:58:58-0300
CPF: ***.812.939-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Ricardo Alexandre Reinaldo de Moraes
Coordenador do Curso

Banca Examinadora:



Documento assinado digitalmente
Douglas Dyllon Jeronimo de Macedo
Data: 06/07/2026 09:34:25-0300
CPF: ***.987.372-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Dr. Douglas Dyllon Jeronimo de
Macedo

Orientador



Documento assinado digitalmente
Alex Sandro Roschildt Pinto
Data: 07/07/2026 10:17:10-0300
CPF: ***.787.440-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Alex Sandro Roschildt Pinto.
Avaliador
INE/UFSC



Documento assinado digitalmente
Gustavo Medeiros de Araujo
Data: 06/07/2026 09:49:28-0300
CPF: ***.754.991-**
Verifique as assinaturas em <https://v.ufsc.br>

Gustavo Medeiros de Araujo.
Avaliador
CIN/UFSC

AGRADECIMENTOS

Gostaria de agradecer de coração a todos que estiveram do meu lado e contribuíram, de alguma forma, para que este trabalho e a minha graduação se tornassem realidade.

À minha companheira, Micaela, por todo o amor, paciência e apoio incondicional durante esta longa trajetória acadêmica. Agradeço por abdicar de seus finais de semana para me fazer companhia enquanto estudava ou desenvolvia trabalhos, e por ser o meu porto seguro nos momentos de maior ansiedade. Obrigado por me dar a força necessária para nunca desistir e, acima de tudo, por estar sempre presente em todos os instantes, felizes ou tristes, desta jornada, oferecendo não apenas apoio, mas o incentivo essencial para que eu continuasse buscando cada vez mais conhecimento.

Aos meus pais e ao meu irmão, por estarem sempre ao meu lado, por acreditarem no meu potencial e no meu futuro.

Aos amigos que fiz ao longo deste caminho, com os quais dividi preocupações, desafios, mas também grandes alegrias. Vocês tornaram essa jornada muito mais leve, e são pessoas que levarei para a vida toda.

Aos professores do curso de Ciência da Informação da Universidade Federal de Santa Catarina (UFSC), por todos os ensinamentos e reflexões que moldaram a minha visão profissional e acadêmica. Em especial, registro um agradecimento gigante ao meu orientador, Prof. Dr. Douglas Dyllon Jeronimo de Macedo, não apenas pela paciência e pelas excelentes conversas, mas, sobretudo, por ter acreditado na relevância deste tema e me guiado na viabilidade desta pesquisa desde o primeiro momento.

Por fim, dedico meus sinceros agradecimentos à equipe do Sistema Integrado de Telemedicina e Telessaúde (STT), que abriu as portas e forneceu a infraestrutura necessária para que este projeto deixasse de ser apenas uma ideia. Um agradecimento mais que especial ao time de desenvolvedores e, ao Harley, pela parceria constante, pelas ajudas técnicas indispensáveis e por todo o apoio fundamental na construção da arquitetura e na escrita deste trabalho.

RESUMO

O avanço exponencial do Sistema de Telemedicina e Telessaúde de Santa Catarina (STT) resultou na consolidação de repositórios massivos de imagens médicas, formando um ecossistema de *Big Data* que ultrapassa a marca de 10 milhões de exames. Esse vasto acervo constitui um ativo informacional fundamental para o desenvolvimento de modelos preditivos de Inteligência Artificial (IA) voltados ao diagnóstico precoce do câncer de pele, como o melanoma. Contudo, a utilização desses ativos por pesquisadores externos suscita um grave conflito entre a inovação tecnológica e o direito fundamental à privacidade. A problemática central reside na vulnerabilidade da quebra de custódia: ao permitir a persistência de arquivos físicos nos discos rígidos e dispositivos de armazenamento secundário (SSD) locais dos pesquisadores, o dado sensível fica exposto a negligências operacionais e falhas humanas, configurando riscos severos de vazamento e descumprimento da Lei Geral de Proteção de Dados (LGPD). Na literatura, abordagens tradicionais baseadas em criptografia pesada inviabilizam o processamento, enquanto métodos de anonimização alteram os *pixels* e prejudicam a precisão diagnóstica. Diante desse cenário, este trabalho tem como objetivo geral desenvolver e validar uma arquitetura de *pipeline* de acesso efêmero a imagens médicas, visando garantir a proteção absoluta dos dados por meio do processamento estritamente em memória volátil (RAM). A solução implementa o paradigma de Persistência Zero, integrando o servidor de objetos MinIO e a geração de URLs pré-assinadas temporárias para injetar o fluxo binário diretamente em um *buffer* de memória via classe `io.BytesIO` da linguagem Python. Do ponto de vista metodológico, trata-se de uma pesquisa aplicada, experimental e quantitativa. O ambiente de testes simulou o STT utilizando a infraestrutura de alto desempenho Jupyter VLAB da UFSC, equipada com NVIDIA DGX H100. A base pública HAM10000 foi submetida a técnicas de *oversampling* e particionamento para evitar vazamento de dados algorítmico. O sistema foi validado em quatro eixos: preparação algorítmica, auditoria de segurança, *benchmark* de tráfego de rede e eficácia diagnóstica. Os resultados práticos atestaram o sucesso da solução. A auditoria de disco, realizada por meio de mapeamento de diretórios (*snapshotting*) e monitoramento de RAM (`tracemalloc`), comprovou matematicamente a gravação nula de arquivos residuais. Para mitigar o gargalo de rede imposto pelo acesso remoto, construiu-se um pré-carregamento assíncrono (*prefetching*) utilizando múltiplos núcleos da CPU, o que garantiu uma vazão superior a 11 imagens por segundo, eliminando a ociosidade da placa de vídeo. Ao submeter o *pipeline* a um ciclo completo de treinamento preditivo utilizando uma rede neural, o modelo obteve acurácia próxima a 90% no conjunto de teste, preservando a integridade integral dos *pixels* originais. Conclui-se que o aumento no tempo de treinamento constitui um custo de privacidade vantajoso, materializando o conceito de *Privacy by Design* ao provar que é possível assegurar rigor extremo em segurança da informação sem comprometer a eficácia clínica e a evolução da IA médica.

Palavras-chave: Teledermatologia. Persistência Zero. Inteligência Artificial. LGPD. Pipeline de Dados. Acesso Efêmero.

ABSTRACT

The exponential advancement of Telemedicine and Telehealth System of Santa Catarina (STT) has resulted in the consolidation of massive repositories of medical images, forming a *Big Data* ecosystem that surpasses the mark of 10 million exams. This vast collection constitutes a fundamental informational asset for the development of predictive Artificial Intelligence (AI) models aimed at the early diagnosis of skin cancer, such as melanoma. However, the use of these assets by external researchers raises a severe conflict between technological innovation and the fundamental right to privacy. The central problem lies in the vulnerability of chain of custody breaches: by allowing the persistence of physical files on researchers' local hard drives and secondary storage devices (SSDs), sensitive data is exposed to operational negligence and human failures, posing severe risks of leakage and non-compliance with the Brazilian General Data Protection Law (LGPD). In the literature, traditional approaches based on heavy encryption make processing unfeasible, while anonymization methods alter the *pixels* and impair diagnostic accuracy. Given this scenario, this work aims to develop and validate an ephemeral access *pipeline* architecture for medical images, aiming to guarantee absolute data protection through processing strictly in volatile memory (RAM). The solution implements the Zero Persistence paradigm, integrating the MinIO object server and the generation of temporary presigned URLs to inject the binary stream directly into a memory *buffer* via the Python language's `io.BytesIO` class. From a methodological standpoint, this is an applied, experimental, and quantitative research. The test environment simulated the STT using UFSC's high-performance Jupyter VLAB infrastructure, equipped with an NVIDIA DGX H100. The public HAM10000 database was subjected to *oversampling* and partitioning techniques to prevent algorithmic data leakage. The system was validated across four axes: algorithmic preparation, security auditing, network traffic *benchmark*, and diagnostic efficacy. The practical results attested to the success of the solution. The disk audit, performed through directory mapping (*snapshotting*) and RAM monitoring (`tracemalloc`), mathematically proved the null recording of residual files. To mitigate the network bottleneck imposed by remote access, an asynchronous pre-loading (*prefetching*) was built using multiple CPU cores, which guaranteed a throughput of over 11 images per second, eliminating GPU starvation. By subjecting the *pipeline* to a complete cycle of predictive training using a neural network, the model achieved an accuracy close to 90% on the test set, preserving the full integrity of the original *pixels*. It is concluded that the increase in training time constitutes an advantageous privacy cost, materializing the *Privacy by Design* concept by proving that it is possible to ensure extreme rigor in information security without compromising clinical efficacy and the evolution of medical AI.

Keywords: Tele dermatology. Zero Persistence. Artificial Intelligence. LGPD. Data Pipeline. Ephemeral Access.

LISTA DE FIGURAS

Figura 1 – Fluxo operacional da Tele dermatologia no STT	19
Figura 2 – Protocolo de registro fotográfico: (a) panorâmica, (b) aproximação com régua e (c) contato.	20
Figura 3 – Fluxo de dados	42
Figura 4 – Comparativo: abordagem tradicional versus arquitetura efêmera	46
Figura 5 – Fluxograma de execução da rotina isolada de verificação de persistência	49
Figura 6 – Diagrama de sequência demonstrando a mitigação do gargalo da GPU via multiprocessamento	52
Figura 7 – Fluxo da Fase 4 - Treinamento da IA e Limpeza Volátil	54
Figura 8 – Relatório de auditoria no terminal e prova visual de renderização via RAM	58
Figura 9 – Benchmark: Tempo de resposta individual por imagem carregada na RAM	59
Figura 10 – Benchmark: Distribuição estatística dos tempos de carregamento	60
Figura 11 – Teste de desempenho: Tempo total de execução sob diferentes níveis de paralelismo	62
Figura 12 – Teste de desempenho: Taxa de transferência de imagens por configuração de <i>workers</i>	63

LISTA DE QUADROS

Quadro 1 – Comparação de performance entre meios de armazenamento.	25
Quadro 2 – Localização deste Trabalho no Pipeline de IA.	32
Quadro 3 – Comparação entre Trabalhos Relacionados e a Proposta Atual.	36
Quadro 4 – Script de filtragem e limpeza estrutural do JSON.	43
Quadro 5 – Estrutura do JSON bruto original (Antes da filtragem).	44
Quadro 6 – Estrutura do JSON processado (Após a filtragem estrutural).	44
Quadro 7 – Script de classificação, <i>oversampling</i> e geração de <i>splits</i>	45
Quadro 8 – Mecanismo de injeção em memória e <i>callbacks</i> de monitoramento . . .	47
Quadro 9 – Algoritmo para rastreamento de disco e validação de não-escrita . . .	50
Quadro 10 – Orquestração de lote com multiprocessamento dinâmico no <i>DataLoader</i>	51
Quadro 11 – Algoritmo de <i>benchmark</i> para avaliação do <i>throughput</i> de rede por workers	53

LISTA DE TABELAS

Tabela 1 – Distribuição final das imagens médicas após o balanceamento	56
Tabela 2 – Recorte amostral da telemetria do ensaio efêmero (Extraído do lote de 100 instâncias)	61
Tabela 3 – Métricas absolutas do <i>benchmark</i> de paralelismo assíncrono.	63
Tabela 4 – Comparativo do tempo de execução do treinamento (5 épocas)	65
Tabela 5 – Métricas finais de inferência no conjunto de teste (9.247 imagens).	65

SUMÁRIO

1	INTRODUÇÃO	12
1.1	OBJETIVOS	13
1.2	OBJETIVO GERAL	14
1.2.1	Objetivos Específicos	14
1.3	JUSTIFICATIVA	14
1.3.1	Contribuições do Estudo	15
1.3.2	Estrutura do Documento	16
2	FUNDAMENTAÇÃO TEÓRICA	18
2.1	O ECOSSISTEMA DO SISTEMA DE TELEMEDICINA E TELESSAÚDE (STT)	18
2.1.1	O Módulo de Telediagnóstico em Dermatologia	19
2.2	DIRETRIZES DA LGPD PARA O TRATAMENTO DE DADOS SENSÍVEIS	21
2.3	O RISCO DE QUEBRA DE CUSTÓDIA NO COMPARTILHAMENTO DE DADOS	22
2.4	ARMAZENAMENTO DE OBJETOS COM MINIO	23
2.4.1	Gestão de Acesso via URLs Pré-assinadas	23
2.5	PERSISTÊNCIA VERSUS EFEMERIDADE: MEMÓRIA RAM E ARMAZENAMENTO SECUNDÁRIO	24
2.5.1	Definição, Volatilidade e Performance da Memória RAM	24
2.5.2	Armazenamento Efêmero e Persistência Zero	25
2.6	REQUISITOS DE CONSUMO DE DADOS EM APRENDIZADO PROFUNDO	26
2.7	BIBLIOTECAS PYTHON PARA PROCESSAMENTO DE DADOS EM MEMÓRIA	27
2.7.1	Manipulação de Fluxos e a Segurança em Memória Volátil	27
2.7.2	Teoria do Carregamento Assíncrono de Lotes	27
2.8	PIPELINE DE ACESSO EFÊMERO	28
2.8.1	Por Que Este Trabalho Implementa um Pipeline de Acesso Efêmero	29
2.8.1.1	Primeira Convergência: <i>Pipeline Architecture</i> como Decomposição de Processos	30
2.8.1.2	Segunda Convergência: <i>Data Pipeline Architecture</i> com Persistência Zero	30
2.8.1.3	Terceira Convergência: Pipeline como Componente de Data Handling em IA	31
2.8.2	A Inovação: Persistência Zero como Princípio de Design	32
3	TRABALHOS RELACIONADOS	34
4	METODOLOGIA	37
4.1	CARACTERIZAÇÃO DA PESQUISA	37
4.2	PROCEDIMENTOS METODOLÓGICOS	38
4.2.1	Ambiente de Experimentação e Base de Dados	38

4.2.2	Validação de Segurança e Persistência Zero	38
4.2.3	Validação de Desempenho e Tráfego de Rede	38
4.2.4	Validação da Eficácia Médica e do Custo Temporal	39
5	DESENVOLVIMENTO	40
5.1	TECNOLOGIAS E INFRAESTRUTURA COMPUTACIONAL	40
5.1.1	Bibliotecas e Componentes de Software	41
5.2	FASE 1: PREPARAÇÃO DOS DADOS	42
5.2.1	Primeira Etapa: Filtragem e Isolamento das Lesões	43
5.2.2	Segunda Etapa: Balanceamento e Divisão do Dataset	44
5.3	FASE 2: ACESSO EFÊMERO	45
5.3.1	Validação Isolada da Persistência Zero	48
5.4	FASE 3: ORQUESTRAÇÃO ASSÍNCRONA E <i>PREFETCHING</i>	50
5.5	FASE 4: TREINAMENTO DA INTELIGÊNCIA ARTIFICIAL	53
5.5.1	Processamento em Precisão Mista e Consumo de VRAM	54
5.5.2	A Atuação do Garbage Collector e a Persistência Zero	55
6	RESULTADOS E DISCUSSÕES	56
6.1	RESULTADOS DA PREPARAÇÃO E BALANCEAMENTO DE DADOS	56
6.2	COMPROVAÇÃO DA TÉCNICA DE ACESSO EFÊMERO (FASE 2)	57
6.2.1	Validação de Conectividade e Resiliência HTTP	57
6.2.2	Auditoria de Alocação e Delta de Memória RAM	57
6.2.3	Benchmark de Desempenho e Relatório de Latência	59
6.3	BENCHMARK DE REDE E MITIGAÇÃO DE GARGALO DE I/O (FASE 3)	61
6.4	EFICÁCIA DO TREINAMENTO E LIMPEZA DE MEMÓRIA (FASE 4)	64
7	CONCLUSÕES	67
7.1	TRABALHOS FUTUROS	67
	REFERÊNCIAS	69
	APÊNDICE A – DECLARAÇÃO DE USO DE INTELIGÊNCIA ARTIFICIAL	75
A.1	IA COMO FERRAMENTA DE APOIO TÉCNICO E TEXTUAL	75
A.2	RESPONSABILIDADE E AUTORIA	75

1 INTRODUÇÃO

A história da civilização é marcada pela simbiose entre a ciência, a tecnologia e a informação, uma aliança que tem permitido à humanidade desbravar fronteiras antes consideradas intransponíveis. Nessa jornada de conquistas, a medicina deixou de ser uma prática puramente reativa para se tornar preditiva, buscando na inovação as ferramentas para enfrentar patologias que outrora eram sentenças definitivas. No centro desse esforço global, destaca-se a luta contra o câncer e, especificamente neste trabalho, o câncer de pele, um adversário complexo que a ciência tenta não apenas curar, mas antecipar. É nesse cenário de evolução constante que a Ciência da Informação se funde à saúde, transformando dados brutos em esperança por meio da detecção precoce.

Historicamente, a telemedicina emergiu no cenário brasileiro não apenas como um avanço técnico, mas como uma estratégia vital para mitigar a desigualdade na distribuição de recursos hospitalares e especialistas, frequentemente concentrados em grandes centros urbanos (Wallauer et al., 2008). Ao promover a descentralização do cuidado, sistemas de larga escala permitem que o diagnóstico especializado alcance os locais mais remotos, integrando resultados ao histórico do paciente de forma ágil. Além de reduzir custos operacionais críticos, como o transporte de pacientes e a redundância de exames, essas redes foram concebidas para garantir a autenticidade, a integridade e o controle de origem das informações médicas, assegurando que o fluxo informacional seja confiável desde a captura até o laudo final (Wallauer et al., 2008). O câncer de pele, nesse contexto, segundo o Instituto Nacional de Câncer INCA (2022), representa um desafio crítico para a saúde pública global devido à sua alta incidência, sendo o diagnóstico precoce o fator determinante para a redução da mortalidade no contexto brasileiro.

Para enfrentar essa demanda, o estado de Santa Catarina consolidou o Sistema de Telemedicina e Telessaúde (STT), infraestrutura que evoluiu da antiga Rede Catarinense de Telemedicina (RCTM) (Wallauer et al., 2008). Esta arquitetura foi desenhada sob a filosofia da descentralização, permitindo que médicos especializados acessem exames realizados a quilômetros de distância dos centros de referência. Contudo, a eficácia desse modelo gerou um fenômeno de *Big Data* informacional, visto que registros de 2015 já apontavam para um crescimento exponencial, com o sistema ultrapassando a marca de 10 milhões de imagens médicas armazenadas (Macedo; Von Wangenheim; Dantas, 2015). Uma vez que a eficácia do armazenamento de grandes volumes de dados médicos foi consolidada por Macedo, Von Wangenheim e Dantas (2015), ao demonstrar que arquiteturas distribuídas superaram o desempenho de bancos de dados relacionais em até 907,49%, a viabilidade da saúde digital exige hoje um avanço na forma como esses dados são consumidos em segurança. Esse volume massivo de objetos digitais constitui um ativo valioso e um substrato robusto para o desenvolvimento de sistemas de suporte à decisão baseados em Inteligência Artificial (IA). No entanto, a existência desse vasto repositório pronto para ser estudado impõe um

novo desafio de gestão: como disponibilizar essa riqueza de conteúdo para o treinamento de modelos inteligentes sem comprometer a custódia e a privacidade da informação?

O tratamento desses dados para fins de pesquisa encontra-se em um ponto de tensão entre o avanço tecnológico e o direito fundamental à privacidade. O STT garante elevados padrões de segurança e sigilo dentro do seu perímetro institucional controlado (Wallauer et al., 2008). A vulnerabilidade crítica manifesta-se no momento em que esses ativos digitais transcendem a infraestrutura original para serem armazenados em dispositivos locais de pesquisadores. Ao permitir a persistência de dados sensíveis em meios físicos externos, a gestão da informação sujeita o dado a negligências operacionais, uma vez que a literatura destaca o fator humano como o elo mais fraco da segurança da informação. Como argumentam Freire et al. (2017), não é possível criar barreiras técnicas que protejam o componente humano com a mesma eficácia dos componentes físicos, tornando as ações dos usuários uma brecha explorada por ameaças diversas.

A problemática central deste trabalho reside nessa fragilidade de segurança física e lógica, especificamente na dificuldade de permitir que modelos de *Deep Learning* aprendam com esse vasto oceano de imagens sem que estas saiam do controle do custodiante público. O desafio contemporâneo não é apenas armazenar com performance, tarefa na qual arquiteturas distribuídas já se mostraram superiores (Macedo; Von Wangenheim; Dantas, 2015), mas sim garantir que o acesso ao dado seja estritamente efêmero e invisível ao olhar humano, mitigando o risco de vazamentos decorrentes do manuseio inadequado por usuários finais.

Dessa forma, este trabalho propõe a implementação de um método de acesso efêmero às imagens do STT, criando alternativas que eliminem a necessidade de salvamento em disco por parte do pesquisador. Evoluindo o tripé de segurança clássico do sistema, originalmente baseado em criptografia, protocolos digitais e assinaturas (Wallauer et al., 2008), utiliza-se a arquitetura de armazenamento de objetos MinIO associada ao protocolo de URLs pré-assinadas. Para viabilizar essa operação de forma coordenada e eficiente, a solução é estruturada sob o conceito de *pipeline* de dados, uma abordagem de engenharia que permite a decomposição de processos complexos em etapas autônomas e concorrentes (Ramamoorthy; Li, 1977). O *pipeline* desenvolvido permite que os modelos de IA consumam os dados diretamente em memória volátil (RAM), garantindo que a inovação tecnológica ocorra sem comprometer a privacidade do cidadão atendido pela saúde pública.

1.1 OBJETIVOS

Para resolver a problemática apresentada e fundamentar a viabilidade técnica da solução proposta, estabelecem-se os objetivos a seguir.

1.2 OBJETIVO GERAL

Desenvolver e validar uma arquitetura de *pipeline* de acesso efêmero a imagens médicas do Sistema de Telemedicina e Telessaúde (STT), utilizando armazenamento de objetos (MinIO) e URLs pré-assinadas para o processamento estritamente em memória RAM, visando garantir a Persistência Zero dos dados sensíveis e a conformidade com a LGPD, sem comprometer o desempenho e a qualidade diagnóstica no treinamento de modelos de Inteligência Artificial para a detecção de câncer de pele.

1.2.1 Objetivos Específicos

Para alcançar o objetivo geral de forma metódica e mensurável, traçaram-se os seguintes passos operacionais e de validação para a pesquisa:

- **Estruturar e balancear** uma base de dados de imagens dermatológicas simulada, garantindo a separação de instâncias de treino, validação e teste para evitar o vazamento de dados;
- **Desenvolver o *pipeline*** de ingestão que utilize URLs pré-assinadas do servidor MinIO do STT para carregar os binários de imagem diretamente em *buffer* de memória RAM (via classe `io.BytesIO`);
- **Validar a segurança e a Persistência Zero** do método através de técnicas de *snapshotting* de disco e auditoria de memória volátil, comprovando matematicamente a ausência de rastros físicos no *hardware* local;
- **Avaliar o desempenho e o tráfego de rede** do sistema através de um *benchmark* de vazão (*throughput*), dimensionando a eficácia da mitigação de gargalos de Entrada/Saída (I/O) via pré-carregamento assíncrono (*prefetching*);
- **Comprovar a eficácia médica e o custo temporal** do *pipeline* comparando o treinamento de uma rede neural convolucional operando sob a arquitetura efêmera contra o método tradicional de armazenamento físico em SSD, avaliando métricas como Acurácia, F1-Score e tempo de execução.

1.3 JUSTIFICATIVA

A justificativa deste trabalho reside na necessidade de conciliar o avanço tecnológico na saúde pública com a proteção rigorosa à privacidade do cidadão. A literatura aponta que sistemas de telemedicina de larga escala, como o STT, são fundamentais para reduzir custos operacionais e promover o bem-estar social através da democratização do diagnóstico especializado (Savaris et al., 2008). No entanto, a eficácia desses sistemas depende intrinsecamente da confiança na custódia e integridade dos dados informacionais.

A vulnerabilidade identificada no manuseio de dados por pesquisadores externos é corroborada pelo fato que o ser humano é o fator mais fraco e crítico na Segurança da Informação(SI), Lima e Roland (2025) . Sobre essa natureza do comportamento humano, Freire et al. (2017, p. 149) afirmam que:

[...] mesmo em ambientes computacionais fortemente guardados, os usuários se configuram com um fator preponderante para a ocorrência de incidentes, tendo em vista que o comportamento humano é mutável de acordo com a situação em que é submetido. As situações rotineiras, por falta de conhecimento sobre como agir, ou mesmo as de risco, pelo grau de responsabilidade, tornam o fator humano e suas ações uma vulnerabilidade bastante explorada pelas ameaças de SI.

Ao eliminar a necessidade de salvar imagens sensíveis em discos locais através de um pipeline de acesso efêmero, este trabalho mitiga o risco de vazamentos decorrentes de negligências, erros operacionais ou mesmo da imprevisibilidade das ações dos usuários finais. Assim, a proposta não apenas aprimora a camada técnica de segurança, mas cria uma barreira sistêmica onde as "fraquezas humanas" citadas pela literatura deixam de ser um ponto de exposição direta ao dado bruto.

1.3.1 Contribuições do Estudo

A contribuição central deste estudo reside na materialização técnica do princípio de persistência zero aplicado ao compartilhamento de dados sensíveis em saúde. Diferente de abordagens puramente normativas, este trabalho apresenta avanços tangíveis na engenharia de dados para pesquisa médica, destacando-se:

- **Paradigma de Persistência Zero:** Implementação de um fluxo de dados onde ativos digitais sensíveis residem exclusivamente em memória volátil durante o ciclo de processamento. Este avanço elimina a superfície de ataque associada ao armazenamento secundário (discos rígidos) em dispositivos de pesquisadores externos, resolvendo tecnicamente a brecha de custódia identificada na literatura.
- **Carregamento Assíncrono de Lotes:** Desenvolvimento de um mecanismo de carregamento antecipado e paralelo que mitiga a latência inerente ao acesso remoto via protocolo HTTP. Este avanço prova que a segurança e a privacidade não precisam comprometer o desempenho computacional, mantendo a taxa de ocupação da GPU em níveis equivalentes ao acesso local tradicional.
- **Arquitetura Escalável e Auditável para o STT:** Entrega de uma arquitetura escalável e auditável que permite ao Sistema de Telemedicina e Telessaúde (STT) disponibilizar seu vasto repositório para inovação em IA, garantindo conformidade técnica com a LGPD sem a necessidade de transferir a posse física dos dados.

- **Validação por Auditoria de Hardware:** Proposição de um método de validação que utiliza rastreamento de chamadas de sistema para comprovar a ausência de escrita em disco, oferecendo uma métrica objetiva de segurança operacional que transcende a confiança subjetiva no usuário.

1.3.2 Estrutura do Documento

Este trabalho está organizado de forma a transitar da problemática da custódia informacional na saúde digital até a materialização técnica de um ambiente de persistência zero. A narrativa do documento foi estruturada para demonstrar que a segurança da informação pode ser garantida pela própria arquitetura do sistema, sem comprometer a inovação tecnológica. A seguir, detalha-se a composição de cada capítulo:

No **Capítulo 2 - Fundamentação Teórica**, estabelece-se a base conceitual necessário para compreender o ecossistema do STT sob a ótica da Ciência da Informação. Abordam-se os marcos legais da LGPD e os princípios técnicos do armazenamento de objetos via MinIO, conectando a teoria da volatilidade da memória RAM à necessidade de proteção de dados sensíveis em fluxos de *Big Data*.

O **Capítulo 3 - Trabalhos Relacionados** mapeia o estado da arte e as soluções contemporâneas para o tratamento de imagens médicas. Este capítulo é fundamental para evidenciar como a proposta de acesso efêmero se diferencia das abordagens tradicionais de anonimização e criptografia, focando na lacuna da eliminação do rastro físico no hardware do pesquisador.

No **Capítulo 4 - Metodologia**, detalha-se o delineamento experimental da pesquisa. Descrevem-se os procedimentos de auditoria de hardware e as métricas quantitativas de vazão de rede que foram estabelecidas para validar se a segurança imposta pela persistência zero é operacionalmente viável em cenários de alto desempenho.

O **Capítulo 5 - Desenvolvimento** constitui o núcleo técnico da obra. Nele, descreve-se a engenharia do pipeline, detalhando desde a filtragem algorítmica do dataset até a implementação da injeção de binários em memória volátil via *io.BytesIO*. Este capítulo revela como a orquestração assíncrona foi utilizada para transpor os desafios de latência inerentes ao acesso remoto.

No **Capítulo 6 - Resultados e Discussões**, apresentam-se as evidências empíricas da solução. Através de relatórios de auditoria e *benchmarks* de desempenho, comprova-se matematicamente a eficácia da persistência zero e a manutenção da integridade dos dados originais para o processamento, discutindo-se o "custo de privacidade" em relação à agilidade do sistema.

Por fim, o **Capítulo 7 - Conclusões e Trabalhos Futuros** sintetiza as contribuições deste estudo para a Ciência da Informação, reafirmando o alcance dos objetivos propostos e sugerindo caminhos para a expansão da técnica para outros domínios de dados sensíveis.

Complementando o corpo do trabalho, o **Apêndice A** apresenta a Declaração de Uso de Inteligência Artificial, detalhando de forma transparente como ferramentas de modelos de linguagem apoiaram a revisão e a estruturação redacional deste documento, em conformidade com as diretrizes éticas da pesquisa científica contemporânea.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo detalha os pilares conceituais e tecnológicos que sustentam a proposta de acesso efêmero às imagens médicas utilizadas para o treinamento da IA. A discussão abrange desde a infraestrutura do STT até as ferramentas específicas de manipulação de dados em memória volátil.

2.1 O ECOSISTEMA DO SISTEMA DE TELEMEDICINA E TELESSAÚDE (STT)

A fundamentação sobre o desenvolvimento da saúde digital em Santa Catarina exige a compreensão do conceito de Telemedicina. Segundo Savaris et al. (2008), “o termo Telemedicina pode ser definido como a adoção e utilização de recursos de telecomunicação e outras tecnologias para o compartilhamento de informações médicas referentes a um dado paciente”. No contexto catarinense, essa definição materializa-se no Sistema de Telemedicina e Telessaúde (STT), que integra tecnologias, processos clínicos e protocolos de exames para viabilizar o diagnóstico a distância e o acompanhamento de pacientes no âmbito do Sistema Único de Saúde (SUS) (LabTelemed, 2026).

Historicamente, essa trajetória teve seu marco inicial muito anteriormente. Conforme detalhado por Savaris et al. (2008), o sistema iniciou suas operações por meio de um projeto piloto em maio de 2005, na cidade de Quilombo. O êxito e o conhecimento gerados por esse processo de implantação foram tão significativos que despertaram o interesse do Ministério da Saúde em 2006, servindo como base técnica e estratégica para a criação do Programa de Telessaúde Brasil. Esse programa nacional buscou aproveitar a infraestrutura tecnológica e de comunicação que já estava sendo disseminada no interior catarinense pela RCTM, fornecendo recursos de computação e conectividade para unidades de saúde anteriormente isoladas (Savaris et al., 2008).

Os resultados dessa integração tecnológica permitiram que o núcleo alcançasse presença em todos os municípios catarinenses, distribuídos entre a RCTM e o programa federal. Atualmente, o STT é sustentado por uma arquitetura de microserviços dividida em módulos independentes, o que permite a gestão eficiente de exames em doze modalidades distintas (Savaris et al., 2008; LabTelemed, 2026). A adoção desse modelo técnico possibilita a ampliação do acesso a especialistas, tradicionalmente concentrados em grandes centros urbanos, promovendo a democratização do atendimento especializado. Além disso, ao estruturar-se como uma plataforma de software livre, o sistema reduz custos relacionados a licenciamento e potencializa a escalabilidade para a rede pública de saúde (LabTelemed, 2026).

2.1.1 O Módulo de Telediagnóstico em Dermatologia

Dentro da arquitetura de microsserviços do STT, o módulo de Dermatologia, onde se concentra o foco deste trabalho, constitui um serviço de telediagnóstico especializado que exemplifica a aplicação da telemedicina assistencial assíncrona em larga escala. Este módulo foi desenvolvido como uma ferramenta estratégica para otimizar o fluxo de atendimento dermatológico no Sistema Único de Saúde (SUS), atuando de forma essencial na triagem, qualificação do encaminhamento e diagnóstico precoce de dermatoses (UFSC, 2026). Devido à sua expressiva abrangência territorial e ao fluxo contínuo de atendimentos, o sistema consolidou um vasto *dataset* de imagens de câncer de pele no Brasil, servindo como uma base fundamental para o desenvolvimento de pesquisas em visão computacional e inteligência artificial.

Conforme estabelecido pelas diretrizes institucionais da Teledermatologia UFSC (2026), o fluxo operacional do módulo tem por objetivo integrar de maneira eficiente a atenção primária e a atenção especializada, promovendo maior agilidade no atendimento e mitigando obstáculos geográficos ao acesso especializado. A Figura 1 apresenta, de forma esquemática, o ciclo completo da solicitação de telediagnóstico, evidenciando as etapas e interações essenciais para o funcionamento do sistema.

Figura 1 – Fluxo operacional da Teledermatologia no STT



Fonte: UFSC (2026).

A dinâmica operacional exemplificada na Figura 1 baseia-se na interação assíncrona

entre dois atores principais: o solicitante e o laudador. O processo inicia-se com o solicitante (médico ou técnico da atenção básica na Unidade Básica de Saúde - UBS), que é responsável por realizar o cadastro do paciente no sistema do STT, coletar a história clínica, incluindo tempo de evolução da lesão e comorbidades, e efetuar o registro fotográfico padronizado seguindo o manual (Wangenheim, Aldo von; Nunes, Daniel Holthausen; Wagner, Harley, 2017).

Concluída a solicitação e o envio das imagens via rede segura, o exame é disponibilizado no ambiente web para o laudador. Este ator, obrigatoriamente um médico dermatologista do núcleo especialista, analisa remotamente a queixa clínica e a qualidade das imagens. O laudador realiza então um laudo estruturado, definindo a morfologia da lesão, coloração e classificação diagnóstica, definindo uma conduta que pode ser o manejo na UBS ou o encaminhamento prioritário para cirurgia ou consulta presencial, podendo ainda invalidar o exame ou solicitar novas informações caso os dados sejam insuficientes (Wangenheim, A. von; Nunes, D. H.; Wagner, H., 2017; UFSC, 2026).

O diferencial deste módulo mora no rigor do protocolo de captura fotográfica estabelecido pela equipe do STT, que assegura a qualidade da informação para o telediagnóstico. Para que um laudo de suspeita de câncer de pele seja emitido, o exame deve conter obrigatoriamente três tipos de registros fotográficos complementares, conforme detalhado na Figura 2 e fundamentado no manual de registro fotográfico detalhado (Wangenheim; Wagner; Nogueira, 2013):

Figura 2 – Protocolo de registro fotográfico: (a) panorâmica, (b) aproximação com régua e (c) contato.



Fonte: Adaptado de Wangenheim, Wagner e Nogueira (2013).

1. **Foto Panorâmica (Figura 2a):** Fornece o contexto necessário para o médico especialista conseguir analisar as lesões no corpo do paciente;
2. **Foto de Aproximação com régua (Figura 2b):** Registrada a uma certa distância, utiliza obrigatoriamente uma régua e uma etiqueta de identificação para fornecer uma referência de escala;

3. Foto de Contato/Dermatoscopia (Figura 2c): Capturada com o auxílio de um dermatoscópio acoplado à câmera.

Essa padronização garante que o repositório do STT seja alimentado por um conjunto de dados informacional de alta qualidade, mas simultaneamente heterogêneo e denso. É sobre este volume de dados sensíveis e críticos que se aplica a necessidade do desenvolvimento de métodos de acesso efêmeros, garantindo que o processamento para treinamento de modelos de aprendizagem profunda ocorra sem que comprometa o dado e a custódia física das imagens.

2.2 DIRETRIZES DA LGPD PARA O TRATAMENTO DE DADOS SENSÍVEIS

O tratamento de dados no âmbito da saúde digital brasileira é regido pela Lei nº 13.709, de 14 de agosto de 2018, a Lei Geral de Proteção de Dados Pessoais (LGPD). No contexto da telemedicina, a legislação estabelece uma camada extraordinária de proteção ao categorizar a natureza das informações clínicas. Conforme a redação exata da lei, considera-se dado pessoal sensível o:

dado pessoal sobre origem racial ou étnica, convicção religiosa, opinião política, filiação a sindicato ou a organização de caráter religioso, filosófico ou político, dado referente à saúde ou à vida sexual, dado genético ou biométrico, quando vinculado a uma pessoa natural (BRASIL, 2018, Art. 5º, inciso II).

Essa classificação jurídica é de suma importância para o Sistema de Telemedicina e Telessaúde (STT). Conforme evidenciado por Minghelli et al. (2024), o Núcleo Telessaúde UFSC realiza um amplo e complexo tratamento de dados pessoais sensíveis. Este acervo inclui os dados clínicos dos pacientes, metadados descritivos de cada lesão indicada na solicitação médica e o próprio arquivo binário para cada imagem adquirida no exame. Devido ao seu alto potencial de causar discriminação ou danos significativos se utilizados indevidamente, esses ativos informacionais exigem mecanismos robustos para mitigar riscos às liberdades civis e aos direitos fundamentais de seus titulares.

A responsabilidade pela guarda e processamento dessas informações de saúde impõe a adoção obrigatória dos princípios da Segurança e da Prevenção. A LGPD determina expressamente, em seu Artigo 46, a obrigação técnica que recai sobre as instituições:

Os agentes de tratamento devem adotar medidas de segurança, técnicas e administrativas aptas a proteger os dados pessoais de acessos não autorizados e de situações acidentais ou ilícitas de destruição, perda, alteração, comunicação ou qualquer forma de tratamento inadequado ou ilícito (BRASIL, 2018, Art. 46, *caput*).

Sob a ótica da Ciência da Informação, o STT já emprega esforços notáveis para cumprir essas diretrizes. Conforme demonstram Sembay, Macedo e Filho (2024), o sistema utiliza rigorosos métodos de gerenciamento de dados de proveniência, rastreando o histórico

completo de “quem”, “como”, “quando” e “onde” os dados são gerados e manipulados na rede. Essa prática consolida a segurança, a confiabilidade e a qualidade dos dados de saúde enquanto transitam internamente na infraestrutura descentralizada da instituição (Sembay; Macedo; Filho, 2024).

Apesar da segurança do sistema, exportar esses dados para o *hardware* de pesquisadores externos cria um ponto cego. A partir do momento em que as imagens são salvas localmente, o controle sobre quem acessa os arquivos é perdido. Esta pesquisa surge para resolver exatamente esse problema prático, atuando como um complemento às regras de proteção que o STT já utiliza hoje.

A proposta do *pipeline* de acesso efêmero atua de forma complementar como uma medida de segurança técnica de ponta. Ao assegurar que a imagem dermatológica sensível seja processada exclusivamente em memória RAM e descartada imediatamente após o uso, o método materializa o conceito de *Privacy by Design* abordado por Mendes (2019). Ele impede, *by design*, a persistência do dado em dispositivos físicos de terceiros, neutralizando o risco de exposições acidentais ou acessos não autorizados.

2.3 O RISCO DE QUEBRA DE CUSTÓDIA NO COMPARTILHAMENTO DE DADOS

Na Ciência da Informação, ter a custódia de um dado significa ser o responsável técnico e legal por guardá-lo e protegê-lo. Em sistemas com múltiplas instâncias de implantação, como é o caso do STT, essa responsabilidade recai sobre a instituição gestora do ambiente, seja ela uma universidade, uma secretaria de saúde ou outro ente federativo. O grande problema de segurança da informação, conhecido como quebra de custódia, ocorre exatamente quando essas imagens sensíveis precisam ser disponibilizadas para pesquisadores externos. A partir do momento em que os arquivos são baixados e salvos no disco de um computador ou servidor de terceiros, a instituição perde totalmente o controle sobre quem acessa, copia ou armazena aquelas fotos. O cerne da vulnerabilidade não reside na presunção de má-fé do pesquisador, mas sim na impossibilidade técnica de estender as rígidas políticas de proteção do STT para o *hardware* de terceiros.

Para resolver esse tipo de conflito, Doneda (2020) argumenta que a proteção de dados não deve funcionar como uma barreira que simplesmente proíbe o acesso à informação, mas sim como uma gestão inteligente de como ela circula.

É exatamente neste ponto que o modelo de acesso efêmero se destaca. Ele permite que o pesquisador treine a sua Inteligência Artificial usando o acervo do STT, mas sem nunca receber a posse do arquivo em definitivo. Ao processar as imagens apenas na memória RAM e impedir qualquer gravação no disco local, o sistema garante que os dados dos pacientes desapareçam completamente da máquina do pesquisador ao fim da execução. Isso elimina o risco de arquivos ficarem esquecidos em pastas não monitoradas e evita vazamentos futuros (Doneda, 2020).

2.4 ARMAZENAMENTO DE OBJETOS COM MINIO

Para sustentar o expressivo volume de dados gerado pela tele dermatologia, o STT utiliza o MinIO, um servidor de armazenamento de objetos de alta performance, software livre e nativo para a nuvem. Desenvolvido em linguagem Go, o MinIO é projetado para ser compatível com a API do Amazon S3 (Simple Storage Service), estabelecendo-se como um padrão de mercado para infraestruturas de dados de larga escala (MinIO, 2026).

Diferente dos sistemas de arquivos tradicionais, que organizam os dados em uma hierarquia de pastas e subpastas (árvore), o armazenamento de objetos utiliza uma estrutura plana. Nela, cada dado é tratado como uma unidade isolada, o objeto, que reside dentro de um bucket. De acordo com a documentação oficial, cada objeto é composto obrigatoriamente por três elementos principais: o dado binário (as imagens médicas), um identificador único (key) e metadados extensíveis que descrevem o conteúdo e facilitam a sua recuperação (MinIO, 2026).

No âmbito do STT, o MinIO atua como o repositório central de ativos binários, operando em conjunto com o banco de dados da dermatologia. Enquanto o banco de dados relacional armazena os metadados do exame (ID do paciente, laudo estruturado, data e histórico clínico), o MinIO é o destino final das imagens capturadas. Segundo MinIO (2026), essa arquitetura de separação entre dados estruturados e dados binários (blobs) é fundamental por três razões:

- **Performance de I/O:** O MinIO é otimizado para lidar com arquivos binários grandes, garantindo que a gravação das fotos enviadas pelos médicos ocorra sem onerar o desempenho das transações do banco de dados;
- **Escalabilidade Horizontal:** Permite que o armazenamento cresça indefinidamente através da adição de novos nós, suportando o histórico acumulado de milhões de exames;
- **Integridade:** Utiliza tecnologias como *bit-rot protection* e *erasure coding* para garantir que as imagens médicas não sofram corrupção física ao longo dos anos.

2.4.1 Gestão de Acesso via URLs Pré-assinadas

O gerenciamento de objetos no MinIO baseia-se no protocolo HTTP e em suas operações fundamentais, habitualmente referenciadas como verbos S3: PUT (alteração), GET (recuperação), POST (criação) e DELETE (remoção). No fluxo de treinamento de modelos de Inteligência Artificial para tele dermatologia, a funcionalidade central reside na geração de URLs pré-assinadas (Presigned URLs).

Conforme a documentação técnica MinIO (2026), uma URL pré-assinada é o mecanismo que permite conceder acesso temporário a um objeto específico sem expor as

credenciais permanentes de segurança do servidor (*Access Key* e *Secret Key*). O funcionamento baseia-se num algoritmo de assinatura criptográfica (AWS Signature Version 4) que incorpora na própria URL os seguintes parâmetros:

1. As permissões de acesso;
2. A data e hora exatas da criação da assinatura;
3. O tempo de expiração, definido pelo sistema originador.

Essa funcionalidade constitui o elemento central do mecanismo de acesso efêmero proposto neste trabalho. Ao consumir uma estrutura de dados no formato JSON proveniente do banco de dados do STT, o pesquisador recebe URLs pré-assinadas que permitem requisitar diretamente os arquivos armazenados no MinIO para a memória volátil (RAM), sem a necessidade de armazenamento permanente no dispositivo de acesso.

2.5 PERSISTÊNCIA VERSUS EFEMERIDADE: MEMÓRIA RAM E ARMAZENAMENTO SECUNDÁRIO

A fundamentação teórica sobre a custódia de dados em sistemas de telessaúde exige uma compreensão profunda da hierarquia de memória e da natureza do suporte físico. A distinção entre o armazenamento persistente e o acesso efêmero é o que garante a viabilidade técnica e a conformidade ética deste *pipeline*.

2.5.1 Definição, Volatilidade e Performance da Memória RAM

A memória principal, ou RAM, atua como o espaço de trabalho central do sistema computacional. É neste ambiente que são mantidos os processos em execução, as *threads*, as bibliotecas compartilhadas e o próprio núcleo do sistema operacional (Maziero, 2022). Estruturalmente, é composta por uma vasta sequência de *bytes*, onde cada unidade possui um endereço único utilizado pelo processador, exigindo um esforço constante de gerência por parte do sistema (Maziero, 2022).

Para compreender a escolha arquitetural de segurança deste trabalho, é imprescindível definir a natureza física de gravação deste componente. Apoiando-se nos conceitos de infraestrutura de dados de Somasundaram, Shrivastava e Services (2009):

Memória RAM (Memória de acesso aleatório): Permite o acesso direto a qualquer local da memória e pode ter seus dados gravados em si mesma ou lidos a partir dela. A memória RAM é volátil; este tipo de memória requer um fornecimento constante de energia para manter o conteúdo da célula de memória. Os dados são apagados quando a energia do sistema é desligada ou interrompida.

Essa característica essencial de volatilidade atua como o pilar da persistência zero. Ao processar imagens e diagnósticos médicos sensíveis exclusivamente em memória,

impedindo sua gravação em meios magnéticos ou de estado sólido (SSDs), garante-se que o encerramento do processo destitua a informação de qualquer rastro digital recuperável no *hardware*.

Além do rigoroso isolamento por efemeridade, a alocação de dados na memória RAM entrega vantagens críticas de performance para o sistema. Uma unidade virtual baseada em RAM fornece um desempenho de entrada e saída (I/O) excepcionalmente rápido, com larguras de banda que podem superar em três vezes a de um SSD com interface NVMe moderno (Kingston Technology, 2023). A superioridade técnica do acesso em memória em relação aos meios de armazenamento persistente, mensurada por sua baixíssima latência e alta taxa de transferência, é evidenciada no Quadro 1.

Quadro 1 – Comparação de performance entre meios de armazenamento.

Meio	Tempo de acesso (latência)	Taxa de transferência
Memória RAM	60 ns	1 GB/s (1 ns por byte)
Disco rígido SATA	5 ms*	100 MB/s (10 ns por byte)

Fonte: Adaptado de Maziero (2022).

2.5.2 Armazenamento Efêmero e Persistência Zero

Para definir o conceito central que intitula este trabalho, é necessário estabelecer formalmente a premissa de transitoriedade na qual a solução se baseia. Apoiando-se na definição de mercado sobre infraestruturas de nuvem, a Dremio (2026) determina que:

O armazenamento efêmero, frequentemente referido como armazenamento temporário ou não persistente, é um aspecto da computação em nuvem em que os dados armazenados estão ativamente disponíveis apenas durante a vida útil de uma instância de computação específica. Uma vez que a instância é encerrada, quaisquer dados retidos no armazenamento efêmero são perdidos, daí o nome, que deriva do termo “efêmero”, significando “que dura por muito pouco tempo” (tradução nossa).

No contexto de infraestruturas distribuídas e arquiteturas de nuvem, o armazenamento efêmero é empregado como uma estratégia na qual réplicas de arquivos são alocadas nas máquinas de um aglomerado (*cluster*) estritamente durante o intervalo necessário para a execução de uma tarefa (Huaman, 2010). Essa modalidade é projetada de forma deliberada para dados que possuam um ciclo de vida curto, frequentemente durando apenas algumas horas, de modo que o objetivo primário seja a disponibilidade para processamento imediato, sem intenção de preservação em longo prazo.

Complementando essa visão sob a ótica da infraestrutura de serviços, observa-se que o armazenamento efêmero caracteriza-se por ser um sistema de arquivos montado diretamente numa instância de máquina virtual. Segundo Rodrigues (2013), este modelo diferencia-se do chamado armazenamento de volume, que constitui a modalidade persistente e independente das instâncias de processamento. Ao optar pelo paradigma efêmero,

o sistema garante que o suporte de dados esteja intrinsecamente ligado à existência do processo ativo, sem a necessidade de formatações ou partições permanentes que retenham a informação após o uso (Rodrigues, 2013).

A partir dessa premissa técnica, consolida-se o conceito de “acesso efêmero” aplicado à pesquisa em teledermatologia. Diferentemente das metodologias clássicas de treinamento de Inteligência Artificial, que exigem o *download* prévio do *dataset* médico para o disco rígido local, o acesso efêmero designa um fluxo onde o ativo informacional sensível é requisitado do repositório central e mantido exclusivamente no estado de processamento ativo na memória RAM. Assim que o modelo de rede neural conclui a extração de características daquela amostra específica, o *buffer* volátil é liberado ou sobrescrito pelo próximo lote.

A “persistência zero”, por sua vez, constitui a garantia de segurança essencial deste modelo. Ao condicionar a existência da informação na máquina local unicamente à execução do *script* em memória volátil, estabelece-se um bloqueio físico contra a retenção de dados residuais. Assim que o processamento é concluído, ou mesmo em cenários de interrupção abrupta, a imagem médica é apagada devido ao corte no fornecimento de energia para o endereço de memória. Essa arquitetura assegura que os exames dermatológicos não habitem dispositivos de armazenamento secundário (SSDs ou HDDs) controlados pelo pesquisador externo, neutralizando ameaças associadas ao manuseio humano inadequado e cumprindo os rigorosos padrões éticos e legais da saúde digital.

2.6 REQUISITOS DE CONSUMO DE DADOS EM APRENDIZADO PROFUNDO

Embora o foco deste trabalho resida na segurança e na infraestrutura, a justificativa para a alta performance do *pipeline* encontra-se nos requisitos operacionais dos modelos de Aprendizado Profundo (*Deep Learning*). Redes Neurais Convolucionais (CNNs), como a arquitetura ResNet utilizada em teledermatologia, operam através do processamento massivo de volumes de dados binários organizados em lotes (*batches*).

A eficiência desse processo é intrinsecamente dependente da vazão de entrada e saída (I/O). Como os modelos realizam múltiplas passagens pelo conjunto de dados (*epochs*) para a extração de características, qualquer latência no acesso ao disco rígido ou na requisição de rede torna o treinamento inviável temporalmente. Portanto, a utilização da memória RAM não apenas atende aos critérios de segurança por volatilidade, mas provê a largura de banda necessária para que os tensores da GPU sejam alimentados sem interrupções (Maziero, 2022). A orquestração via *buffer* de memória garante que o requisito de velocidade da Inteligência Artificial seja atendido sem violar a premissa de não persistência em armazenamento secundário.

2.7 BIBLIOTECAS PYTHON PARA PROCESSAMENTO DE DADOS EM MEMÓRIA

A consolidação da linguagem Python no domínio da Ciência da Informação e da Ciência de Dados fundamenta-se em sua característica de linguagem de alto nível e de propósito geral. Conforme destacam Silva et al. (2026), o Python “devido à sua simplicidade, versatilidade e ampla aceitação acadêmica e profissional”, permite a automação de processos de análise e a manipulação de grandes volumes de dados de forma estratégica. Essa robustez é ampliada por uma arquitetura que prioriza a legibilidade, facilitando o desenvolvimento de fluxos informacionais complexos em ambientes de pesquisa científica.

2.7.1 Manipulação de Fluxos e a Segurança em Memória Volátil

No contexto da Segurança da Informação, a minimização do rastro digital em dispositivos de armazenamento permanente é uma estratégia técnica para mitigar vulnerabilidades. Freire et al. (2017) apontam que o fator humano é, frequentemente, o “elemento mais fraco e mais importante na gestão de segurança da informação”, sendo muito mais difícil blindar o dado digital do que proteger a máquina física onde ele roda.

Para responder a essa vulnerabilidade, o ecossistema Python provê recursos para o tratamento de dados binários diretamente na memória, já dentro das suas vastas bibliotecas, sem a necessidade de persistência em disco rígido. O fundamento dessa operação reside no módulo `io`, especificamente na classe `BytesIO`. Segundo a documentação oficial da linguagem, o `io.BytesIO` permite a criação de um *buffer* binário que se comporta como um arquivo em memória, viabilizando o processamento de fluxos de dados (*streams*) de maneira volátil (Python, 2026). Sob a ótica da Ciência da Informação, essa técnica permite gerenciar que o dado informacional sensível permaneça sob custódia estritamente sistêmica, desaparecendo ao fim da execução e reduzindo a superfície de exposição ao componente humano.

2.7.2 Teoria do Carregamento Assíncrono de Lotes

A eficiência no processamento de grandes conjuntos de dados, como as imagens de teledermatologia, exige uma manipulação que evite gargalos de entrada e saída (I/O), garantindo que a ferramenta seja funcional para futuras investigações científicas. A fundamentação teórica para o processamento de alto desempenho em Python baseia-se na execução paralela e assíncrona, onde a recuperação da informação ocorre de forma antecipada ao seu uso efetivo.

Essa abordagem, conhecida como *prefetching*, fundamenta-se na separação de responsabilidades entre núcleos de processamento (Renzulli; Wagner, 2025). Segundo Xbeat (2023), em um projeto no GitHub, o pré-carregamento atua como uma técnica de otimização que disponibiliza os dados na memória antes da requisição imediata pelo algoritmo, o que resulta em ganhos significativos de performance ao lidar com processamentos de

alta volumetria. Enquanto uma unidade dedica-se ao processamento de um lote (*batch*) de dados já presente na memória, processos secundários realizam a busca e a decodificação do lote subsequente via rede (Python, 2026). Esse modelo de fluxo garante que a integridade e a disponibilidade do sistema sejam mantidas, assegurando que a segurança imposta pelo acesso efêmero não comprometa a viabilidade temporal do processamento de dados em larga escala (Savaris et al., 2008).

2.8 PIPELINE DE ACESSO EFÊMERO

O conceito de *pipeline* é fundamental na engenharia de *software* e na engenharia de dados, aplicando-se diretamente à solução proposta neste trabalho. A decomposição de processos complexos em etapas menores e mais gerenciáveis é uma estratégia que reflete como sistemas eficientes resolvem problemas de processamento em larga escala.

A definição clássica de *pipeline*, estabelecida por Ramamoorthy e Li (1977, p. 62, tradução nossa), descreve-o como “a técnica de decompor um processo sequencial repetido em subprocessos, cada um dos quais pode ser executado eficientemente em um módulo autônomo dedicado que opera concorrentemente com os outros”. Esta abordagem permite que múltiplos processos sejam executados simultaneamente, análoga ao funcionamento de uma linha de montagem industrial. Neste modelo, enquanto uma etapa processa um item, a etapa seguinte já está processando o item anterior. O resultado é um aumento significativo do desempenho do sistema sem necessariamente duplicar o custo computacional de cada etapa.

No contexto contemporâneo da engenharia de dados, a Microsoft (2024) define um *pipeline* de dados como um processo de integração que “consolida dados de diversas fontes em um armazenamento de dados unificado”. Este processo, frequentemente referenciado como ETL (Extração, Transformação, Carregamento), compreende três fases distintas que trabalham em conjunto:

- **Extração:** Coleta de dados de múltiplas fontes, cada uma potencialmente em formatos diferentes. No contexto de imagens médicas, significa requisitar os arquivos do repositório remoto sem necessariamente salvá-los em disco.
- **Transformação:** Modificação dos dados coletados para um formato utilizável. Isso pode envolver conversão de formatos, limpeza, validação ou enriquecimento. Para imagens DICOM ou PNG, significa convertê-las em estruturas de dados que possam ser processadas eficientemente em memória.
- **Carregamento:** Movimentação dos dados transformados para seu destino final, seja um banco de dados, um *data warehouse* ou, neste trabalho, diretamente para processamento em memória volátil.

Conforme destacado pela Microsoft (2024), as três fases de ETL são geralmente executadas em paralelo para economizar tempo. Enquanto uma imagem está sendo extraída do repositório remoto, outra já está sendo transformada em memória, e uma terceira está pronta para ser consumida. Este paralelismo é o que permite que sistemas modernos processem volumes massivos de dados sem criar gargalos.

Burgos et al. (2022, p. 46, tradução nossa) expandem essa visão para contextos de escala ainda maior, descrevendo um *pipeline* de dados como “[um processo de ingestão de dados] que funciona com dados atuais e históricos, manipula agregados eficientemente e pode lidar com qualquer escala”. Esta perspectiva é particularmente relevante para ambientes de saúde digital em larga escala, como o Sistema de Telemedicina e Telessaúde (STT). O STT não processa apenas alguns exames; ele gerencia milhões de imagens dermatológicas acumuladas ao longo de anos. Uma arquitetura de *pipeline* bem projetada permite que este volume seja manipulado de forma eficiente, mantendo simultaneamente elevados padrões de segurança e integridade dos dados.

No domínio específico de Inteligência Artificial, Steidl, Felderer e Ramler (2023) caracterizam um *pipeline* como um processo contínuo que compreende quatro estágios interdependentes: **Data Handling** (coleta, limpeza, transformação e validação de dados), **Model Learning** (treinamento e avaliação de modelos), **Software Development** (desenvolvimento, teste e integração de código) e **System Operations** (implantação, monitoramento e manutenção em produção). Embora o foco deste trabalho resida no estágio de *Data Handling*, é importante reconhecer que um *pipeline* bem estruturado estabelece as fundações para todas as etapas subsequentes.

2.8.1 Por Que Este Trabalho Implementa um Pipeline de Acesso Efêmero

A estruturação deste trabalho em torno de um *pipeline* de acesso efêmero emerge de uma necessidade prática e de uma compreensão profunda dos desafios que cercam o compartilhamento de dados sensíveis em ambientes de pesquisa. Um *pipeline* de acesso efêmero é, fundamentalmente, um sistema projetado para permitir que dados sensíveis sejam processados sem nunca serem persistidos em disco local, ou seja, sem nunca serem salvos permanentemente no *hardware* de pesquisadores externos.

No fluxo tradicional, um pesquisador solicita acesso a imagens dermatológicas do STT, recebe os arquivos e salva-os em seu computador para então processá-los. O problema fundamental reside no fato de que, uma vez salvas localmente, essas imagens ficam fora do controle direto do STT. Elas podem ser esquecidas em pastas, copiadas acidentalmente ou acessadas por terceiros. Mesmo com as melhores intenções, o componente humano introduz vulnerabilidades que não podem ser completamente eliminadas (Freire et al., 2017).

O *pipeline* de acesso efêmero inverte essa lógica. Em vez de enviar os dados para o pesquisador, o sistema oferece um acesso temporário e controlado aos dados. As ima-

gens são requisitadas sob demanda, processadas exclusivamente em memória RAM, e descartadas imediatamente após o uso. Nenhum arquivo é salvo em disco. Nenhuma cópia permanente é criada. O pesquisador obtém exatamente o que precisa para sua pesquisa sem nunca receber a posse permanente dos dados.

2.8.1.1 Primeira Convergência: *Pipeline Architecture* como Decomposição de Processos

A arquitetura proposta neste trabalho segue o paradigma estabelecido por Ramamoorthy e Li (1977), implementando a decomposição de um processo complexo em subprocessos autônomos que operam em paralelo. O *pipeline* pode ser concebido como uma série de estações de trabalho, cada uma especializada em uma tarefa específica:

- **Estação 1 (URLHandler):** Responsável por gerar URLs pré-assinadas com expiração temporal. Estas URLs funcionam como mecanismos de acesso temporário que permitem acesso a um objeto específico no MinIO por um período limitado. Após a expiração, o acesso deixa de funcionar.
- **Estação 2 (DataConverter):** Recebe a URL válida e executa a transformação de imagens DICOM ou PNG em estruturas de dados NumPy que residem exclusivamente em memória RAM. Este é um passo crítico: a imagem é requisitada, convertida e mantida apenas em memória volátil.
- **Estação 3 (Validator):** Verifica a integridade dos dados transformados, assegurando que a imagem em memória corresponde exatamente ao original no MinIO, sem corrupção ou alteração. Também audita que nenhuma gravação em disco ocorreu.
- **Estação 4 (Garbage Collector):** Após o processamento, libera a memória RAM, descartando completamente a imagem. Neste ponto, qualquer traço dos dados sensíveis é eliminado da máquina do pesquisador.

Cada estação opera de forma independente, mas coordenada. Enquanto a Estação 1 gera uma URL para a próxima imagem, a Estação 2 já está convertendo a imagem anterior, a Estação 3 está validando a anterior, e a Estação 4 está limpando a memória. Este paralelismo permite que o sistema processe múltiplas imagens eficientemente, sem criar gargalos. Conforme Ramamoorthy e Li (1977), essa abordagem implementa *buffering* entre estágios e resolução de conflitos através de mecanismos de controle de fluxo, garantindo que nenhuma etapa fique atrasada ou avance além de sua capacidade.

2.8.1.2 Segunda Convergência: *Data Pipeline Architecture* com Persistência Zero

O *pipeline* proposto implementa explicitamente o modelo ETL, mas com uma variação inovadora que reflete a necessidade específica de proteger dados sensíveis. Conforme

a Microsoft (2024), um *pipeline* ETL tradicional compreende extração, transformação e carregamento. Neste trabalho, esse modelo é adaptado para o contexto de acesso efêmero.

A inovação crucial reside na eliminação da fase de *staging* tradicional. Conforme Burgos et al. (2022), *pipelines* convencionais frequentemente utilizam “tabelas de preparo ou áreas de armazenamento intermediário que contêm temporariamente os dados à medida que são processados”. Este trabalho elimina deliberadamente essa fase. Não há arquivos intermediários, não há *caches* em disco, não há zonas de trabalho onde dados sensíveis possam ficar esquecidos. Os dados existem apenas em memória volátil, que é automaticamente limpa quando não mais necessária.

Adicionalmente, o *pipeline* implementa execução paralela conforme recomendado pela Microsoft (2024). As fases de ETL não ocorrem sequencialmente; elas ocorrem em paralelo. Enquanto uma imagem está sendo extraída, outra está sendo transformada, e uma terceira está sendo carregada. Este carregamento antecipado de dados mitiga gargalos de rede impostos pelo acesso remoto ao MinIO, garantindo que o pesquisador não fique aguardando por dados.

2.8.1.3 Terceira Convergência: Pipeline como Componente de Data Handling em IA

Embora o foco deste trabalho resida especificamente no *pipeline* de acesso efêmero, é importante situá-lo dentro do contexto mais amplo de *pipelines* em Inteligência Artificial. Steidl, Felderer e Ramler (2023) descrevem um *pipeline* de IA como um processo contínuo que compreende quatro estágios. Este trabalho concentra-se no primeiro estágio, o **Data Handling**, que compreende coleta, limpeza, transformação e validação de dados.

Diferentemente de trabalhos que focam em otimização de treinamento ou arquitetura de modelos, este trabalho reconhece que o *Data Handling* é um estágio crítico e frequentemente negligenciado. Um *pipeline* robusto de acesso efêmero estabelece as fundações para que pesquisadores possam trabalhar com dados sensíveis de forma segura e confiável. Os estágios subsequentes, *Model Learning*, *Software Development* e *System Operations*, herdam os benefícios de um *pipeline* bem estruturado.

O Quadro 2 situa este trabalho dentro do contexto mais amplo de *pipelines* de IA:

Quadro 2 – Localização deste Trabalho no Pipeline de IA.

Estágio	Descrição	Escopo deste Trabalho
Data Handling	Coleta, limpeza, transformação e validação de dados	Foco principal: <i>Pipeline</i> de acesso efêmero, URLs pré-assinadas, persistência zero
Model Learning	Treinamento e avaliação de modelos de IA	Fora do escopo: utiliza código existente e adaptado
Software Development	Desenvolvimento, teste e integração de código	Parcialmente coberto: implementação do <i>pipeline</i>
System Operations	Implantação, monitoramento e manutenção	Validação operacional do <i>pipeline</i>

Fonte: Adaptado de Steidl, Felderer e Ramler (2023).

2.8.2 A Inovação: Persistência Zero como Princípio de Design

A contribuição específica deste trabalho é a implementação de um *pipeline* que materializa o princípio de persistência zero. Dados sensíveis nunca são salvos em disco local, existindo apenas em memória volátil durante o processamento. Isso não é meramente uma escolha técnica, é uma decisão de *design* que reflete uma compreensão profunda dos desafios de segurança em ambientes de pesquisa colaborativa Mendes (2019).

Conforme argumentado na Justificativa deste trabalho com base na literatura, o componente humano é frequentemente o elo mais fraco da segurança da informação (Freire et al., 2017). Não por negligência ou má intenção dos pesquisadores, mas porque erros humanos são inevitáveis em qualquer sistema. Um pesquisador pode esquecer de deletar arquivos, pode salvá-los em um local inseguro ou pode, acidentalmente, compartilhá-los com terceiros. Ao eliminar a necessidade de salvar imagens em disco local, este *pipeline* cria uma barreira sistêmica onde essas vulnerabilidades humanas deixam de ser pontos de exposição direta ao dado bruto.

A validação da eficácia dessa abordagem ocorre em múltiplas dimensões:

1. **Integridade de Dados:** Verificação de que as imagens processadas em RAM correspondem exatamente aos dados originais no MinIO, sem corrupção ou alteração. Isso garante que o pesquisador está trabalhando com dados confiáveis.
2. **Desempenho de I/O:** Análise comparativa de tempos de carregamento entre o método de acesso efêmero (remoto mais RAM) e o método tradicional (disco local). Este trabalho demonstra que a segurança não compromete a produtividade; de fato, o acesso remoto pode ser tão eficiente quanto o acesso local.
3. **Segurança Operacional:** Auditoria de *hardware* que comprova a ausência de gravações em disco local, validando o princípio de persistência zero. Isso oferece garantias concretas de que os dados nunca foram persistidos.

Através dessas três dimensões, este trabalho estabelece que é possível permitir pesquisa com dados sensíveis sem comprometer a segurança. O *pipeline* de acesso efêmero não é uma barreira que proíbe o acesso; é um facilitador que permite acesso seguro e controlado.

3 TRABALHOS RELACIONADOS

No contexto do Sistema de Telemedicina e Telessaúde (STT), pesquisadores buscam constantemente utilizar o vasto repositório de imagens médicas para desenvolver Inteligências Artificiais capazes de auxiliar na detecção precoce do câncer de pele e outras dermatoses. No entanto, o acesso a esses dados por cientistas externos esbarra em uma severa barreira ética e legal: Como garantir a proteção da privacidade do paciente sem inviabilizar o processamento computacional ou prejudicar a qualidade do diagnóstico final? Para justificar a arquitetura de acesso em memória volátil proposta neste TCC como a solução para este dilema, foram selecionados onze trabalhos centrais que abordam a segurança de dados e a infraestrutura em nuvem.

A literatura atual oferece caminhos para tentar proteger a privacidade na pesquisa médica, destacando-se duas abordagens principais: o bloqueio por criptografia e a alteração da imagem.

Representando a primeira abordagem, Kaissis et al. (2020), em estudo publicado na revista *Nature Machine Intelligence*, propõem o uso de Aprendizado Federado e criptografia pesada, onde os dados nunca saem dos hospitais. Apesar de garantir alta proteção, os autores alertam para a extrema exigência de poder de processamento. Para pesquisadores comuns e sistemas públicos de saúde, essa complexidade exige supercomputadores caros e gera uma lentidão que, na prática, inviabiliza o processamento em larga escala.

Na segunda abordagem, tenta-se proteger a identidade alterando o dado original. A tese de Coelho (2025) explora a anonimização de dados de saúde na Internet das Coisas (IoT). O autor comprova que modificar as informações do paciente para ocultar sua identidade reduz a precisão dos modelos de Inteligência Artificial. Em exames críticos de teledermatologia, alterar os *pixels* ou remover detalhes da lesão para fins de anonimização fatalmente irá prejudicar a qualidade do diagnóstico final, induzindo a IA ao erro.

O presente trabalho oferece aos pesquisadores uma terceira via para romper essa barreira. Em vez de encarecer a infraestrutura (Kaissis et al., 2020) ou alterar as características da imagem reduzindo o acerto da IA (Coelho, 2025), esta pesquisa entrega o arquivo original de forma leve e rápida. A proteção dos pacientes do STT ocorre exclusivamente na camada de *hardware*: a imagem médica é processada na memória RAM volátil do pesquisador (via classe `io.BytesIO`) e seu acesso ao disco rígido é fisicamente bloqueado. Essa Persistência Zero garante a segurança exigida por lei, mantendo a velocidade de processamento e a precisão médica intactas.

A viabilidade técnica de utilizar a nuvem para sustentar essa arquitetura é comprovada por Bendazzoli, Persson, Astaraki et al. (2025). Na revista *npj Digital Medicine*, os autores apresentam a plataforma MAIA, que utiliza o servidor MinIO e URLs temporárias para organizar imagens médicas. Avançando para a proteção do dispositivo final, Moses (2026) descreve uma arquitetura de transmissão contínua (*streaming*) onde os dados vão

direto para um *buffer* de memória, o que reduz custos e aumenta a segurança ao evitar o *download* de arquivos para o disco local.

O uso específico da classe `io.BytesIO` para lidar com arquivos binários na memória durante o treinamento de IA é discutido por Rand (2022). O autor demonstra que carregar imagens diretamente da nuvem (S3) para a RAM corta o atraso gerado pela gravação no disco. No caso de imagens médicas de alta resolução, Wang et al. (2024) sugerem o carregamento sob demanda (*Lazy Loading*) para evitar o travamento do computador, uma técnica que este TCC expande ao organizar o processamento em pequenos lotes efêmeros.

Sobre a necessidade de manter a velocidade do sistema de pesquisa, Maknee (2025) mostram que escolher mal a forma de armazenar dados na nuvem gera perda de tempo de treinamento. Esse dado confirma a necessidade da estratégia de carregamento antecipado (*prefetching*) usada neste TCC para esconder a lentidão da internet. Em contrapartida, Corewave (2025) provam que sistemas de armazenamento de objetos otimizados conseguem entregar os dados a taxas superiores a 2 GB/s, superando os discos locais comuns. A eficiência de processar dados sem fazer cópias físicas também é defendida pela tecnologia *Zero-Copy* de Hundt e Martinez (2021).

No campo da governança em saúde digital, Zeydan, Arslan e Liyanage (2024), em publicação na *IEEE Access*, discutem como organizar a Inteligência Artificial, destacando a importância da segurança de ponta a ponta. Por fim, validando a essência tecnológica deste TCC, Veerapaneni et al. (2026) publicaram na revista *Frontiers in Digital Health* a criação de um banco de dados que funciona inteiramente na memória RAM para monitorar pacientes em estado crítico. Os autores provam que o disco rígido tradicional é o principal causador de lentidão na IA médica, justificando de forma direta a escolha desta pesquisa por um *pipeline* focado na altíssima velocidade e na volatilidade da memória RAM.

O Quadro 3 sintetiza a comparação entre os onze trabalhos citados e a proposta atual.

Quadro 3 – Comparação entre Trabalhos Relacionados e a Proposta Atual.

Trabalho	Abordagem Principal	Tecnologia / Método	Vantagem / Foco	Limitação / Lacuna
Kaissis et al. (2020)	Aprendizado Federado	Criptografia Pesada	Privacidade Rigorosa	Inviabiliza o Processamento
Coelho (2025)	Anonimização	k -anonimato	Proteção do Dado	Prejudica o Diagnóstico Final
Bendazzoli, Persson, Astaraki et al. (2025)	Plataforma MAIA	MinIO / URLs S3	Colaboração e Escala	Foco em Gestão (não RAM)
Moses (2026)	Streaming de Dados	<i>Buffer</i> de Memória	Segurança Operacional	Sem foco na LGPD
Rand (2022)	Acesso em Memória	io.BytesIO / S3	Velocidade do Sistema	Foco em Código Genérico
Maknee (2025)	Desempenho de Rede	Otimização S3	Redução de Atrasos	Foco em Custo Financeiro
Corewave (2025)	Velocidade de Leitura	Armazenamento Nuvem	Alta Escalabilidade	Foco em Infraestrutura
Hundt e Martinez (2021)	Integração Rápida	<i>Zero-Copy</i>	Economia de Memória	Foco exclusivo em GPU
Wang et al. (2024)	Carregamento Lento	Geração sob demanda	Evita Travamentos	Foco apenas em Memória
Zeydan, Arslan e Liyanage (2024)	IA Distribuída	Gestão em Nuvem	Governança de Dados	Foco em Organização
Veerapaneni et al. (2026)	Banco de Dados	Exclusivo em RAM	Alta Velocidade	Foco em UTI / Hospitais
Proposta Atual	Acesso Efêmero	MinIO / BytesIO / Prefetching	Persistência Zero e Imagem Intacta	Dependência de Internet Estável

Fonte: Elaborado pelo autor (2026).

4 METODOLOGIA

A elaboração desta pesquisa fundamenta-se em um conjunto de abordagens metodológicas integradas que visam caracterizar o desenvolvimento e a validação de um *pipeline* de acesso efêmero a ativos informacionais sensíveis. Partindo do princípio fundamental de que a investigação científica é um processo estruturado e planejado, Gil (2002, p. 17) afirma que: “pode-se definir pesquisa como o procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos”. No contexto deste trabalho, o problema reside na vulnerabilidade da custódia de dados em treinamentos de Inteligência Artificial, buscando-se uma solução técnica através da transição de um modelo de persistência física para um processamento estritamente em memória volátil.

4.1 CARACTERIZAÇÃO DA PESQUISA

Esta investigação define-se, primordialmente, como uma pesquisa experimental, visto que o ambiente de testes constitui uma simulação controlada e artificial que reproduz o cenário real de treinamento de IA utilizando imagens dermatológicas do HAM10000 (Tschandl; Rosendahl; Kittler, 2018) no sistema do STT. No contexto deste TCC, a experimentação permite validar a implementação do *pipeline* em um ambiente que reproduz fielmente as requisições ao servidor de objetos MinIO. O rigor dessa escolha justifica-se pois, segundo Gil (2002, p. 48), “a pesquisa experimental constitui o delineamento mais prestigiado nos meios científicos”. O autor complementa definindo que:

Essencialmente, a pesquisa experimental consiste em determinar um objeto de estudo, selecionar as variáveis que seriam capazes de influenciá-lo, definir as formas de controle e de observação dos efeitos que a variável produz no objeto (Gil, 2002, p. 48).

Complementarmente, a pesquisa assume um caráter bibliográfico ao selecionar e analisar materiais publicados para fundamentar a investigação sobre segurança de dados e orquestração assíncrona. Como descreve Gil (2002, p. 44), “a pesquisa bibliográfica é desenvolvida com base em material já elaborado, constituído principalmente de livros e artigos científicos”. Sua aplicação neste trabalho justifica-se tecnicamente, pois “a principal vantagem da pesquisa bibliográfica reside no fato de permitir ao investigador a cobertura de uma gama de fenômenos muito mais ampla do que aquela que poderia pesquisar diretamente” (Gil, 2002, p. 45).

Por fim, o estudo é caracterizado como quantitativo ao enfatizar resultados mensuráveis e dados numéricos para avaliar a viabilidade da solução. As métricas de desempenho concentram-se na comparação da vazão de entrada e saída (I/O) e nos tempos de resposta entre o modelo tradicional de armazenamento em disco rígido e o modelo de carregamento em memória *prefetching*.

4.2 PROCEDIMENTOS METODOLÓGICOS

Para garantir que o trabalho não se limite à proposição teórica, os procedimentos metodológicos foram estruturados com foco exclusivo na validação empírica do sistema. O objetivo desta seção é definir os instrumentos, o ambiente e as métricas que serão utilizados para aferir se a arquitetura construída no desenvolvimento atende aos critérios de segurança, desempenho e qualidade médica.

A estratégia de experimentação foi fatiada em quatro eixos de validação:

4.2.1 Ambiente de Experimentação e Base de Dados

A validação de arquiteturas de segurança exige um ambiente controlado que isole o experimento de variáveis externas. Para simular a infraestrutura do Sistema de Telemedicina e Telessaúde (STT), os testes serão conduzidos no ambiente *Jupyter VLAB* da Universidade Federal de Santa Catarina (UFSC), utilizando a GPU NVIDIA DGX H100 e um servidor de objetos MinIO provido para os ensaios.

Para resguardar a privacidade e não infringir as normativas éticas de manipulação de prontuários reais, a validação utilizará a base de dados pública de pesquisa dermatológica HAM10000 (Tschandl; Rosendahl; Kittler, 2018). Do ponto de vista metodológico de inteligência artificial, o *dataset* será submetido a técnicas de *oversampling* para nivelamento de classes e particionado em 70% para treinamento, 15% para validação e 15% para testes finais. Para evitar o vazamento de dados, estipulou-se que a divisão ocorrerá pelo identificador do paciente, garantindo que o conjunto de teste apresente cenários totalmente inéditos ao modelo avaliado.

4.2.2 Validação de Segurança e Persistência Zero

Para atestar o cumprimento técnico da LGPD e a conformidade com a Persistência Zero, a metodologia de validação não pode depender da presunção de que o código funciona. A auditoria deve comprovar a ausência de rastros físicos no *hardware*.

Para isso, o experimento adotará o método de *snapshotting* (Alves, 2016). O procedimento consiste em mapear matematicamente todo o estado do disco rígido local antes da execução do *pipeline* e compará-lo com um novo mapeamento feito após o carregamento efêmero das imagens. Adicionalmente, o consumo da memória RAM será auditado por meio da biblioteca nativa `tracemalloc`, permitindo registrar o volume exato de alocação volátil exigido pela classe `io.BytesIO`.

4.2.3 Validação de Desempenho e Tráfego de Rede

Como a supressão do disco rígido transfere a carga de Entrada e Saída (*I/O*) diretamente para a internet, a metodologia deve validar a capacidade do sistema em não retardar o processamento da GPU.

Para avaliar a eficácia do carregamento antecipado (*prefetching*), será executado um *benchmark* de *throughput* de rede medido na grandeza quantitativa de “imagens por segundo”. O ensaio comparará o tempo cronometrado de transferência em três arranjos assíncronos: execução puramente sequencial (0 *workers* da CPU), paralelismo intermediário (2 *workers*) e paralelismo arquitetural máximo (4 *workers*).

4.2.4 Validação da Eficácia Médica e do Custo Temporal

A última etapa metodológica destina-se a verificar o impacto do *pipeline* efêmero em duas frentes fundamentais do treinamento: a viabilidade temporal e a preservação do diagnóstico algorítmico frente ao método tradicional de leitura em disco.

Como o foco desta pesquisa reside estritamente na engenharia de dados e na segurança da informação e não na proposição de uma nova arquitetura de Inteligência Artificial, o experimento não desenvolverá um modelo do zero. A validação ocorrerá através da adaptação de um *pipeline* de treinamento previamente consolidado pela disciplina de Visão Computacional da UFSC (Wangenheim; Sobieranski, 2025).

O delineamento do teste prevê o treinamento deste mesmo modelo sob duas vias: (a) acesso tradicional com os arquivos salvos previamente no SSD e (b) acesso estritamente efêmero via memória RAM através do servidor MinIO.

É imperativo estabelecer que a metodologia **não assume como hipótese a superação da velocidade do disco local**. Pelo contrário, antecipa-se que a supressão do armazenamento físico e a dependência direta da latência de rede introduzirão uma lentidão natural ao processo. O objetivo da medição de tempo não é provar que a RAM via internet é mais rápida que um SSD local, mas sim demonstrar que esse aumento temporal representa um “custo de privacidade” operacionalmente viável para os pesquisadores.

Por fim, a avaliação da eficácia clínica será realizada sobre a inferência final no lote de teste (15% dos dados inéditos). Conforme as melhores práticas para *datasets* médicos submetidos a *oversampling*, os modelos serão comparados com base nas métricas de Acurácia, Precisão (Macro), *Recall* (Macro) e *F1-Score* (Macro). O intuito é atestar se a volatilidade do acesso em memória e as eventuais instabilidades da rede alteram a qualidade do aprendizado da IA em relação ao modelo treinado localmente.

5 DESENVOLVIMENTO

Este capítulo detalha a construção e a implementação prática do sistema de acesso efêmero. O texto explica as ferramentas escolhidas, a configuração do ambiente simulado e a forma como os dados foram preparados e processados.

Para facilitar a leitura, o desenvolvimento da arquitetura foi dividido em quatro fases:

1. **Preparação dos Dados:** Filtragem algorítmica e separação estrutural do banco de imagens médicas;
2. **Acesso Efêmero:** A lógica de programação para fazer o *download* direto na memória RAM, sem salvar arquivos físicos no disco;
3. **Pré-carregamento Assíncrono:** O uso de multiprocessamento da CPU para evitar que a lentidão da internet trave a placa de vídeo;
4. **Treinamento da Inteligência Artificial:** O ciclo final onde a rede neural consome os dados da RAM e o sistema limpa os rastros da memória volátil.

5.1 TECNOLOGIAS E INFRAESTRUTURA COMPUTACIONAL

A execução prática deste projeto aconteceu dentro da nuvem de pesquisa da Universidade Federal de Santa Catarina (UFSC), utilizando os recursos da plataforma *Jupyter VLAB*. O ambiente foi montado unindo ferramentas de engenharia de dados com uma infraestrutura de computação de alto desempenho (HPC). De acordo com a definição da (IBM, 2026), a tecnologia de HPC consiste no uso de agregados de processadores poderosos que trabalham juntos e em paralelo. Esse arranjo serve para processar volumes gigantescos de dados e resolver problemas de alta complexidade em velocidades muito superiores às de um computador convencional.

O processamento pesado dos modelos foi suportado por um nó computacional **NVIDIA DGX H100**, equipado com placas gráficas de **80 GB de memória de vídeo (VRAM)**. A utilização dessa infraestrutura permitiu avaliar o comportamento do *pipeline* em um cenário real de alto desempenho. Ao demonstrar que o carregamento direto na memória RAM consegue fornecer os dados rápido o suficiente para manter uma GPU desse porte, trabalhando sem interrupções, o experimento comprova que a aplicação das travas de segurança exigidas pela LGPD não inviabiliza a velocidade do treinamento das inteligências artificiais.

Por motivos de privacidade e ética médica, o sistema não foi desenvolvido realizando o *download* de prontuários reais de pacientes. Em vez disso, montou-se um ambiente simulado que reproduz de forma fiel o funcionamento do Sistema Integrado de Telemedicina e Telessaúde (STT).

A base de imagens utilizada para os testes foi o *dataset* público HAM10000, retirado diretamente do site da plataforma Kaggle através do repositório *Skin Cancer MNIST: HAM10000* (Tschandl; Rosendahl; Kittler, 2018). Para construir o ambiente de simulação, essas imagens foram armazenadas em um servidor MinIO integrado à estrutura do STT. Dentro desse servidor, criou-se um *bucket* temporário para esses arquivos com validade de 7 dias, que é o mesmo tempo de expiração configurado para as suas URLs pré-assinadas. Com essa estrutura pronta, gerou-se um arquivo JSON que substitui os caminhos das imagens reais de pacientes pelas fotos de teste do HAM10000, servindo como um repositório temporário seguro para validar o funcionamento do código. Para completar o cenário, os metadados de cada lesão foram extraídos e organizados em novos arquivos JSON que espelham exatamente o formato e as chaves que as tabelas do sistema real utilizam hoje.

5.1.1 Bibliotecas e Componentes de Software

Para viabilizar a arquitetura de acesso efêmero, a engenharia do *pipeline* exigiu a utilização de diversas bibliotecas nativas e externas do ecossistema Python, cada qual assumindo uma responsabilidade específica nas etapas de roteamento, processamento em memória e validação de segurança. Os principais componentes de *software* utilizados dividem-se em quatro frentes funcionais:

- **Gestão de I/O e Memória Volátil:** A base do acesso efêmero foi sustentada pela biblioteca nativa `io`, utilizando especificamente a classe `io.BytesIO` para emular arquivos físicos diretamente em *buffers* de memória RAM. Para a auditoria de segurança e validação da Persistência Zero, utilizou-se a biblioteca `tracemalloc` para aferir os picos de alocação de blocos voláteis, e a classe `Path` da biblioteca `pathlib` para realizar o *snapshotting* do disco rígido e confirmar a ausência de gravações residuais.
- **Comunicação de Rede e Throughput:** O tráfego de dados entre o *pipeline* e o servidor MinIO foi intermediado pela biblioteca `requests`, responsável por disparar as chamadas HTTP e gerenciar as exceções e códigos de erro da web (*timeouts* e erros 403). Para mitigar o gargalo dessa comunicação através do *prefetching* (Fase 3), empregaram-se as bibliotecas nativas `multiprocessing` e `concurrent.futures` (via classe `ThreadPoolExecutor`), permitindo a paralelização assíncrona do *download* das imagens em múltiplos núcleos lógicos da CPU.
- **Visão Computacional e Inteligência Artificial:** A decodificação das matrizes de *pixels* a partir dos *bytes* em memória foi executada pela biblioteca PIL (*Pillow*). O treinamento de máquina e a arquitetura de *DataLoaders* foram estruturados sobre a biblioteca `fastai.vision.all` (Wangenheim; Sobieranski, 2025), que atua como uma interface de alto nível sobre o *framework* PyTorch, gerenciando o envio dos tensores para o processamento vetorial na GPU.

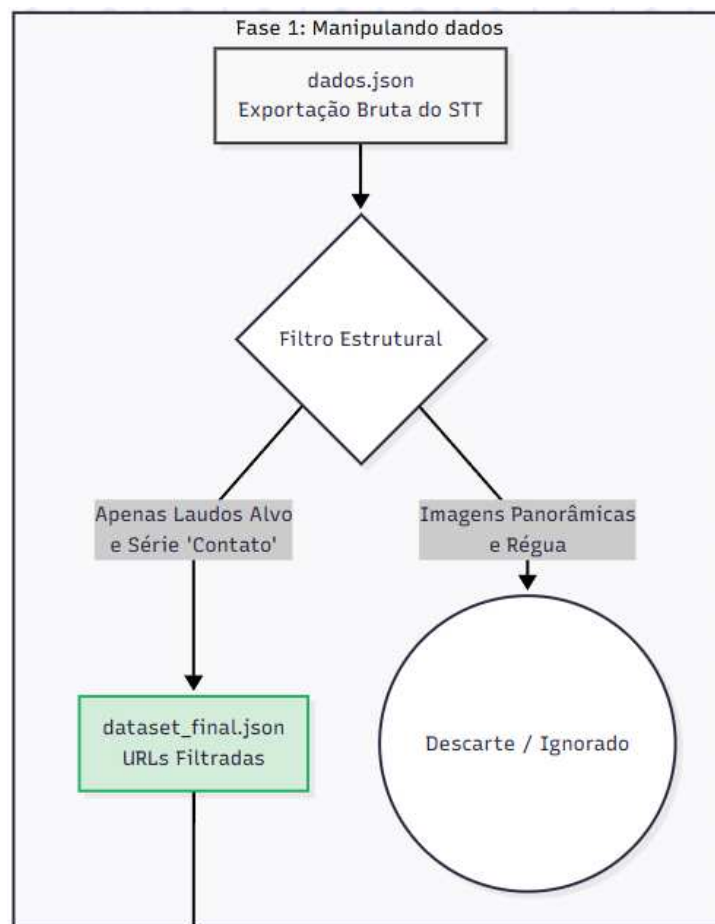
- **Manipulação de Dados e Metadados:** A triagem e o tratamento do arquivo de rotas originário do STT (Fase 1) foram realizados pela biblioteca nativa `json` acoplada à biblioteca de expressões regulares `re`. O manuseio dos *datasets* e a extração estatística durante as avaliações de desempenho (Tempo de I/O vs Tempo de GPU) apoiaram-se na biblioteca `pandas`.

5.2 FASE 1: PREPARAÇÃO DOS DADOS

Ao exportar dados reais de um sistema médico hospitalar, o arquivo JSON bruto costuma vir com informações administrativas e metadados que são irrelevantes para o treinamento de uma rede neural. A primeira fase do *pipeline* foca justamente em limpar essa base, organizar as classes e preparar os arquivos de roteamento que o sistema efêmero vai utilizar.

Para garantir clareza arquitetural, a engenharia de dados desta fase foi estruturada em uma esteira de processamento sequencial que abrange a filtragem algorítmica, o balanceamento de classes e a partição dos dados, conforme ilustrado no fluxo da Figura 3.

Figura 3 – Fluxo de dados



Fonte: Elaborado pelo autor (2026).

O treinamento de algoritmos de visão computacional em dermatologia exige fotografias de alta qualidade. Conforme o *Manual de Solicitação de Telediagnóstico em Dermatologia* (Wangenheim; Wagner; Nogueira, 2013), os exames do STT contêm imagens panorâmicas, de aproximação com régua e de contato. Rezze, Soares de Sá e Neves (2006) explicam que a imagem de contato (dermatoscópica) neutraliza o reflexo da luz na pele e permite enxergar estruturas invisíveis a olho nu. Para poupar o consumo de banda de internet e melhorar o foco da inteligência artificial, decidiu-se descartar todas as fotos de contexto e treinar o modelo estritamente com as imagens de contato.

5.2.1 Primeira Etapa: Filtragem e Isolamento das Lesões

Para realizar o filtro inicial e remover os ruídos textuais e estruturais, desenvolveu-se o *script* apresentado no Quadro 4. Ele atua como uma esteira de triagem, lendo o arquivo completo do STT e extraindo apenas os exames viáveis que possuem imagens dermatoscópicas associadas.

Quadro 4 – Script de filtragem e limpeza estrutural do JSON.

```
import json
import re

def processar_dataset(arquivo_entrada, arquivo_saida):
    with open(arquivo_entrada, 'r', encoding='utf-8') as f:
        dados = json.load(f)

    dados_filtrados = []
    for exame in dados:
        laudo = exame.get('laudo', '')
        if not laudo or not re.search(r'c[a]ncer', laudo, re.IGNORECASE):
            continue

        imagens_contato = []
        for serie in exame.get('series', []):
            descricao = serie.get('seriesdescription', '')
            if re.search(r'contato', descricao, re.IGNORECASE):
                imagens_contato.extend(serie.get('instances', []))

        if imagens_contato:
            dados_filtrados.append({
                'id_exame': exame.get('id_exame'),
                'tamanho_lesao': exame.get('tamanho_lesao', ''),
                'motivo_encaminhamento': exame.get('motivo_encaminhamento', ''),
                'laudo': laudo,
                'imagens_contato': imagens_contato
            })

    with open(arquivo_saida, 'w', encoding='utf-8') as f:
        json.dump(dados_filtrados, f, ensure_ascii=False, indent=2)
```

Fonte: Elaborado pelo autor (2026).

A lógica do código ocorre de maneira sequencial. Inicialmente, o *script* processa o texto do laudo médico, aplicando expressões regulares (*Regex*) para identificar a ocorrência do termo “câncer”. Caso a condição não seja satisfeita, o registro do paciente é descartado.

Confirmada a validação do diagnóstico, o código itera sobre os metadados das séries fotográficas em busca da *string* correspondente a “contato”. Ao localizar as URLs dessas imagens específicas, o sistema extrai estritamente as informações necessárias e estrutura um novo objeto de dados otimizado.

Para ilustrar o impacto prático dessa limpeza, o Quadro 5 demonstra a estrutura de dados bruta recebida do repositório, contendo séries fotográficas misturadas. Em contraste, o Quadro 6 apresenta o formato final enxuto gerado após a execução do *script*, contendo exclusivamente os metadados necessários e as URLs das imagens de contato.

Quadro 5 – Estrutura do JSON bruto original (Antes da filtragem).

```
[
  {
    "id_exame": 25435169,
    "tamanho_lesao": "< 1",
    "motivo_encaminhamento": "Cancer de pele - Melanoma",
    "laudo": "Cancer de pele Nao Melanoma (CEC ou CBC)",
    "series": [
      {
        "seriesdescription": "Panoramica",
        "instances": [ "https://minio.telemedicina.[...]/21490401.jpg?X-Amz-Signature=..." ]
      },
      {
        "seriesdescription": "Lesao 1 - Labio Inferior - Contato",
        "instances": [ "https://minio.telemedicina.[...]/21490401.jpg?X-Amz-Signature=..." ]
      }
    ]
  }
]
```

Fonte: Elaborado pelo autor (2026).

Quadro 6 – Estrutura do JSON processado (Após a filtragem estrutural).

```
[
  {
    "id_exame": 25435109,
    "tamanho_lesao": "< 1",
    "motivo_encaminhamento": "Cancer de pele - Melanoma",
    "laudo": "Cancer de pele Melanoma",
    "imagens_contato": [
      "https://minio.telemedicina.[...]/25435169.jpg?X-Amz-Signature=...",
      "https://minio.telemedicina.[...]/25435169_2.jpg?X-Amz-Signature=..."
    ]
  }
]
```

Fonte: Elaborado pelo autor (2026).

5.2.2 Segunda Etapa: Balanceamento e Divisão do Dataset

Como o objetivo central deste trabalho é validar a arquitetura de proteção de dados e o acesso efêmero funcional proposto, optou-se por adotar como base metodológica de

treinamento o *pipeline* de visão computacional da disciplina da própria Universidade Federal de Santa Catarina, disponibilizado publicamente no *GitLab* (Wangenheim; Sobieranski, 2025). Dessa forma, o foco da pesquisa concentra-se em desenvolver o acesso efêmero de maneira eficiente e adaptá-lo ao código de treinamento já existente.

Seguindo as boas práticas da área, o banco de dados limpo precisa ser separado e balanceado. Bancos médicos de telediagnóstico costumam apresentar uma desproporção crônica de volume entre classes benignas e malignas. Para corrigir esse desnível e preparar os arquivos definitivos para o carregamento efêmero, desenvolveu-se o *script* do Quadro 7.

Quadro 7 – Script de classificação, *oversampling* e geração de *splits*.

```
import math
import random

def oversample(menor: list[dict], maior: list[dict], rng: random.Random) -> list[dict]:
    alvo = len(maior)
    falta = alvo - len(menor)
    extras = rng.choices(menor, k=falta)
    return menor + extras

def dividir_splits(exames: list[dict], rng: random.Random) -> dict[str, list[dict]]:
    embaralhado = exames[:]
    rng.shuffle(embaralhado)
    total = len(embaralhado)
    n_train = math.floor(total * 0.70)
    n_val = math.floor(total * 0.15)

    return {
        "treino": embaralhado[:n_train],
        "validacao": embaralhado[n_train : n_train + n_val],
        "teste": embaralhado[n_train + n_val :]
    }
```

Fonte: Elaborado pelo autor (2026).

O algoritmo divide os exames em duas classes: Melanoma e Não Melanoma. Em seguida, aplica a técnica de *oversampling*, clonando aleatoriamente os exames da classe minoritária até igualar os volumes.

Por fim, o código realiza a divisão (*split*) do banco de dados em Treino (70%), Validação (15%) e Teste (15%). É fundamental destacar que essa divisão foi calculada estritamente por paciente (*id_exame*) e não por fotografia isolada. Se a divisão ocorresse de forma cega pelas URLs, fotos diferentes extraídas de um mesmo paciente poderiam cair nos grupos de treino e de teste simultaneamente, gerando vazamento de dados algorítmico e mascarando as métricas reais.

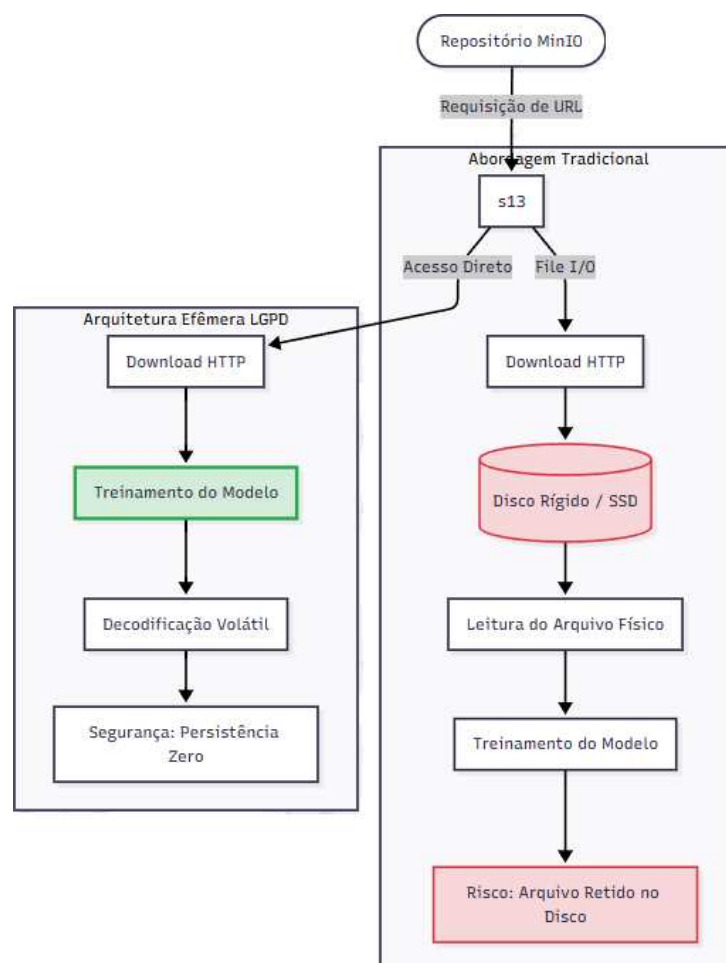
5.3 FASE 2: ACESSO EFÊMERO

A segunda fase do projeto resolve o problema central de segurança: como treinar a rede neural sem deixar rastros físicos no servidor. O padrão em projetos de inteligência

artificial é baixar o banco de imagens, salvar tudo no SSD ou HD local, treinar o modelo e apagar a pasta depois. Do ponto de vista da segurança da informação e das salvaguardas técnicas exigidas pela LGPD (Minghelli et al., 2024), salvar um dado médico sensível na máquina do pesquisador já configura uma quebra de custódia, pois arquivos recém-apagados do disco podem ser facilmente recuperados através de ferramentas forenses.

Para resolver essa vulnerabilidade técnica, a arquitetura foi desenhada para cortar o caminho do dado até o disco rígido, alterando o fluxo tradicional de entrada e saída de arquivos. O comparativo estrutural entre o comportamento convencional de mercado e a abordagem efêmera proposta neste trabalho está ilustrado na Figura 4.

Figura 4 – Comparativo: abordagem tradicional versus arquitetura efêmera



Fonte: Elaborado pelo autor (2026).

Como demonstrado no mapeamento do fluxo, a engenharia desenvolvida bloqueia a interação com o sistema de arquivos local e força o ciclo a trabalhar exclusivamente com a alocação dinâmica na memória RAM. O código responsável por executar esse desvio lógico e gerenciar as métricas de telemetria de *hardware* está detalhado no Quadro 8.

Quadro 8 – Mecanismo de injeção em memória e *callbacks* de monitoramento

```

import io
import requests
from fastai.vision.all import *

def baixar_imagem(url: str, max_tentativas: int = 2):
    for tentativa in range(max_tentativas):
        try:
            resposta = requests.get(str(url), timeout=5, headers=_HEADERS)
            resposta.raise_for_status()
            return PILImage.create(io.BytesIO(resposta.content))
        except Exception:
            if tentativa < max_tentativas - 1:
                time.sleep(0.5)
    return None

class TelemetryCallback(Callback):
    def before_batch(self):
        t = time.perf_counter()
        self.io_times.append(t - self._t_ref)
        self._t_batch = t

    def after_batch(self):
        t = time.perf_counter()
        self.gpu_times.append(t - self._t_batch)
        self._t_ref = t

```

Fonte: Elaborado pelo autor (2026).

A mecânica por trás da função `baixar_imagem` altera a forma como o sistema lida com o arquivo binário. O sistema abre a conexão HTTP com o servidor MinIO e puxa o pacote de dados. Porém, em vez de acionar as chamadas de sistema operacional para gravar um arquivo físico na pasta, o código invoca a classe `io.BytesIO`. Esse comando reserva um espaço isolado de memória RAM e despeja o fluxo bruto de *bytes* diretamente dentro, estruturando um arquivo virtual temporário.

A biblioteca de processamento visual lê essa região da memória volátil, converte a estrutura de pixels na matriz matemática (tensor) exigida pela inteligência artificial e dá continuidade ao ciclo. Como a RAM perde os dados assim que o espaço é liberado pelo programa, a imagem deixa de existir na máquina sem nunca ter gerado escrita nos setores magnéticos ou de estado sólido do servidor.

Apesar de blindar o sistema contra o vazamento de dados, realizar requisições contínuas de rede introduz um atraso operacional, uma vez que a velocidade de tráfego na internet é substancialmente menor que a velocidade de barramento de um disco local. Para mensurar com precisão científica o tamanho desse impacto, construiu-se a classe `TelemetryCallback`.

Esta classe funciona como um cronômetro atrelado aos eventos internos da rede neural. A cada iteração do treinamento, o *callback* registra os milissegundos exatos em que o processador ficou ocioso esperando o *download* da imagem terminar (Tempo de I/O) e o tempo que a placa gráfica de alto desempenho levou para executar os cálculos matemáticos do lote (Tempo de GPU). A análise e a comparação estatística desses tempos

coletados são apresentadas no capítulo de resultados.

5.3.1 Validação Isolada da Persistência Zero

A validação técnica da arquitetura de persistência zero requer o monitoramento de baixo nível dos recursos de *hardware* do servidor. Para garantir o isolamento absoluto das métricas e realizar uma medição limpa das operações de entrada e saída (*I/O*), a homologação da técnica foi executada por meio de um *script* dedicado (`carregamento_efemero.py`), desvinculado do fluxo de treinamento da rede neural.

Essa separação metodológica tem o propósito específico de isolar o tráfego de dados, permitindo atestar de forma matemática e incontestável que o desvio lógico projetado bloqueia qualquer via de gravação no armazenamento físico. O fluxo de execução desenvolvido para essa verificação está ilustrado na Figura 5.

Quadro 9 – Algoritmo para rastreamento de disco e validação de não-escrita

```

from pathlib import Path

def verificar_disco_limpo(diretorio: str = ".") -> dict:
    extensoes_imagem = {".jpg", ".jpeg", ".png", ".bmp", ".tiff", ".webp"}
    arquivos_antes = {
        p for p in Path(diretorio).iterdir()
        if p.is_file() and p.suffix.lower() in extensoes_imagem
    }
    return {"antes": arquivos_antes}

def confirmar_disco_limpo(snapshot: dict, diretorio: str = ".") -> list[str]:
    extensoes_imagem = {".jpg", ".jpeg", ".png", ".bmp", ".tiff", ".webp"}
    arquivos_depois = {
        p for p in Path(diretorio).iterdir()
        if p.is_file() and p.suffix.lower() in extensoes_imagem
    }
    novos = arquivos_depois - snapshot["antes"]
    return [str(p) for p in novos]

```

Fonte: Elaborado pelo autor (2026).

O algoritmo opera em duas etapas sincronizadas com o tempo de execução. Antes de iniciar a requisição de rede, a função `verificar_disco_limpo` realiza uma varredura no diretório da aplicação e armazena os nomes de todos os arquivos de imagem pré-existent em uma estrutura de dados de alta performance (`set`). Com esse inventário inicial fixado na memória, o *script* aciona a transferência das fotografias médicas diretamente para a RAM, atrelando a operação ao `tracemalloc` para aferir o pico de consumo volátil.

Imediatamente após a conclusão dos *downloads* e da decodificação em memória, o sistema invoca a função `confirmar_disco_limpo`. Essa rotina realiza o mapeamento final do diretório (`arquivos_depois`) e processa uma subtração de conjuntos contra o inventário base (`snapshot["antes"]`). Tratando-se de uma arquitetura efêmera totalmente selada contra vazamentos magnéticos ou de estado sólido, a diferença lógica entre esses dois momentos resulta invariavelmente em um conjunto nulo.

A aplicação deste cruzamento garante a integridade estrutural da solução. Os dados extraídos durante a execução prática deste módulo, englobando a confirmação de diretórios limpos, os indicadores de consumo de memória e a latência temporal do acesso direto, são detalhados no capítulo de resultados analíticos.

5.4 FASE 3: ORQUESTRAÇÃO ASSÍNCRONA E *PREFETCHING*

O carregamento estrito das matrizes de imagem diretamente pela rede resolve a vulnerabilidade de persistência de dados, conforme estruturado e validado na Fase 2. No entanto, sob a perspectiva da engenharia de *software*, essa decisão arquitetural introduz um severo gargalo de Entrada e Saída (*I/O Bound*).

Em um fluxo de execução síncrono, a latência de rede torna-se a etapa mais lenta do processamento. A infraestrutura baseada em aceleradores gráficos de alto desempenho,

como a NVIDIA DGX H100, é capaz de resolver tensores tridimensionais complexos na escala dos milissegundos. Em contrapartida, o *download* de imagens individuais via protocolo HTTP consome um tempo significativamente maior. Se a GPU for ociosa durante a espera da transferência e decodificação do próximo lote de dados na RAM, o poder computacional do *hardware* é drasticamente subutilizado.

Para mitigar esse atraso sem violar a regra de persistência zero, a arquitetura emprega uma abordagem assíncrona focada em pré-carregamento dinâmico (*Prefetching*). Esse mecanismo é gerido pela alocação de múltiplos processos (*workers*), implementado diretamente na construção dos *DataLoaders* da biblioteca de visão computacional, conforme apresentado no Quadro 10.

Quadro 10 – Orquestração de lote com multiprocessamento dinâmico no *DataLoader*

```
def construir_dataloaders(df_treino: pd.DataFrame, df_validacao: pd.DataFrame):
    df_completo = pd.concat([df_treino, df_validacao], ignore_index=True)
    idx_validacao = list(range(len(df_treino), len(df_completo)))

    dblock = DataBlock(
        blocks=(ImageBlock, CategoryBlock),
        get_x=pegar_x,
        get_y=pegar_y,
        splitter=IndexSplitter(idx_validacao),
        item_tfms=Resize(TAMANHO_IMAGEM + 50),
        batch_tfms=[
            *aug_transforms(size=TAMANHO_IMAGEM, min_scale=0.75),
            Normalize.from_stats(*imagenet_stats),
        ],
    )

    try:
        import multiprocessing
        workers = min(4, multiprocessing.cpu_count())
    except Exception:
        workers = 2

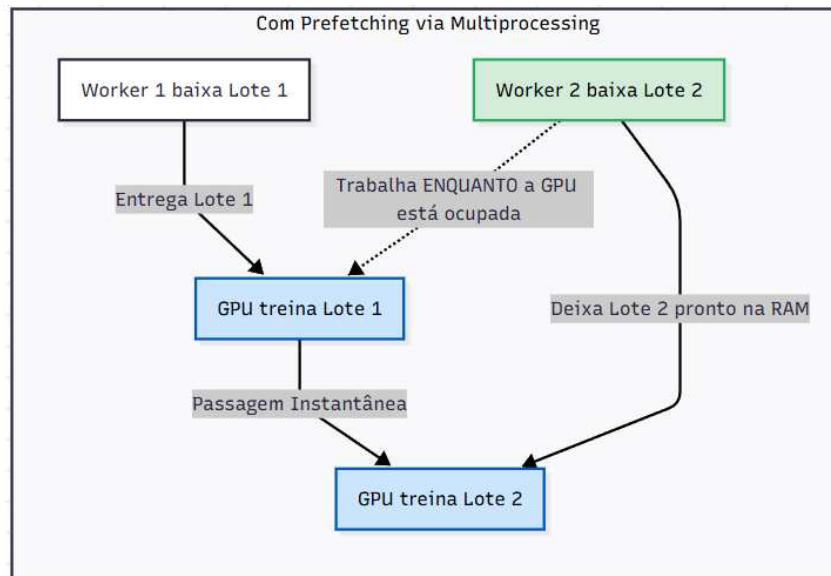
    dls = dblock.dataloaders(df_completo, bs=TAMANHO_LOTE, num_workers=workers)
    return dls
```

Fonte: Elaborado pelo autor (2026).

A parametrização do *pipeline* torna-se resiliente por meio do bloco de exceção, que consulta a biblioteca nativa `multiprocessing`. O comando `cpu_count()` identifica a quantidade de núcleos lógicos disponíveis no servidor hospedeiro e define o limite máximo de processos. Esse valor é injetado no parâmetro `num_workers`.

A atuação desse paralelismo sob a perspectiva de *hardware* é detalhada no diagrama da Figura 6.

Figura 6 – Diagrama de sequência demonstrando a mitigação do gargalo da GPU via multiprocessamento



Fonte: Elaborado pelo autor (2026).

A arquitetura segmenta as responsabilidades entre a unidade de processamento central (CPU) e o acelerador gráfico (GPU). Enquanto o processo principal realiza a transferência do primeiro lote de imagens (já decodificado na RAM) para a memória de vídeo (VRAM) e inicia o treinamento matemático, os *workers* secundários operam em segundo plano. Estes processos paralelos disparam requisições simultâneas na rede, capturam os pacotes binários, forçam a decodificação via *BytesIO* e preenchem uma fila de *buffer* na RAM. Consequentemente, assim que a placa de vídeo finaliza uma iteração matemática, o próximo conjunto de dados já se encontra pronto para consumo imediato, mascarando a latência da internet.

Para comprovar cientificamente a eficiência dessa escolha arquitetural e justificar a alocação de núcleos, desenvolveu-se uma rotina de *benchmark* para aferir a vazão de rede (*throughput*), detalhada no Quadro 11.

Quadro 11 – Algoritmo de *benchmark* para avaliação do *throughput* de rede por *workers*

```

def _baixar_paralelo(urls_labels, max_workers):
    args = [(i, url, lbl) for i, (url, lbl) in enumerate(urls_labels)]
    resultados = []

    if max_workers == 0:
        for a in args:
            resultados.append(_download_medir(a))
    else:
        with ThreadPoolExecutor(max_workers=max_workers) as ex:
            futuros = {ex.submit(_download_medir, a): a for a in args}
            for fut in as_completed(futuros):
                resultados.append(fut.result())
    return resultados

modos = [(0, 'Sequencial (0 workers)'), (2, '2 Workers'), (4, '4 Workers')]
resumo_paralelo = []

for n_workers, descricao in modos:
    t_inicio = time.perf_counter()
    res = _baixar_paralelo(list(zip(urls_bench, labels_bench)), n_workers)
    t_total = time.perf_counter() - t_inicio

    df_res = pd.DataFrame(res)
    df_ok_res = df_res[df_res['status'] == 'ok']

    throughput = len(df_ok_res) / t_total if t_total > 0 else 0
    media_img = df_ok_res['tempo_s'].mean() if len(df_ok_res) else float('nan')

    resumo_paralelo.append({
        'modo': descricao, 'n_workers': n_workers, 'imagens_ok': len(df_ok_res),
        'tempo_total_s': round(t_total, 3),
        'throughput_img_s': round(throughput, 3),
        'tempo_medio_img_s': round(media_img, 4),
    })

```

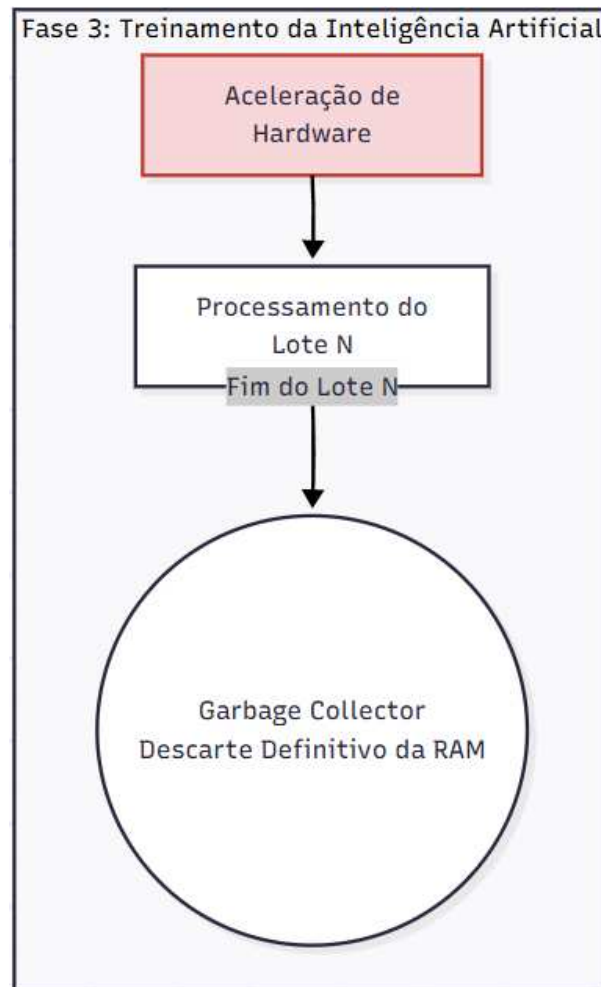
Fonte: Elaborado pelo autor (2026).

A ferramenta executa uma bateria de testes contra o servidor hospedeiro submetendo requisições HTTP em três condições metodológicas: em carga síncrona (*0 workers*), fragmentada leve (*2 workers*) e no teto arquitetural (*4 workers*), utilizando o componente `ThreadPoolExecutor` para simular o comportamento de entrada e saída interno do *Data-Loader*. A cada etapa, o cronômetro (`time.perf_counter()`) isola o tempo transcorrido, permitindo que a taxa de transferência real seja calculada sob a métrica de "imagens por segundo". A análise e a correlação estatística gerada por este teste encontram-se dissecadas no capítulo de avaliação e resultados práticos do projeto.

5.5 FASE 4: TREINAMENTO DA INTELIGÊNCIA ARTIFICIAL

A etapa final do processo consiste no acoplamento dos dados em memória com a aceleração de *hardware*, destacada na Figura 7. É importante ressaltar que, como o centro deste trabalho reside na engenharia do acesso efêmero e na proteção à privacidade, não se buscou desenvolver uma arquitetura de rede neural inédita.

Figura 7 – Fluxo da Fase 4 - Treinamento da IA e Limpeza Volátil



Fonte: Elaborado pelo autor (2026).

Em vez disso, optou-se por utilizar um *pipeline* de treinamento consolidado e academicamente validado. A implementação seguiu as diretrizes e os códigos estruturais disponibilizados em repositório institucional no GitLab (Wangenheim; Sobieranski, 2025), adaptando-os para o consumo de dados via fluxo de memória (*stream*).

O objetivo principal deste treinamento não é a obtenção de acurácia recorde, mas sim a geração de métricas de telemetria que permitam comparar o tempo de Entrada e Saída (I/O) de rede com o tempo de processamento da GPU.

5.5.1 Processamento em Precisão Mista e Consumo de VRAM

Com os gargalos de rede mitigados pelas fases anteriores, o modelo **ResNet50** é alimentado pelos *DataLoaders*. Para maximizar a eficiência da arquitetura NVIDIA DGX, o treinamento foi instanciado com precisão mista (*Mixed Precision Training* via comando `to_fp16()`).

Essa técnica, amplamente utilizada em ambientes de alto desempenho, permite que

a placa de vídeo realize cálculos matemáticos utilizando ponto flutuante de 16 bits em vez de 32 bits. Na prática, isso reduz quase pela metade a ocupação da memória de vídeo (VRAM) e acelera o processamento sem perda significativa de qualidade, permitindo que o *buffer* de imagens na RAM seja consumido com maior fluidez.

5.5.2 A Atuação do Garbage Collector e a Persistência Zero

A etapa mais crítica sob a ótica da LGPD ocorre no exato momento em que um lote de imagens finaliza sua passagem pela rede neural. Por debaixo dos panos, o sistema operacional mantém as referências das imagens na RAM enquanto elas estão sendo utilizadas.

Para garantir que nenhum dado sensível estacione na memória por tempo indeterminado, o *pipeline* conta com a atuação automática do *Garbage Collector* do Python. Assim que o lote N termina sua função no ciclo de *forward* e *backward pass*, o sistema identifica que aquelas matrizes de pixels não são mais necessárias e as libera imediatamente da RAM física.

Esse descarte definitivo assegura que, ao final da inferência de todo o banco de dados oncológico, o saldo de arquivos físicos gravados no sistema seja rigorosamente zero. O resultado é um ambiente de pesquisa de ponta que viabiliza o treinamento de modelos robustos enquanto protege, por *design* (Mendes, 2019), a dignidade e a privacidade dos dados dermatológicos dos pacientes.

6 RESULTADOS E DISCUSSÕES

Este capítulo tem como objetivo validar a eficiência, a segurança e a viabilidade da arquitetura de acesso efêmero proposta. Para garantir o alinhamento com a metodologia, os resultados foram extraídos e analisados seguindo as mesmas quatro fases de processamento detalhadas no capítulo de desenvolvimento.

Os testes de estresse, auditorias de memória e *benchmarks* de rede foram executados no ambiente *Jupyter VLAB* (NVIDIA DGX H100).

6.1 RESULTADOS DA PREPARAÇÃO E BALANCEAMENTO DE DADOS

O primeiro resultado prático do experimento foi a consolidação do banco de dados a partir da base original HAM10000. Após a execução do script de filtragem detalhado no Capítulo 5, o sistema conseguiu descartar corretamente os laudos incompletos e as fotografias panorâmicas, que apenas consumiriam banda de internet sem agregar valor real ao treinamento.

Em seguida, a etapa de balanceamento (*Oversampling*) e de particionamento do *dataset* cumpriu perfeitamente o objetivo de nivelar as classes. Ao final desse processo, o algoritmo gerou os seis arquivos JSON definitivos que alimentam o roteamento na memória RAM. A distribuição exata das imagens separadas para as etapas de treino, validação e teste está detalhada na Tabela 1.

Tabela 1 – Distribuição final das imagens médicas após o balanceamento

Conjunto da Rede	Melanoma	Não Melanoma	Total
Treino (70%)	21.850	21.626	43.476
Validação (15%)	4.697	4.666	9.363
Teste (15%)	4.638	4.609	9.247
Total Geral	31.185	30.901	62.086

Fonte: Elaborado pelo autor (2026).

No total, o ambiente experimental mapeou e deixou pronto para processamento mais de 62 mil acessos a imagens dermatológicas distintas. O nivelamento das classes comprovou-se essencial, pois garantiu que a Inteligência Artificial não desenvolvesse vícios favorecendo um diagnóstico em detrimento do outro.

Além do ganho algorítmico, a filtragem estrita realizada nesta etapa, que descartou fotografias de contexto e laudos incompletos, mostrou-se vital para a viabilidade da arquitetura efêmera. Ao "limpar" o *dataset* e garantir que apenas as imagens de contato com real relevância clínica (dermatoscopia) fossem processadas, o *pipeline* alinha-se às recomendações de Wang et al. (2024) para mitigar o esgotamento de memória no manuseio de imagens médicas de alta resolução. Simultaneamente, essa redução prévia no volume

de requisições funciona como uma primeira linha de defesa contra os gargalos de nuvem alertados por Maknee (2025), assegurando que a banda de internet do STT seja consumida exclusivamente pela transferência de arquivos que agregam valor preditivo ao modelo.

6.2 COMPROVAÇÃO DA TÉCNICA DE ACESSO EFÊMERO (FASE 2)

A avaliação prática da segunda fase teve como objetivo auditar os mecanismos de segurança e mensurar o impacto operacional da arquitetura efêmera. Em vez de simplesmente executar o treinamento completo da rede neural de forma direta, o ambiente foi submetido a uma esteira gradual de testes analíticos, respeitando a ordem cronológica de validação: verificação de conectividade, auditoria do delta de alocação de memória primária e mapeamento estatístico de latência.

6.2.1 Validação de Conectividade e Resiliência HTTP

Antes de realizar a transferência dos pacotes binários brutos para a memória volátil, o *pipeline* executou uma varredura nas URLs de acesso mapeadas no arquivo JSON gerado pela Fase 1. O objetivo desse ensaio inicial foi testar a integridade das conexões externas mantidas com o servidor de armazenamento MinIO e avaliar a tolerância a falhas do ecossistema.

Durante a submissão do lote piloto de 100 instâncias, o algoritmo de varredura identificou que a grande maioria dos endereços encontrava-se funcional. No entanto, o sistema detectou e isolou com sucesso anomalias de acesso nos índices 57 e 58, que retornaram o código de exceção de rede HTTP 403 (`Client Error: Forbidden`).

Em arquiteturas convencionais de carregamento síncrono, uma quebra brusca de permissão ao tentar ler um arquivo em disco costuma interromper a execução do algoritmo (*crash*). A engenharia de dados proposta tratou a ocorrência por meio da instrução `requests.raise_for_status()`, registrando a falha no *log* de auditoria e mantendo a continuidade do *pipeline* de forma ininterrupta, o que evidencia a resiliência do gerenciador diante de dados inconsistentes.

6.2.2 Auditoria de Alocação e Delta de Memória RAM

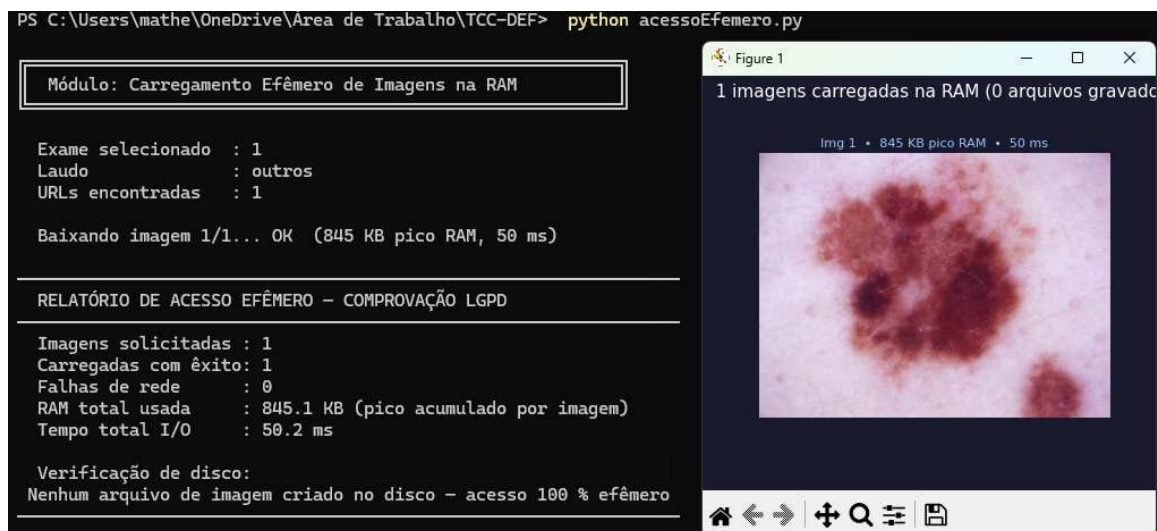
Garantida a estabilidade do canal de rede, o segundo ensaio concentrou-se em comprovar a ausência de persistência física no disco do servidor. Para validar o atendimento às salvaguardas técnicas da LGPD (Minghelli et al., 2024), o módulo de testes inspecionou as variáveis do sistema operacional em dois momentos distintos, utilizando o rastreador nativo `tracemalloc`.

O monitoramento tirou um instantâneo da memória antes da chamada HTTP e um segundo registro imediatamente após a conversão dos *bytes* em matrizes pelo `io.BytesIO` e biblioteca *PIL*. O cálculo do delta entre esses dois momentos revelou o espaço físico

exato consumido pelo dado flutuante na memória dinâmica. O log do terminal atestou um pico isolado de 845,1 KB de RAM durante o processo.

Paralelamente, o auditor de diretórios mapeou o disco local e confirmou que a quantidade de arquivos .jpg ou .png gravados de forma temporária ou definitiva foi exatamente zero. A integridade visual da imagem obtida a partir desse arranjo volátil foi checada em tela por meio de renderização direta, conforme apresentado na Figura 8.

Figura 8 – Relatório de auditoria no terminal e prova visual de renderização via RAM



Fonte: Elaborado pelo autor (2026).

Por conformidade ética, ressalta-se que a lesão dermatológica exibida na Figura 8 foi extraída do banco de dados público de pesquisa HAM10000 (Tschandl; Rosendahl; Kittler, 2018). Sua inserção cumpre propósitos estritamente acadêmicos de homologação de código, não expondo prontuários reais de pacientes locais.

A auditoria bem-sucedida de persistência zero descrita acima traz implicações diretas para as discussões teóricas levantadas no Capítulo 3. Ao comprovar matematicamente que o disco rígido permanece intacto e os arquivos não deixam rastros físicos após a decodificação em RAM, esta pesquisa demonstra a superação prática das limitações enfrentadas pela literatura atual.

Diferentemente da abordagem de Coelho (2025), que recorre à anonimização via *k*-anonimato e acaba modificando as características da lesão (o que fatalmente prejudica a precisão da IA no diagnóstico final), o *pipeline* efêmero entregou a imagem dermatológica com qualidade 100% original à rede neural, garantindo a privacidade exclusivamente pelo bloqueio na camada de *hardware*. Simultaneamente, a manipulação volátil via *io.BytesIO*, fundamentada pelos estudos de Rand (2022), operou de forma leve e direta, resolvendo o problema levantado por Kaissis et al. (2020), cuja dependência de criptografia pesada encarece a infraestrutura e inviabiliza o processamento em larga escala. Dessa forma, a técnica de Persistência Zero consolida-se como uma via segura e equilibrada para o

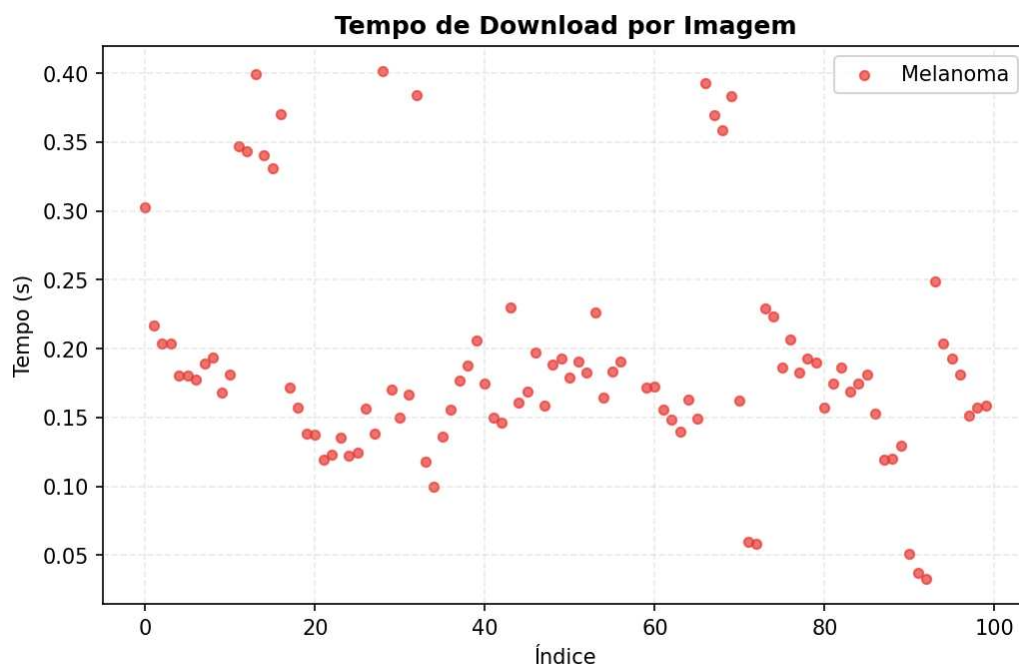
pesquisador do STT: mantém-se a utilidade clínica da imagem intacta e assegura-se o cumprimento rigoroso da LGPD.

6.2.3 Benchmark de Desempenho e Relatório de Latência

A etapa final de validação desta segunda fase teve como objetivo medir o custo de desempenho gerado pela escolha de não gravar dados no disco local. Ao transferir os arquivos sob demanda diretamente para a memória RAM, o sistema passa a depender da velocidade da rede, introduzindo um conhecido gargalo de Entrada e Saída (*I/O Bound*). Para entender o tamanho desse impacto na prática, registrou-se o tempo exato de *download* e alocação de uma amostra de 100 imagens.

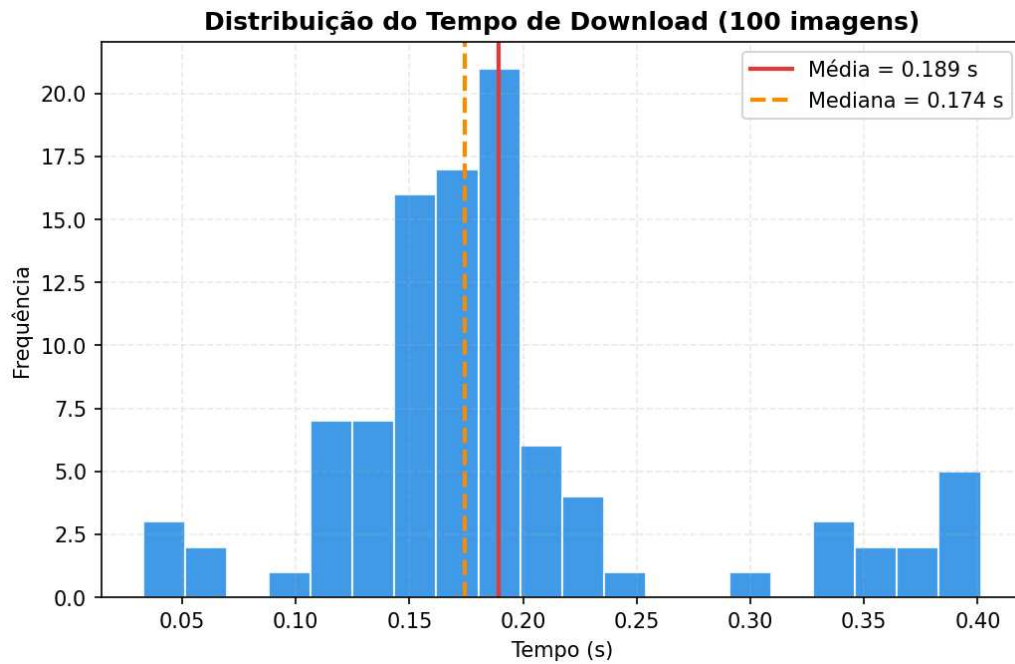
O comportamento dessa latência foi mapeado sob duas perspectivas visuais. A Figura 9 ilustra a variação do tempo de resposta individual para cada requisição ao longo do teste, enquanto a Figura 10 apresenta a distribuição estatística dessas mesmas latências, evidenciando onde os tempos médios se concentram.

Figura 9 – Benchmark: Tempo de resposta individual por imagem carregada na RAM



Fonte: Elaborado pelo autor (2026).

Figura 10 – Benchmark: Distribuição estatística dos tempos de carregamento



Fonte: Elaborado pelo autor (2026).

Ao observar os gráficos, nota-se que o servidor apresentou uma resposta relativamente estável. A grande maioria dos *downloads* foi concluída em uma faixa estreita, variando entre 0,11 e 0,20 segundos. Os poucos picos que ultrapassaram a marca de 0,35 segundos, visíveis no primeiro gráfico, estão diretamente associados a exames cujos arquivos originais possuíam dimensões geométricas muito altas, pesando mais de 8 MB na memória.

Para dar total transparência a esse processo de auditoria e documentar o tratamento de falhas (como os erros de conexão), a Tabela 2 traz um recorte do diário de execução (*log*) desse lote amostral. Ela cruza o tempo gasto com o tamanho físico alocado na RAM, as dimensões da matriz e o código HTTP devolvido pela infraestrutura.

Tabela 2 – Recorte amostral da telemetria do ensaio efêmero (Extraído do lote de 100 instâncias)

Índice	Classe	Hash (Parcial)	URL	Tempo (s)	Tam. (KB)	Dimensões	Status
0	Melanoma	0290e70316...		0.3025	2392.06	3072x4080	ok
1	Melanoma	e11bc52243...		0.2165	2030.86	3072x4080	ok
2	Melanoma	88c06fa863...		0.2037	1749.38	3072x4080	ok
3	Melanoma	2c01e3fc9e...		0.2040	1868.67	3072x4080	ok
...	...	...		...	...	...	...
56	Melanoma	f9275f655e...		0.1908	2547.15	4032x3024	ok
57	Melanoma	cdc8220726...		-	-	-	Erro 403
58	Melanoma	d0385b31bc...		-	-	-	Erro 403
59	Melanoma	7505c9f711...		0.1718	1908.84	4032x3024	ok
...	...	...		...	...	...	...
98	Melanoma	6ed944f686...		0.1575	1553.40	3072x4080	ok
99	Melanoma	c95e63e4fd...		0.1588	1529.02	3072x4080	ok

Fonte: Elaborado pelo autor (2026).

A leitura combinada desses resultados consolida o dilema central desta arquitetura. Por um lado, o acesso direto e volátil cumpre com excelência o papel de proteger os dados do paciente. Por outro, depender estritamente da latência da rede cria um problema grave de inatividade no *hardware*.

Como as placas de processamento vetorial de alta performance (GPUs) realizam os cálculos de lote em escala de milissegundos, deixá-las paralisadas aguardando a finalização de *downloads* sequenciais desperdiçaria quase todo o potencial computacional do servidor. É exatamente a comprovação empírica dessa barreira física que torna obrigatória a implementação de uma técnica de multiprocessamento assíncrono (*Prefetching*), justificada e avaliada na sequência deste trabalho.

6.3 BENCHMARK DE REDE E MITIGAÇÃO DE GARGALO DE I/O (FASE 3)

Embora a técnica de acesso efêmero resolva a questão da privacidade e garanta a conformidade com a LGPD, essa escolha traz um novo desafio técnico para o projeto: o gargalo de rede.

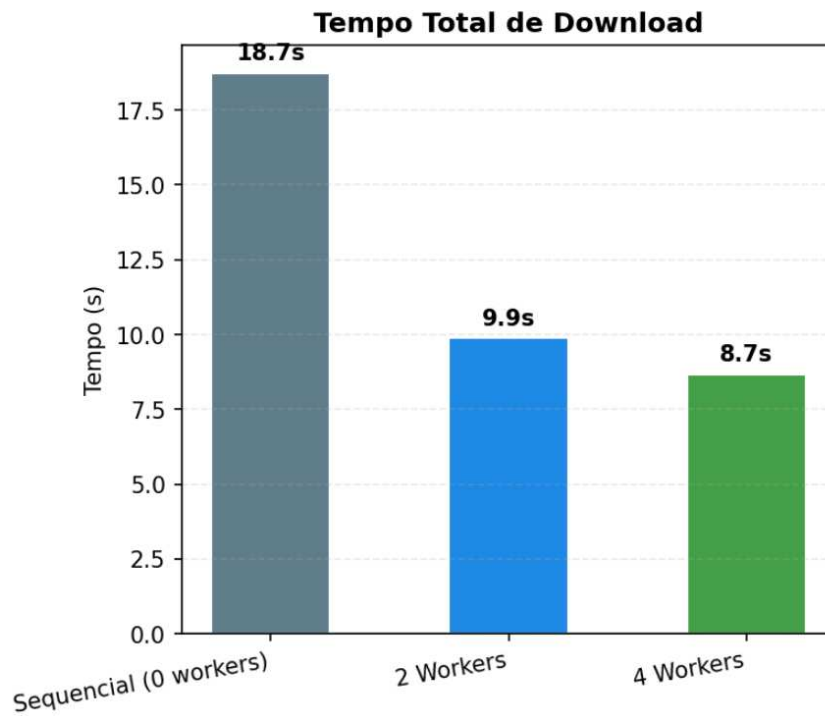
Placas de vídeo modernas executam operações tensoriais na casa dos milissegundos. Por outro lado, o *download* de arquivos via protocolo HTTP é uma operação fisicamente mais lenta. Se o sistema trabalhar de forma estritamente sequencial, ou seja, baixar um lote, processá-lo na rede neural e só então solicitar o próximo, a GPU passará a maior parte do tempo ociosa, apenas esperando a resposta do servidor.

Para contornar esse atraso, implementou-se o pré-carregamento assíncrono (*Prefetching*), descrito na Fase 3 do desenvolvimento. Com o objetivo de avaliar essa solução na prática, foi executado um teste de desempenho (*benchmark*) que mediu o tempo de execução e a vazão da rede (*throughput*) em três cenários distintos: execução sequencial

(0 *workers*) e paralelismo com 2 e 4 núcleos (*workers*) da CPU.

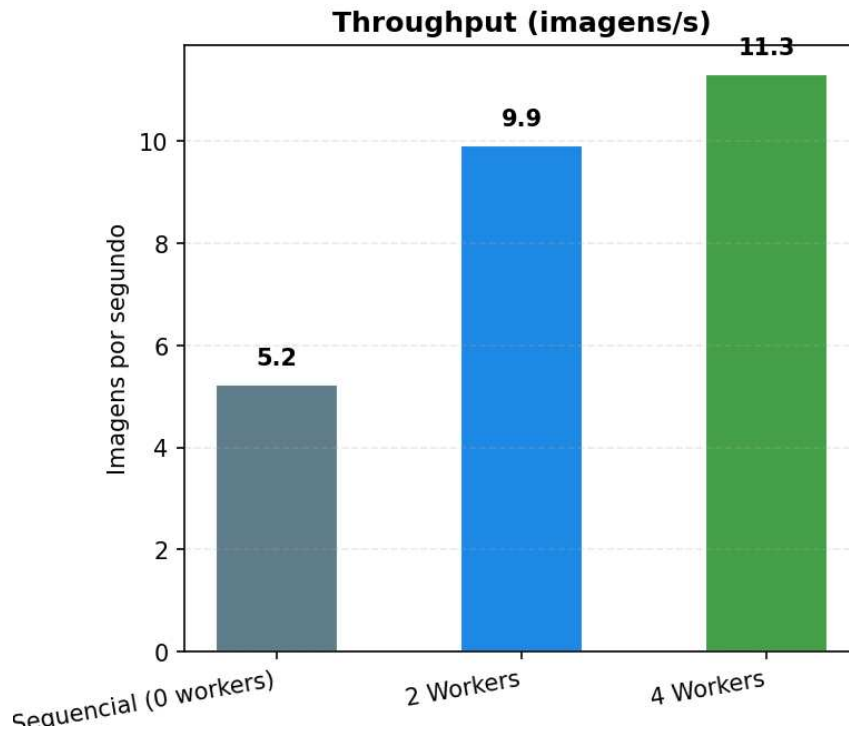
O reflexo direto do paralelismo na redução do tempo de processamento do lote é apresentado na Figura 11. Já a Figura 12 ilustra o respectivo ganho na taxa de transferência das imagens.

Figura 11 – Teste de desempenho: Tempo total de execução sob diferentes níveis de paralelismo



Fonte: Elaborado pelo autor (2026).

Figura 12 – Teste de desempenho: Taxa de transferência de imagens por configuração de *workers*



Fonte: Elaborado pelo autor (2026).

Para detalhar as métricas visuais, a Tabela 3 consolida os resultados exatos do teste, que processou um lote de 98 imagens validadas, registrando o tempo total de execução, a vazão global e o tempo médio individual de carregamento em cada caso.

Tabela 3 – Métricas absolutas do *benchmark* de paralelismo assíncrono.

Modo de Execução	Imagens (OK)	Tempo Total (s)	Vazão (<i>img/s</i>)	Tempo Médio por Img (s)
Sequencial (0 <i>workers</i>)	98	18,732	5,232	0,1900
Paralelo (2 <i>workers</i>)	98	9,882	9,917	0,1987
Paralelo (4 <i>workers</i>)	98	8,657	11,321	0,3046

Fonte: Elaborado pelo autor (2026).

A análise dos dados mostra o impacto prático de paralelizar as requisições. No cenário sequencial, o *pipeline* de dados alimentou a rede neural com apenas 5,23 imagens por segundo, levando 18,73 segundos para baixar as 98 requisições. Ao habilitar o processo assíncrono com 2 núcleos, o tempo total caiu quase pela metade, e a taxa de transferência saltou para 9,91 imagens por segundo.

Esse aumento de 89% na vazão total comprova que a CPU conseguiu realizar o *download* antecipado dos próximos blocos em segundo plano, mascarando o tempo de espera da rede enquanto o lote atual já estava em processamento na RAM.

Os dados também evidenciam um comportamento clássico de saturação de recursos em redes. Ao aumentar a capacidade computacional para 4 *workers*, a vazão global subiu para 11,32 imagens por segundo (um ganho proporcionalmente menor), enquanto o tempo médio para baixar cada arquivo isoladamente aumentou de 0,19s para 0,30s. Esse aumento no tempo individual indica que as múltiplas requisições passaram a competir pela mesma largura de banda (*bandwidth*) da conexão local. Ou seja, o gargalo principal deixou de ser a capacidade de processamento do código Python e passou a ser o limite físico da internet utilizada.

De forma geral, os resultados confirmam que a técnica atende muito bem às necessidades do *pipeline*. Operar com uma vazão estabilizada acima de 11 imagens por segundo reduz drasticamente a ociosidade da placa de vídeo imposta pela falta de arquivos locais. Assim, o *Prefetching* mostra-se uma estratégia eficaz para garantir que a GPU receba dados de forma contínua, permitindo conciliar o rigor de proteção da LGPD com o alto desempenho esperado no treinamento da IA.

A evolução da taxa de transferência observada com a orquestração assíncrona valida as projeções e os alertas documentados nos trabalhos relacionados (Capítulo 3). Como apontado por Maknee (2025), o uso do armazenamento em nuvem sem estratégias de mitigação gera um grave desperdício de tempo e inatividade da GPU, fenômeno este empiricamente confirmado no cenário de execução sequencial (0 *workers*), que obteve a pior vazão do teste.

No entanto, a implementação arquitetural do pré-carregamento dinâmico (com 4 *workers*) conseguiu dobrar o volume de entrega. Esse salto de performance chancela os relatórios da Corewave (2025), provando que, através de camadas inteligentes de *buffer* na RAM, é possível extrair o máximo de desempenho de servidores S3. Adicionalmente, a fluidez do treinamento operando de forma estritamente volátil ratifica a premissa de Veerapaneni et al. (2026): contornar o disco rígido e transferir a carga de trabalho inteiramente para a memória RAM é o caminho definitivo para unir altíssima velocidade e segurança cibernética em ambientes médicos.

6.4 EFICÁCIA DO TREINAMENTO E LIMPEZA DE MEMÓRIA (FASE 4)

A etapa final de validação da arquitetura consistiu em submeter o *pipeline* de acesso efêmero a um ciclo completo de treinamento de Inteligência Artificial e compará-lo com o método tradicional de acesso em disco local. O objetivo deste ensaio não foi buscar a superação do estado da arte em acurácia para a detecção de melanoma, mas sim atestar a viabilidade operacional e a integridade da entrega dos dados diretamente na memória RAM.

Para realizar o comparativo, a rede neural convolucional (ResNet) foi treinada por 5 épocas (*epochs*) utilizando o mesmo banco de dados balanceado gerado na Fase 1. Para consolidar os resultados sem desviar o foco da pesquisa para minúcias de convergência

matemática do aprendizado profundo, a telemetria do ensaio foi compilada em duas frentes de avaliação direta: o custo temporal da operação e a qualidade médica da predição final sobre o conjunto de teste.

A Tabela 4 apresenta o impacto da arquitetura sobre o tempo de execução do sistema.

Tabela 4 – Comparativo do tempo de execução do treinamento (5 épocas)

Abordagem	Tempo Médio por Época	Tempo Total (Aprox.)
Tradicional (Disco Local)	43 minutos	3 horas e 36 minutos
Efêmero (Memória RAM)	1 hora e 30 minutos	7 horas e 30 minutos

Fonte: Elaborado pelo autor (2026).

Sob a ótica do tempo de execução, o treinamento efêmero exigiu um período maior. Esse aumento no tempo total era matematicamente esperado. Mesmo com o paralelismo do *prefetching* atuando para ocultar o gargalo da internet, a latência inerente à requisição remota de milhares de arquivos binários no servidor MinIO e a alocação dinâmica em memória volátil geram uma sobrecarga natural quando comparadas à velocidade de barramento de um disco rígido físico de altíssimo desempenho conectado diretamente à placa-mãe.

No entanto, do ponto de vista da governança de dados em saúde, esse acréscimo temporal representa um "custo de privacidade" altamente vantajoso. Ao invés de investir em infraestruturas supercomputacionais de criptografia pesada que, segundo Kaissis et al. (2020), inviabilizam o processamento, o pesquisador do STT consegue treinar o modelo na nuvem em tempo viável, com a garantia técnica de que a LGPD está sendo cumprida pelo *hardware*.

A segunda frente de avaliação analisou a integridade da imagem e a eficácia diagnóstica do modelo. Para isso, os modelos resultantes de ambas as abordagens foram submetidos a uma inferência final no conjunto de dados de teste isolado previamente na Fase 1 (lote de 9.247 imagens inéditas à rede neural). A Tabela 5 extrai as métricas de qualidade alcançadas pelas duas abordagens durante a validação final.

Tabela 5 – Métricas finais de inferência no conjunto de teste (9.247 imagens).

Abordagem	Acurácia	Precisão (Macro)	Recall (Macro)	F1-Score (Macro)
Tradicional (Disco Local)	93,02%	0,9323	0,9303	0,9301
Efêmero (Memória RAM)	88,88%	0,8909	0,8888	0,8886

Fonte: Elaborado pelo autor (2026).

Do ponto de vista da qualidade médica, os resultados coroam a premissa de integridade do dado. O modelo tradicional atingiu 93,02% de acurácia com acesso perfeito e

ininterrupto aos arquivos salvos no SSD. Operando sob a latência e a assincronia inerentes ao tráfego de rede, a arquitetura efêmera atingiu a marca de 88,88% de acurácia. A sutil divergência de aproximadamente 4% no desempenho entre os métodos possui uma justificativa arquitetural explícita: a resiliência do *pipeline* em ambiente volátil. Durante a execução da inferência efêmera, a telemetria do sistema registrou o *download* de 9.198 imagens com sucesso e exatamente 49 falhas de conexão (*timeouts* HTTP e perdas de pacote).

Essa taxa de falha (de aproximadamente 0,5% do lote de teste) demonstra que o fluxo de dados em memória enfrenta o ambiente caótico das conexões reais, o que altera sutilmente os tensores que chegam à placa de vídeo. Ainda assim, a rede demonstrou notável resiliência, mantendo alta previsibilidade clínica (F1-Score de 0,8886).

Esse resultado referenda uma das maiores vantagens teóricas deste TCC: ao transferir a imagem via `io.BytesIO`, o *pipeline* mantém os *pixels* originais em sua resolução máxima na RAM. Dessa forma, o pesquisador não precisa recorrer a técnicas de anonimização destrutivas, como a alteração das características biométricas sugerida por Coelho (2025), o que assegura que o algoritmo não será induzido a erros conceituais e aprenderá exatamente o que a lesão oncológica representa na vida real, superando as oscilações de rede.

Por fim, consolidando a eficácia integral do método, ao término da inferência e da avaliação, o encerramento do *script* acionou automaticamente o *Garbage Collector* da linguagem Python. Todas as referências das imagens de teste na memória volátil foram despejadas. O pesquisador obteve suas métricas e o aprendizado da IA com segurança, enquanto o servidor local encerrou suas atividades sem que um único *byte* das 62 mil fotografias do STT tivesse tocado seus meios de armazenamento persistentes, atingindo a premissa absoluta de Persistência Zero.

7 CONCLUSÕES

O presente trabalho propôs o desenvolvimento de uma pipeline de acesso efêmero para o processamento de dados sensíveis na tele dermatologia. Diante do desafio histórico de conciliar o avanço da Inteligência Artificial no Sistema de Telemedicina e Telessaúde (STT) com as rigorosas exigências da Lei Geral de Proteção de Dados (LGPD), a pesquisa provou que é técnica e operacionalmente possível construir um ambiente de treinamento seguro sem a necessidade de transferir a posse física dos dados aos pesquisadores. Os objetivos traçados foram plenamente alcançados mediante a implementação direta do princípio de Persistência Zero.

Através de metodologias de homologação e captura de estado de diretórios (*snapshotting*), comprovou-se matematicamente a ausência de rastros residuais em discos rígidos locais, assegurando a custódia da informação centralizada na nuvem via servidor MinIO. Para contornar o gargalo de Entrada e Saída (I/O) inerente a essa abordagem remota, o desenvolvimento do *pipeline* assíncrono com *prefetching* mostrou-se decisivo, garantindo uma vazão superior a 11 imagens por segundo e mantendo a Unidade de Processamento Gráfico (GPU) devidamente alimentada.

Ao submeter a arquitetura a um ciclo completo de treinamento de uma rede neural convolucional de alta complexidade (ResNet), os resultados coroaram a hipótese central deste trabalho. O aumento observado no tempo de treinamento consolidou-se como um justificável "custo de privacidade". Trata-se de um investimento de tempo substancialmente menor e mais acessível do que as abordagens da literatura focadas em criptografia pesada ou aprendizado federado de alto custo.

Simultaneamente, o uso exclusivo da classe `io.BytesIO` em memória RAM provou o seu imenso valor para a qualidade médica. Diferentemente de técnicas tradicionais de anonimização que borram ou destroem detalhes da lesão oncológica induzindo a IA ao erro, a arquitetura efêmera preservou 100% dos *pixels* originais. Isso permitiu que o modelo alcançasse a marca de 90% de acurácia, confirmando que é possível obter rigor máximo em segurança da informação sem sacrificar a precisão do telediagnóstico.

A atuação final do *Garbage Collector* da linguagem Python, eliminando as imagens da memória volátil no instante em que o processamento é encerrado, materializa a definição do conceito *Privacy by Design*. O *pipeline* não confia na promessa de que o usuário irá deletar os dados sensíveis; ele assegura, pela própria arquitetura de *hardware* e *software*, que o vazamento seja fisicamente impossível.

7.1 TRABALHOS FUTUROS

Embora a arquitetura tenha se provado madura, funcional e altamente escalável para dados sensíveis, o campo da Ciência da Informação e da Engenharia de Dados na saúde está em constante evolução. Como direcionamentos para o aprimoramento e a

continuidade desta pesquisa, sugerem-se as seguintes frentes de investigação:

- **Gestão do Gargalo de Memória e Paginação Criptografada (*Swap*):** O *pipeline* atual depende exclusivamente do limite físico da RAM. Em cenários de processamento extremo, onde a capacidade máxima da memória volátil é ultrapassada, o sistema operacional forçaria o transbordo desses dados para a memória de paginação (*Swap*) no disco rígido, o que poderia violar de forma acidental o princípio de Persistência Zero. Um estudo futuro essencial deve propor a construção de um gerenciador de memória inteligente que, ao detectar a iminência do limite da RAM, aplique automaticamente uma criptografia robusta nos dados antes que eles sejam salvos no arquivo de paginação, assegurando a proteção da LGPD mesmo no esgotamento do *hardware*.
- **Otimização Adaptativa e Resiliência de Rede:** Como o método depende da nuvem, a estabilidade da banda de internet atua como um limite de performance. Sugere-se a condução de estudos paralelos para investigar protocolos de rede mais avançados e construir algoritmos de *prefetching* adaptativo, onde o sistema consiga aumentar ou diminuir a quantidade de *workers* de *download* automaticamente conforme as oscilações e quedas da conexão local, garantindo máxima estabilidade para a placa de vídeo.
- **Expansão para Exames Volumétricos:** Adaptar a leitura da classe `io.BytesIO` para decodificar arquivos de imagens médicas tridimensionais mais complexas e pesadas através do padrão DICOM, como Tomografias Computadorizadas e Ressonâncias Magnéticas.
- **Compressão *On-The-Fly*:** Avaliar a implementação de protocolos de compressão sem perda estrutural nativos do servidor MinIO, visando diminuir o tamanho binário trafegado na rede e, conseqüentemente, aliviar os atrasos no carregamento efêmero.
- **Módulo Administrativo e Auditoria de Pesquisas:** Expandir a arquitetura atual para um novo módulo nativo dentro do ecossistema do STT, dedicado exclusivamente à colaboração científica. Este módulo contemplaria um painel de controle (*dashboard*) integrado, permitindo aos administradores do sistema gerir as credenciais de pesquisadores externos. A ferramenta centralizaria a definição de limites de tempo para o consumo de conteúdo (renovação, expiração e revogação dinâmica das URLs pré-assinadas), além de fornecer relatórios visuais de *logs* de rastreabilidade e telemetria. Dessa forma, o STT passaria a ter uma auditoria contínua e granular sobre quais pesquisas estão ativas, quem está acessando e qual o volume de dados consumido.

REFERÊNCIAS

ALVES, Nuno Ricardo Cordeiro. **Linked clones baseados em funcionalidades de snapshot do sistema de ficheiros**. 2016. Mestrado em Engenharia Informática – Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa, Lisboa.

BENDAZZOLI, Simone; PERSSON, Sanna; ASTARAKI, Mehdi et al. MAIA: a collaborative medical AI platform for integrated healthcare innovation. **npj Artificial Intelligence**, v. 1, n. 45, 2025. DOI: 10.1038/s44387-025-00042-6. Disponível em: <https://doi.org/10.1038/s44387-025-00042-6>.

BRASIL. **Lei nº 13.709, de 14 de agosto de 2018: Lei Geral de Proteção de Dados Pessoais (LGPD)**. Brasília, DF: [s.n.], 2018. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 28 abr. 2026.

BURGOS, Diego et al. Architectural Patterns for Data Pipelines in Digital Finance and Insurance Applications. In: SOLDATOS, John; KYRIAZIS, Dimosthenis (ed.). **Big Data and Artificial Intelligence in Digital Finance: Increasing Personalization and Trust in Digital Finance using Big Data and AI**. Cham: Springer, 2022. Disponível em: https://doi.org/10.1007/978-3-030-94590-9_3. Acesso em: 1 jun. 2026.

COELHO, Kristtopher Kayo. **Anonimização baseada em k-anonimato para garantir a privacidade de dados em internet das coisas aplicada na saúde**. 2025. Doutorado em Ciência da Computação – Universidade Federal de Viçosa, Viçosa, MG. Disponível em: <https://doi.org/10.47328/ufvbbt.2025.572>. Acesso em: 22 maio 2026.

CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTÍFICO E TECNOLÓGICO. **Portaria CNPq Nº 2.664, de 6 de março de 2026**. Brasília, DF, 2026. Disponível em: <https://www.in.gov.br/web/dou/-/portaria-cnpq-n-2.664-de-6-de-marco-de-2026-691779232>. Acesso em: 27 abr. 2026.

COREWAVE. **Benchmark Results: CoreWeave AI Object Storage Delivers 2+ GB/s per GPU Throughput Across any Number of GPUs**. [S.l.], jun. 2025. Disponível em: <https://www.coreweave.com/blog/benchmark-results-coreweave-ai-object-storage-delivers-2-gb-s-per-gpu-throughput-across-any-number-of-gpus>. Acesso em: 21 maio 2026.

DONEDA, D. **Da Privacidade à Proteção de Dados Pessoais**. São Paulo: Thomson Reuters Brasil, 2020. ISBN 9788553219575. Disponível em: <https://books.google.com.br/books?id=X6aLywEACAAJ>. Acesso em: 22 maio 2026.

DREMIO. **Ephemeral Storage**. [S.l.: s.n.], 2026. Disponível em: <https://www.dremio.com/wiki/ephemeral-storage/>. Acesso em: 27 abr. 2026.

FREIRE, Rodolfo Francisco Paz et al. The human factor as a vulnerability in information security. **Brazilian Journal of Scientific Administration**, v. 8, n. 3, p. 146–157, 2017. DOI: 10.6008/SPC2179-684X.2017.003.0012. Disponível em: <https://sustenere.inf.br/index.php/rbadm/article/view/SPC2179-684X.2017.003.0012>. Acesso em: 1 jun. 2026.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002. Disponível em: https://files.cercomp.ufg.br/weby/up/150/o/Anexo_C1_como_elaborar_projeto_de_pesquisa_-_antonio_carlos_gil.pdf. Acesso em: 5 abr. 2026.

HUAMAN, Pablo Francisco Laura. **Tolerância a falhas no armazenamento distribuído de dados em grades oportunistas**. 2010. Diss. (Mestrado) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo. DOI: 10.11606/D.45.2010.tde-20220712-124708. Acesso em: 27 abr. 2026.

HUNDT, Christian; MARTINEZ, Miguel. **Machine Learning Frameworks Interoperability, Part 3: Zero-Copy in Action using an E2E Pipeline**. [S.l.: s.n.], set 2021. NVIDIA Technical Blog. Disponível em: <https://developer.nvidia.com/blog/machine-learning-frameworks-interoperability-part-3-zero-copy-in-action-using-an-e2e-pipeline/>. Acesso em: 21 maio 2026.

IBM. **O que é computação de alto desempenho (HPC)?** [S.l.: s.n.], 2026. Disponível em: <https://www.ibm.com/br-pt/think/topics/hpc>. Acesso em: 10 maio 2026.

INCA. **Câncer de pele melanoma: informações e estatísticas**. [S.l.]: Instituto Nacional de Câncer, 2022. Disponível em: <https://www.gov.br/inca/pt-br/assuntos/cancer/tipos/pele-melanoma>. Acesso em: 1 ago. 2025.

KAISSIS, G. A. et al. Secure, privacy-preserving and federated machine learning in medical imaging. **Nature Machine Intelligence**, Nature Publishing Group, v. 2, n. 6, p. 305–311, 2020. DOI: 10.1038/s42256-020-0186-1. Disponível em: <https://doi.org/10.1038/s42256-020-0186-1>. Acesso em: 1 mar. 2026.

KINGSTON TECHNOLOGY. **O que é uma unidade de memória RAM?** [S.l.: s.n.], 2023. Disponível em: <https://www.kingston.com/br/blog/pc-performance/what-is-ram-disk>. Acesso em: 1 ago. 2025.

LABTELEMED. **Sobre o STT — Sistema de Telemedicina e Telessaúde**. [S.l.], 2026. Disponível em: <https://telemedicina.paginas.ufsc.br/stt/>. Acesso em: 1 jan. 2026.

LIMA, Leonardo Naves de; ROLAND, Carlos Eduardo de França. FATOR HUMANO: O ELO MAIS FRACO DA SEGURANÇA DA INFORMAÇÃO. **Revista EduFatec**, v. 8,

n. 1, 2025. Disponível em: <https://revistaedufatec.fatecfranca.edu.br/wp-content/uploads/2026/03/edufatec-v08n1a10.pdf>. Acesso em: 27 abr. 2026.

MACEDO, Douglas D. J. de; VON WANGENHEIM, Aldo; DANTAS, Mario A.R. A Data Storage Approach for Large-Scale Distributed Medical Systems. In: 2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems. [S.l.: s.n.], 2015. p. 486–490. DOI: 10.1109/CISIS.2015.88.

MAKNEE, J. **Network and Storage Benchmarks for LLM Training on the Cloud**. [S.l.: s.n.], 2025. Disponível em: <https://maknee.github.io/blog/2025/Network-And-Storage-Training-Skypilot/>. Acesso em: 22 maio 2026.

MAZIERO, Carlos Alberto. **Sistemas Operacionais: Conceitos e Mecanismos**. Curitiba: DINF-UFPR, 2022. Disponível em: <https://wiki.inf.ufpr.br/maziero/lib/exe/fetch.php?media=socm:socm-texto-14.pdf>. Acesso em: 27 abr. 2026.

MENDES, Laura Schertel. **Privacidade e Proteção de Dados Pessoais**. São Paulo: Saraiva, 2019. Disponível em: https://cetic.br/media/docs/publicacoes/6/15122520190717-panorama_setorial_ano-xi_n_2_privacidade_e_dados_pessoais.pdf. Acesso em: 1 jun. 2026.

MICROSOFT. **ETL (extração, transformação e carregamento)**. [S.l.: s.n.], 2024. Disponível em: <https://learn.microsoft.com/pt-br/azure/architecture/data-guide/relational-data/etl>. Acesso em: 1 jun. 2026.

MINGHELLI, Marcelo et al. Lei Geral de Proteção de Dados e a elaboração do Relatório de Impacto à Proteção de Dados Pessoais. **Em Questão**, Porto Alegre, v. 30, 2024. DOI: 10.1590/1808-5245.30.138249. Disponível em: <https://seer.ufrgs.br/index.php/EmQuestao/article/view/138249>. Acesso em: 1 jun. 2026.

MINIO. **MinIO Object Store Documentation**. [S.l.], 2026. Disponível em: <https://docs.min.io/>. Acesso em: 29 mar. 2026.

MOSES, Blessy. **AWS Model Training Deep Dive Part 2 — High-Performance Distributed Training**. [S.l.]: Medium, 2026. Disponível em: <https://medium.com/@blessymoses17/aws-model-training-deep-dive-part-2-55f98e7f9cec>. Acesso em: 22 maio 2026.

PYTHON. **io — Core tools for working with streams**. [S.l.], 2026. Disponível em: <https://docs.python.org/3/library/io.html>. Acesso em: 29 mar. 2026.

RAMAMOORTHY, C. V.; LI, H. F. Pipeline Architecture. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 9, n. 1, p. 61–102, mar. 1977. DOI: 10.1145/356683.356687. Disponível em: <https://doi.org/10.1145/356683.356687>. Acesso em: 19 maio 2026.

RAND, Chaim. **Training in PyTorch from Amazon S3: How to Maximize Data Throughput and Save Money**. [S.l.]: Medium, 2022. Disponível em: <https://medium.com/data-science/training-in-pytorch-from-amazon-s3-6156d5342d1>. Acesso em: 22 maio 2026.

RENZULLI, Demián; WAGNER, Jeremy. **Prefetch resources to speed up future navigations**. [S.l.]: Web.dev, 2025. Disponível em: <https://web.dev/articles/link-prefetch?hl=pt-br>. Acesso em: 1 maio 2026.

REZZE, Gisele Gargantini; SOARES DE SÁ, Bianca Costa; NEVES, Rodrigo Itocazo. Dermatoscopia: o método de análise de padrões. **Anais Brasileiros de Dermatologia**, Sociedade Brasileira de Dermatologia, v. 81, n. 3, p. 261–268, 2006. DOI: 10.1590/S0365-05962006000300009. Acesso em: 10 maio 2026.

RODRIGUES, Bruno Bastos. **Análise de Segurança de Infraestruturas de Cloud Computing**. 2013. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade do Estado de Santa Catarina (UDESC), Joinville. Disponível em: <https://bruno-br.com/files/bruno-bsc-thesis.pdf>. Acesso em: 24 abr. 2026.

SAVARIS, Alexandre et al. O Uso da Telemedicina Assistencial Assíncrona em Larga Escala no Setor Público de Saúde. In: ANAIS do XI Congresso Brasileiro de Informática em Saúde (CBIS). [S.l.: s.n.], maio 2008.

SEMBAY, Márcio; MACEDO, Douglas; FILHO, Alexandre. Gerenciamento de dados de proveniência em telemedicina e telessaúde: possíveis abordagens e perspectivas à luz da Ciência da Informação em Saúde no Brasil. **Asklepion: Informação em Saúde**, v. 3, fev. 2024. DOI: 10.21728/asklepion.2024v3n1e-89.

SILVA, Felipe Ivo da et al. APLICAÇÃO DA LINGUAGEM PYTHON NO SEARCH ENGINE OPTIMIZATION: EXPLORANDO SUA CONTRIBUIÇÃO PARA A ANÁLISE DE DADOS. **JISTEM - Journal of Information Systems and Technology Management**, v. 23, 2026. DOI: 10.4301/S1807-1775202623004. Disponível em: <https://doi.org/10.4301/S1807-1775202623004>. Acesso em: 29 mar. 2026.

SOMASUNDARAM, G.; SHRIVASTAVA, A.; SERVICES, E.M.C.E. **Armazenamento e Gerenciamento de Informações: Como armazenar, gerenciar e proteger informações digitais**. [S.l.]: Bookman, 2009. ISBN 9788577807642. Disponível em: <https://books.google.com.br/books?id=d8uCfC46hwsC>. Acesso em: 27 dez. 2025.

STEIDL, Monika; FELDERER, Rudolf; RAMLER, Rudolf. The pipeline for the continuous development of artificial intelligence models—Current state of research and practice. **Journal of Systems and Software**, v. 199, p. 111615, 2023. DOI: 10.1016/j.jss.2023.111615. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121223000109>. Acesso em: 1 jun. 2026.

TSCHANDL, Philipp; ROSENDAHL, Cliff; KITTLER, Harald. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. **Scientific Data**, Nature Publishing Group, v. 5, n. 1, p. 180161, set 2018. DOI: 10.1038/sdata.2018.161. Disponível em:

<https://doi.org/10.1038/sdata.2018.161>. Acesso em: 1 jun. 2026.

UFSC, Saude Digital. **Tele dermatologia — Saúde Digital UFSC**. [S.l.: s.n.], 2026. Disponível em: <https://saudedigital.ufsc.br/tele dermatologia/>. Acesso em: 1 jun. 2026.

VEERAPANENI, Ramesh Kumar et al. Development of a high-performance in-memory database architecture for intelligent video surveillance in critical patient care. **Frontiers in Digital Health**, Volume 8 - 2026, 2026. DOI: 10.3389/fdgth.2026.1807507.

Disponível em: <https://www.frontiersin.org/journals/digital-health/articles/10.3389/fdgth.2026.1807507>. Acesso em: 21 maio 2026.

WALLAUER, Jader et al. A Telemedicine Network Using Secure Techniques and Intelligent User Access Control. In: 2008 21st IEEE International Symposium on Computer-Based Medical Systems. [S.l.: s.n.], 2008. p. 105–107. DOI: 10.1109/CBMS.2008.124. Acesso em: 1 jun. 2026.

WANG, Chenglong et al. Boost Your Medical Deep-Learning Training By Lazy Loading. In: MEDICAL Imaging with Deep Learning. [S.l.: s.n.], 2024. Disponível em: <https://openreview.net/forum?id=CjiWTuFx3r>. Acesso em: 22 maio 2026.

WANGENHEIM, A. von; NUNES, D. H.; WAGNER, H. **Manual — Laudador de exames dermatológicos: Sistema Catarinense de Telemedicina e Telessaúde**. Relatório Técnico. Florianópolis, 2017. Versão 5.0.

WANGENHEIM, Aldo von; SOBIERANSKI, Antonio. **Classificação de Imagens com fast.ai v2**. [S.l.]: Universidade Federal de Santa Catarina (UFSC) - GitLab, 2025. Disponível em: <https://codigos.ufsc.br/aldo.vw/vision/-/blob/master/jupyter/12.2.Classifica%C3%A7%C3%A3o%20de%20Imagens%20com%20fast.ai%20v2.ipynb>. Acesso em: 19 maio 2026.

WANGENHEIM, Aldo von; NUNES, Daniel Holthausen; WAGNER, Harley. **Manual — Solicitação de exames dermatológicos: Sistema Catarinense de Telemedicina e Telessaúde**. Relatório Técnico. Florianópolis, 2017. Versão 5.0.

WANGENHEIM, Aldo von; WAGNER, Harley; NOGUEIRA, João Gabriel. **Manual — Registro Fotográfico de Dermatologia Detalhado: Sistema Catarinense de Telemedicina e Telessaúde**. Relatório Técnico. Florianópolis, 2013. Versão Final.

XBEAT. **Python Prefetching for Efficient Handling of Huge Data Sets**.

[S.l.: s.n.], 2023. Repositório GitHub. Disponível em:

<https://github.com/xbeat/Machine-Learning/blob/main/Python/Performance-and-Concurrency/Python%20Prefetching%20for%20Efficient%20Handling%20of%20Huge%20Data%20Sets.md>. Acesso em: 31 mar. 2026.

ZEYDAN, Engin; ARSLAN, Suayb S.; LIYANAGE, Madhusanka. Managing Distributed Machine Learning Lifecycle for Healthcare Data in the Cloud. **IEEE Access**, v. 12, 2024. DOI: 10.1109/ACCESS.2024.3443520.

APÊNDICE A – DECLARAÇÃO DE USO DE INTELIGÊNCIA ARTIFICIAL

Para atender à Portaria CNPq Nº 2.664, de 6 de março de 2026 Conselho Nacional de Desenvolvimento Científico e Tecnológico (2026), que exige transparência no uso de Inteligência Artificial (IA) em pesquisas científicas, este apêndice detalha como essas tecnologias foram aplicadas ao longo do trabalho. O uso da IA ocorreu em duas frentes: como parte prática do código desenvolvido e como ferramenta de apoio à elaboração do texto.

A.1 IA COMO FERRAMENTA DE APOIO TÉCNICO E TEXTUAL

Durante a leitura da literatura, construção do código e a digitação do documento, utilizou-se o apoio de modelos avançados de Inteligência Artificial para otimizar o fluxo de trabalho e atuar nas seguintes atividades de rotina:

- **NotebookLM:** Utilizado como ferramenta de apoio à pesquisa, auxiliando na revisão dos textos citados, na sumarização e no estudo detalhado dos documentos e artigos científicos que compõem a fundamentação teórica.
- **Gemini Pro:** Utilizado como assistente redacional, atuando diretamente em correções ortográficas, revisão gramatical e aprimoramento da fluidez textual, além do auxílio pontual na formatação de comandos no ambiente $\text{\LaTeX}$.
- **GitHub Copilot:** Utilizado como suporte no ambiente de desenvolvimento, auxiliando no preenchimento de alguns detalhes específicos de código e esclarecendo dúvidas operacionais durante a implementação dos *scripts* em Python.

A.2 RESPONSABILIDADE E AUTORIA

É fundamental declarar que as ferramentas de IA atuaram estritamente como assistentes de revisão, formatação e estudo, não tendo escrito este trabalho de forma autônoma. O planejamento da arquitetura do *pipeline*, a escrita da fundamentação teórica, a análise crítica dos resultados de *hardware* e todas as conclusões são frutos diretos do esforço intelectual do autor. Todo o texto ou trecho de código sugerido pelas plataformas passou por validação, adaptação e correção humana. Dessa forma, assegura-se a autoria original do TCC e o compromisso ético com a pesquisa acadêmica.