UNIVERSIDADE ESTADUAL PAULISTA JÚLIO DE MESQUITA FILHO
FACULDADE DE CIÊNCIAS
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Renato Paes da Rosa Gonzaga de Oliveira

Image Poisoning como Mecanismo de Defesa contra o Uso Não Autorizado de Imagens por Inteligência Artificial

BAURU
2025

Renato Paes da Rosa Gonzaga de Oliveira

Image Poisoning como Mecanismo de Defesa contra o Uso Não Autorizado de Imagens por Inteligência Artificial

Monografia de Trabalho de Conclusão de Curso apresentado na Universidade Estadual Paulista ”Júlio de Mesquita Filho” - UNESP como parte dos requisitos para conclusão do Curso de Bacharelado em Sistemas de Informação.

Orientador: Douglas Rodrigues

BAURU

2025

Image Poisoning como Mecanismo de Defesa contra o Uso Não Autorizado de Imagens por Inteligência Artificial

Renato Paes da Rosa Gonzaga de Oliveira

Monografia de Trabalho de Conclusão de Curso apresentado na Universidade Estadual Paulista ”Júlio de Mesquita Filho” - UNESP como parte dos requisitos para conclusão do Curso de Bacharelado em Sistemas de Informação.

Douglas Rodrigues, Dsc.
Orientador

Nota da Banca Examinadora: _____

Banca Examinadora:

Douglas Rodrigues, Dr.
Orientador

José Remo Ferreira Brega, Dr.
Membro

Camila Colombo de Moraes, Dra.
Membro

O48i

Oliveira, Renato Paes da Rosa Gonzaga de

Image poisoning como mecanismo de defesa contra o uso não autorizado de imagens por inteligência artificial. / Renato Oliveira. -- Bauru, 2025

47 p. : fotos

Trabalho de conclusão de curso (Bacharelado - Sistemas de Informação) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru

Orientador: Douglas Rodrigues

1. Inteligência artificial. 2. Aprendizado do computador. 3. Proteção de dados. 4. Processamento de imagens. 5. Algoritmos de computador. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

AGRADECIMENTOS

Aos meus pais, pelo amor incondicional, apoio diário e por nunca medirem esforços para que eu pudesse conquistar meus objetivos.

Aos meus avós, exemplos de dedicação e sabedoria, cujo carinho e incentivo sempre me motivaram a seguir em frente.

À minha namorada, pela paciência, compreensão e por estar ao meu lado nos momentos mais desafiadores deste percurso, oferecendo apoio e motivação constantes.

Ao meu orientador, pela orientação segura, disponibilidade, conselhos fundamentais e contribuição indispensável para o desenvolvimento deste trabalho.

Aos meus colegas de curso, que compartilharam conhecimento, experiências e momentos marcantes ao longo da jornada acadêmica.

A todos que, de alguma forma, contribuíram direta ou indiretamente para a minha formação e para a realização deste trabalho, deixo aqui o meu sincero muito obrigado.

RESUMO

Com o avanço acelerado da Inteligência Artificial (IA), especialmente em modelos de visão computacional, cresce a preocupação com o uso indevido de imagens pessoais, que podem ser capturadas, analisadas ou até modificadas sem o consentimento do usuário. Neste contexto, o presente trabalho propõe o desenvolvimento de uma plataforma web simples que permita ao usuário aplicar técnicas de image poisoning, uma forma de ataque adversarial voltada à proteção de imagens contra modelos de IA. A solução consiste em oferecer algoritmos capazes de introduzir pequenas perturbações imperceptíveis nas imagens, de modo que estas passem a ser interpretadas de forma incorreta por sistemas automatizados, sem comprometer a visualização humana. O objetivo principal é criar uma ferramenta acessível que viabilize a proteção de imagens pessoais contra usos indesejados por inteligências artificiais, contribuindo para a preservação da privacidade digital em ambientes cada vez mais automatizados.

Palavras-chaves: Image Poisoning, Inteligência Artificial, Privacidade Digital, Ataques Adversariais, Visão Computacional, Data Poisoning e Proteção de Imagens.

ABSTRACT

With the rapid advancement of Artificial intelligence (AI), particularly in computer vision models, concerns are growing over the misuse of personal images, which can be captured, analyzed, or even modified without user consent. In this context, this project proposes the development of a simple web platform that enables users to apply image poisoning techniques, a form of adversarial attack aimed at protecting images from AI systems. The solution involves implementing algorithms capable of introducing subtle, imperceptible perturbations to images, causing them to be misinterpreted by automated systems while remaining visually intact to the human eye. The main objective is to create an accessible tool for protecting personal photos against unauthorized use by artificial intelligence, thus contributing to digital privacy preservation in increasingly automated environments.

Keywords: Image Poisoning, Artificial Intelligence, Digital Privacy, Adversarial Attacks, Computer Vision, Data Poisoning and Image Protection

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ABSP	Anuário Brasileiro de Segurança Pública
AI	Artificial intelligence
DE	*Differential Evolution*
IA	Inteligência Artificial
ILSVRC	ImageNet Large Scale Visual Recognition Challenge

SUMÁRIO

1 INTRODUÇÃO

Com o avanço das tecnologias de inteligência artificial, especialmente na área de visão computacional, cresce também a preocupação com a exposição e o uso indevido de imagens pessoais. Hoje, algoritmos são capazes de identificar rostos, classificar objetos e até gerar versões modificadas de fotos reais, muitas vezes sem o consentimento dos proprietários originais. Esse cenário levanta questões urgentes sobre privacidade digital, consentimento e segurança no uso de dados visuais. A motivação deste trabalho parte justamente da necessidade de oferecer às pessoas meios concretos de proteger suas imagens contra sistemas de IA que podem explorá-las para fins não autorizados.

O impacto dessa realidade já é perceptível em diferentes esferas sociais. De acordo com dados do Anuário Brasileiro de Segurança Pública (ABSP) de 2025, apenas no último ano foram registrados mais de 165 mil casos de estelionato no país, um aumento de 7,8% em relação à 2023, com um aumento de 17% nos casos de estelionato eletrônico (FBSP, 2025). Estados como Santa Catarina (64.230), Minas Gerais (35.749), Distrito Federal (15.580) e Espirito Santo (15.277) lideram em ocorrências de golpes de estelionato por meio eletrônico, golpes deste tipo apoiados por inteligencia artificial se tornam ainda mais perigosos e convincentes. Tecnologias de clonagem de voz e rosto, conhecidas como *deepfakes*, têm sido utilizadas para simular identidades e aplicar golpes em tempo real, inclusive em chamadas de vídeo. Essas técnicas, antes restritas a especialistas, tornaram-se acessíveis até mesmo a usuários com pouca experiência técnica, o que amplia ainda mais os riscos (Machado, 2024).

Como proposta de solução, este trabalho visa desenvolver uma plataforma web simples que permita a aplicação de técnicas de *image poisoning*, ou envenenamento de dados visuais, prática que consiste em inserir pequenas perturbações nas imagens para influenciar ou distorcer o comportamento de modelos de IA. Essa abordagem utiliza ataques adversariais, isto é, algoritmos capazes de gerar modificações sutis projetadas especificamente para induzir erros nos modelos de *machine learning*. A plataforma permitirá que os usuários carreguem suas imagens pessoais e, com o uso desses algoritmos, apliquem perturbações discretas que tornam as imagens menos legíveis ou inutilizáveis para sistemas de IA, sem comprometer sua aparência para o olho humano. Dessa forma, a plataforma atua como uma barreira de proteção contra o uso indevido de imagens em ambientes digitais.

1.1 Objetivo

Objetivo Geral

O objetivo geral deste trabalho é desenvolver uma plataforma web que permita ao usuário aplicar técnicas de ataques adversariais em imagens pessoais, realizando o envenenamento (*Image Poisoning*) dessas imagens com o intuito de dificultar sua utilização por modelos de inteligência artificial não autorizados.

Objetivos Específicos

- Implementar uma interface web intuitiva para upload, seleção de técnica e download de imagens protegidas.
- Avaliar a eficácia das imagens envenenadas frente a modelos de visão computacional (testes controlados com classificadores padrão).
- Garantir que as alterações aplicadas mantenham a perceptibilidade visual para humanos dentro de limites aceitáveis (critério de imperceptibilidade).
- Documentar o processo de implementação e os resultados obtidos, oferecendo recomendações para extensões futuras (novas técnicas, otimizações e validação em outros modelos).

1.2 Hipótese

Parte-se da hipótese de que a aplicação de técnicas de envenenamento de imagens, por meio de ataques adversariais sutis e quase imperceptíveis ao olho humano, pode reduzir significativamente a capacidade de sistemas de inteligência artificial em interpretar e utilizar essas imagens de forma adequada. Assim, acredita-se que a criação de uma plataforma web acessível, que permita a aplicação prática dessas técnicas, possa representar uma solução viável para proteção de dados visuais pessoais, contribuindo para a preservação da privacidade digital e dificultando o uso não autorizado de conteúdos em treinamentos de modelos de aprendizado de máquina que possam ser utilizados para aplicação de golpes, roubo de identidade e outros crimes do ambiente digital.

1.3 Estrutura

Este trabalho está organizado de forma progressiva para apresentar, fundamentar e validar a proposta desenvolvida. O Capítulo 2 reúne a fundamentação teórica necessária, abordando conceitos essenciais como deepfakes, ataques adversariais e técnicas de envenenamento de imagens. O Capítulo 3 apresenta os principais trabalhos relacionados, descrevendo soluções como Glaze e Nightshade, além de uma análise comparativa entre elas. No Capítulo 4, são detalhados os diagramas de modelagem da plataforma CloakMe, incluindo casos de uso, sequência, classes e a arquitetura lógica que sustenta o sistema. O Capítulo 5 apresenta os materiais e métodos utilizados, descrevendo em profundidade as técnicas adversariais implementadas, os conjuntos de dados, modelos e parâmetros empregados. O Capítulo 6 reúne as tecnologias que compõem a implementação, cobrindo aspectos de *Front-end*, *backend*, processamento de imagens e infraestrutura. O Capítulo 7 descreve os procedimentos de validação, critérios de avaliação e análise dos resultados obtidos, incluindo a comparação entre as diferentes técnicas adversariais. Por fim, o Capítulo 8 apresenta a conclusão, destacando as contribuições alcançadas, limitações observadas e possibilidades para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O avanço da inteligência artificial, especialmente nas áreas de visão computacional e geração de conteúdo sintético, tem transformado profundamente o modo como dados visuais são produzidos, interpretados e utilizados. Essas tecnologias, que antes estavam restritas a ambientes acadêmicos e corporativos, tornaram-se amplamente acessíveis, impulsionando o desenvolvimento de soluções inovadoras em diversos setores, como saúde, segurança, entretenimento e marketing. No entanto, esse mesmo avanço trouxe consigo novos desafios relacionados à privacidade, à ética e à segurança digital.

De acordo com Juliboni (2024), a Gartner, empresa americana de consultoria em tecnologia, estima que até 2027 as ferramentas baseadas em inteligência artificial serão responsáveis por aproximadamente 17% dos crimes cibernéticos. Apenas em 2024 já foi observado um aumento de 11% nas ocorrências globais em relação ao ano anterior, com destaque para a América Latina, onde o crescimento chegou a 72%. Esse cenário revela não apenas o aumento na sofisticação dos ataques, mas também a ampliação do acesso a tecnologias antes restritas, agora utilizadas em esquemas de fraude, manipulação e espionagem digital (Juliboni, 2024).

2.1 Deepfakes

Entre as aplicações mais controversas da inteligência artificial estão os *deepfakes*, termo derivado da junção de “deep learning” e “fakes”. Essa tecnologia utiliza redes neurais profundas e técnicas de aprendizado de máquina para criar imagens, áudios e vídeos falsos de pessoas reais, de forma altamente realista e com grande potencial de engano. O fenômeno teve origem em comunidades online, como o Reddit, onde usuários começaram a compartilhar códigos abertos capazes de sobrepor rostos de celebridades em outros vídeos. Com a disseminação desses algoritmos, o número de falsificações cresceu exponencialmente, e o termo passou a designar qualquer mídia manipulada por IA com fins de simulação de identidade ou desinformação (Kietzmann et al., 2020).

Os *deepfakes* tornaram-se alarmantes devido a dois fatores principais: credibilidade e acessibilidade. A credibilidade está ligada à capacidade dessas mídias de explorar a confiança humana em registros visuais e sonoros. Já a acessibilidade refere-se à popularização das ferramentas de criação, que atualmente permitem que qualquer pessoa, com um smartphone e alguns segundos de vídeo, produza falsificações convin-

centes. Tais conteúdos têm sido usados para difundir desinformação, cometer fraudes, manipular a opinião pública e violar direitos de imagem, o que demonstra o poder e o perigo dessa tecnologia no contexto social e digital contemporâneo (Kietzmann et al., 2020).

2.2 Ataques Adversariais

Os ataques adversariais (*adversarial attacks*) representam uma das principais vulnerabilidades dos modelos de aprendizado de máquina. Esses ataques consistem na introdução de pequenas perturbações nos dados de entrada, quase imperceptíveis ao olho humano, mas capazes de causar erros significativos nas previsões de um modelo. Em termos gerais, os ataques adversariais podem ocorrer em duas fases distintas: durante o treinamento, conhecidos como *poisoning attacks*, ou durante a inferência, conhecidos como *evasion attacks* ou *fooling attacks* (Mohanty; Roudsari; Lashkari, 2022).

As perturbações são cuidadosamente calculadas para enganar o modelo, alterando sua classificação sem comprometer a aparência original dos dados. Em aplicações de visão computacional, isso significa que uma imagem pode parecer idêntica ao olho humano, mas ser interpretada de forma completamente diferente por uma Inteligência Artificial (IA). Tais ataques têm sido amplamente estudados em contextos de cibersegurança, especialmente na detecção de tráfego de rede e análise de dados em ambientes sensíveis, onde pequenas alterações podem permitir a passagem de informações maliciosas sem serem detectadas (Mohanty; Roudsari; Lashkari, 2022).

Além disso, ataques adversariais demonstram que os modelos de aprendizado profundo, apesar de sua alta precisão, podem ser frágeis diante de pequenas manipulações de entrada. Essa vulnerabilidade destaca a necessidade de desenvolver mecanismos de defesa e robustez, tornando-se um dos maiores desafios atuais na área de segurança de IA (Mohanty; Roudsari; Lashkari, 2022).

2.3 Envenenamento de Imagens

O envenenamento de imagens é uma categoria específica dos ataques adversariais, que ocorre durante a fase de treinamento dos modelos de IA. Nesse tipo de ataque, o conjunto de dados é intencionalmente modificado com imagens alteradas de maneira imperceptível, mas que distorcem o aprendizado do modelo. O resultado é um sistema treinado com informações contaminadas, que passa a classificar incorretamente novos dados de forma previsível e controlada.

Enquanto nos ataques de evasão as alterações são feitas nas imagens de teste, o envenenamento atua no nível estrutural, alterando o comportamento do modelo a longo prazo. Essa estratégia é especialmente perigosa em contextos de aprendizado supervisionado, onde grandes volumes de imagens são coletados automaticamente da internet para treinar modelos. Caso imagens venenosas sejam incluídas nesses conjuntos, o desempenho e a confiabilidade do sistema podem ser gravemente comprometidos.

Por outro lado, o mesmo princípio pode ser aplicado de forma ética e defensiva como neste trabalho, em que o *image poisoning* é utilizado como uma ferramenta de proteção. Ao adicionar perturbações sutis às imagens originais, o objetivo não é enganar o usuário, mas impedir que modelos de IA as utilizem indevidamente em treinamentos futuros. Assim, o envenenamento de imagens torna-se uma forma de resistência digital e de preservação da privacidade pessoal.

2.4 Descrição detalhada dos métodos de envenenamento

2.4.1 *One-Pixel Attack* com *Differential Evolution*

O *One-Pixel Attack* altera apenas um pixel da imagem para causar a classificação incorreta da imagem. A formulação adotada aqui codifica cada candidato como um vetor de dimensão $d = 5$:

$$\mathbf{x} = (u, v, r, g, b),$$

onde $u \in [0, W-1]$ e $v \in [0, H-1]$ representam coordenadas de pixel (imagens com largura W e altura H) e $r, g, b \in [0, 1]$ representam canais de cor normalizados.

Dado um classificador $f(\cdot)$ que produz logits ou probabilidades, adotamos uma função objetivo do tipo margem:

$$J(\mathbf{x}) = p_{orig}(\mathbf{x}) - \max_{c \neq orig} p_c(\mathbf{x}),$$

onde $p_c(\cdot)$ é a probabilidade da classe c após aplicar a perturbação correspondente a $\mathbf{x}$, e $orig$ é a classe prevista para a imagem original. O objetivo do ataque é *minimizar* $J(\mathbf{x})$; se $J(\mathbf{x}) < 0$ então a predição final difere da original (sucesso de envenenamento).

Em vez de busca aleatória simples, utilizamos o algoritmo de *Differential Evolution* (DE), introduzido por Jiawei Su, Danilo Vasconcellos Vargas e Sakurai Kouichi em *"One pixel attack for fooling deep neural networks"*, para otimização global sobre o espaço contínuo das variáveis.

A DE explora população de candidatos e opera por combinação (mutação por diferença) e seleção, sendo adequada para espaços multimodais como o codificado por posição e cor do pixel. Após a convergência aplicamos um passo de *polish* (refinamento local) para melhorar a solução final.

- Representação: (u, v) são arredondados para índices inteiros na avaliação; (r, g, b) são mantidos contínuos em $[0, 1]$.
- Avaliação do candidato: gera-se uma cópia da imagem original com o pixel (v, u) substituído por (r, g, b), prepara-se o tensor, aplica-se normalização e obtêm-se as probabilidades do modelo para calcular $J(\mathbf{x})$.
- Critério de parada precoce: se algum candidato produzir mudança da classe prevista (ou, no caso alvo, atingir a probabilidade alvo), interrompe-se a busca e retorna-se a solução.

A seguir, apresenta-se o pseudocódigo que resume o fluxo geral do processo de otimização utilizado pelo algoritmo de Evolução Diferencial no contexto do One Pixel Attack, destacando como a população é iterativamente refinada até que uma solução eficaz seja encontrada:

1. Inicializar população de DE com dimensões $popsize \times d$.
2. Repetir por G gerações (máximo):
   a) Criar mutantes e realizar recombinação.
   b) Avaliar $J(\cdot)$ para cada indivíduo (aplicar imagem, normalizar, inferir).
   c) Se existir indivíduo com $J < 0$ (ou critério de alvo), retornar solução.
3. Aplicar *polishing* à melhor solução e retornar candidato final.

Para ilustrar visualmente o impacto mínimo porém eficaz do método, a Figura 1 apresenta uma comparação direta entre a imagem original e sua versão envenenada pelo *One-Pixel Attack*. Mesmo alterando apenas um único pixel — uma modificação praticamente imperceptível ao observador humano — observa-se que essa perturbação é suficiente para afetar a confiança da análise do modelo significativamente, evidenciando a eficácia do ataque na manipulação do classificador:

Figura 1 – Exemplo de Envenenamento do *One Pixel Attack*

Fonte: Autor, 2025

2.4.2 *Projected Gradient Descent* - PGD

O ataque PGD é uma generalização iterativa do *Fast Gradient Sign Method* (FGSM) que aplica passos do tipo gradiente (sinal do gradiente) seguidos de projeção para manter a perturbação dentro de uma norma especificada. (Madry et al., 2019). Trabalhamos com restrição ℓ_∞ (a mais comum em estudos práticos). A atualização típica é:

$$x^{(t+1)} = \Pi_{B_\infty(x,\varepsilon)}\left(x^{(t)} + \alpha \cdot \operatorname{sign}(\nabla_x \mathscr{L}(x^{(t)}, y))\right),$$

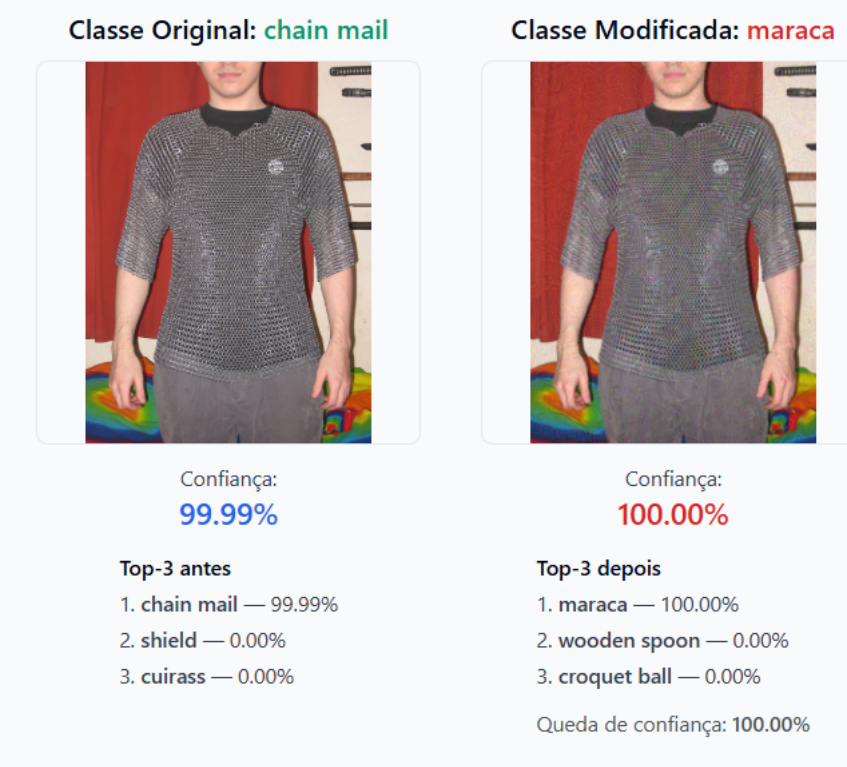
onde:

- $x^{(0)}$ é a imagem inicial (opcionalmente com um início aleatório: ruído uniforme em $[-\varepsilon, \varepsilon]$);
- $\mathscr{L}$ é a função de perda (por exemplo, *cross-entropy* entre predição e rótulo verdadeiro y);
- α é o passo por iteração;
- $\Pi_{B_\infty(x,\varepsilon)}$ é a projeção ponto a ponto que aplica *clipping* para garantir $\|x' - x\|_\infty \leq \varepsilon$.

A escolha da ℓ_∞ garante que nenhum pixel seja alterado além de ε, preservando a aparência visual. O método PGD é determinístico dado $x^{(0)}$ e os parâmetros, e tende a encontrar ataques fortes quando T e ε são suficientemente grandes.

Para evidenciar o efeito sutil, porém direcionado, introduzido pelo ataque PGD, a Figura 2 apresenta a imagem original lado a lado com sua versão adversarial:

Figura 2 – Exemplo de Envenenamento do PGD



Fonte: Autor, 2025

2.4.3 Carlini & Wagner (CW)

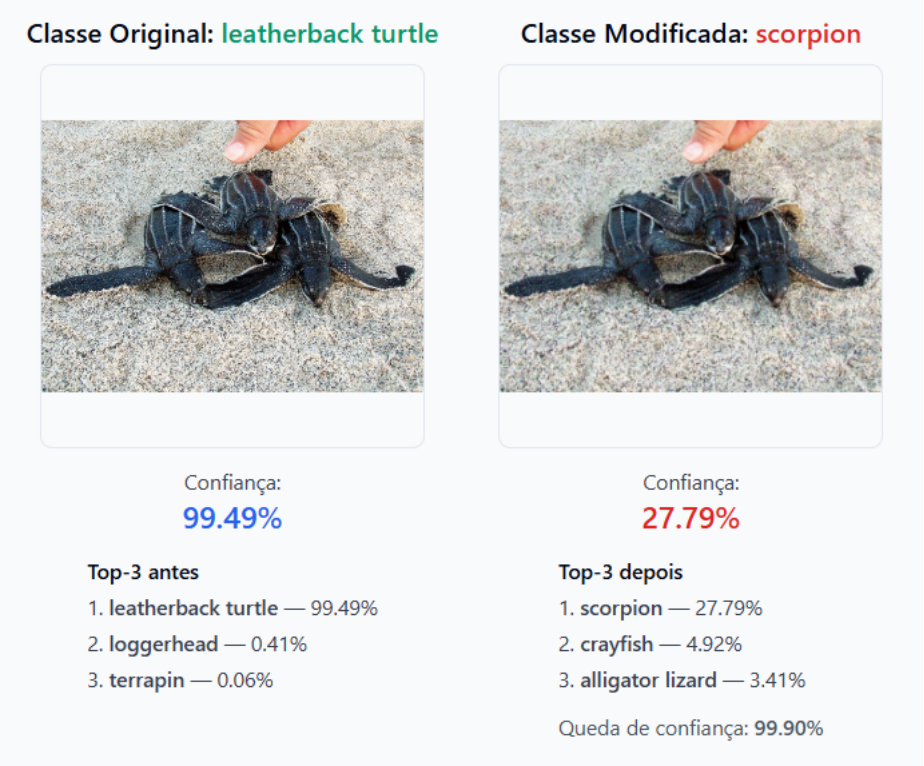
O ataque CW é baseado em formulação de otimização contínua que minimiza a norma da perturbação sujeita à condição de alteração da predição. (Carlini; Wagner, 2016). Uma forma usual (modo não direcionado (*untargeted*) adaptado) é resolver:

$$\min_{\delta} \|\delta\|_p + c \cdot g(x+\delta), \quad sujeito \quad a \quad x+\delta \in [0,1]^n,$$

onde $g(\cdot)$ é uma função que penaliza que a classe correta permaneça com maior logit do que outras (por exemplo $g(z) = \max(\max_{i \neq t} z_i - z_t, -\kappa)$ em versão direcionada ou para variações não direcionadas), $c > 0$ é um parâmetro que equilibra termo de perda e norma, e κ controla margem desejada. A otimização é tipicamente realizada por métodos de primeira ordem sobre uma reparametrização que assegura a restrição box ($[0,1]$) — por exemplo, atuando sobre variáveis ω com $x' = \sigma(\omega)$ ou com *clipping* explícito após cada iteração. (Carlini; Wagner, 2016)

CW tende a produzir perturbações de baixa norma (altamente imperceptíveis) mas exige mais iterações e custo computacional. A Figura 3 ilustra o resultado do ataque Carlini & Wagner ao colocar lado a lado a imagem original e sua contraparte adversarial:

Figura 3 – Exemplo de Envenenamento do CW



Fonte: Autor, 2025

3 TRABALHOS CORRELATOS

Duas soluções atualmente disponíveis no mercado são as plataformas Glaze (Shan et al., 2025) e Nightshade (Shan et al., 2024). O primeiro é voltado à proteção de estilos artísticos, atuando como uma barreira preventiva contra a apropriação indevida por modelos de IA. Já o segundo funciona como uma ferramenta de retaliação, projetada para envenenar conjuntos de dados utilizados no treinamento de inteligências artificiais que fazem uso não autorizado de imagens. As seções 3.1 e 3.2 apresentarão e explicarão, respectivamente, as tecnologias Glaze e Nightshade. Já a seção 3.3 será dedicado à realização de uma análise comparativa entre ambas as soluções

Segundo Shan, S. et al. (2023), o Glaze é uma ferramenta desenvolvida com foco específico na proteção de artistas contra o chamado *style mimicry* ou imitação de estilo, prática em que modelos de texto para imagem aprendem a replicar o estilo visual característico de um artista a partir de um número reduzido de amostras. Essa técnica tem sido amplamente utilizada para treinar modelos que geram imagens “no estilo de” artistas famosos, muitas vezes sem qualquer autorização.

O funcionamento do Glaze baseia-se na geração de *cloaks*, pequenas perturbações visuais aplicadas às obras antes que sejam publicadas online. Essas perturbações são calculadas de forma a alterar a representação da imagem no espaço de características utilizado pelos modelos de IA, sem modificar significativamente sua aparência visual. Para isso, o Glaze utiliza modelos de *style transfer*, que transformam a obra original em uma versão com estilo visual diferente (como cubismo ou impressionismo). Em seguida, é realizada uma otimização que calcula a menor perturbação possível para fazer com que a imagem original passe a ser representada de forma semelhante à versão com o novo estilo no espaço interno do modelo. (Shan et al., 2025).

Dessa forma, ao treinar um modelo com imagens ”*cloaked*” (camufladas), a IA aprenderá uma versão distorcida do estilo do artista, tornando ineficaz qualquer tentativa de mimetismo. Glaze tem se mostrado altamente eficaz, com mais de 90% de sucesso em testes com artistas profissionais, mesmo quando apenas parte do portfólio do artista é protegido. (Shan et al., 2025).

Entre suas principais vantagens estão a facilidade de uso, a invisibilidade das alterações e sua eficácia com poucas imagens. No entanto, a ferramenta atua exclusivamente na proteção de estilo artístico e não interfere na geração de conteúdo geral por modelos de IA.

Em contraste com o Glaze, Nightshade tem como objetivo sabotar diretamente a capacidade dos modelos generativos de responder a prompts específicos, como “cachorro”, “carro” ou “pintura futurista” (Shan et al, 2023). A ferramenta explora uma vulnerabilidade pouco explorada: a esparsidade conceitual. Embora os modelos de texto para imagem sejam treinados com bilhões de imagens, a quantidade de exemplos associados a um único conceito costuma ser relativamente pequena. Isso torna possível que pequenas quantidades de imagens venenosas sejam suficientes para dominar o aprendizado de um conceito específico.

Inicialmente, os autores de Nightshade propuseram um ataque simples conhecido como *dirty-label poisoning*, em que imagens de um conceito A (por exemplo, “vaca”) são propositalmente rotuladas com um texto que descreve outro conceito B (por exemplo, “carro”). Essa técnica se mostrou eficaz, mas de fácil detecção por mecanismos automáticos de filtragem.

A inovação do Nightshade está na versão avançada do ataque, que utiliza perturbações furtivas para alterar imagens reais de um conceito, sem modificar sua aparência visual. Nesse processo, imagens naturais de um conceito B (como “cachorro”) são ligeiramente perturbadas para que, no espaço de características do modelo, se assemelhem a imagens de um conceito A (como “gato”). A otimização busca minimizar a distância entre a imagem perturbada e uma imagem gerada de A. O resultado é uma imagem de “cachorro” que parece um cachorro para o olho humano, mas é interpretada como “gato” pelo modelo durante o treinamento. (Shan et al., 2024).

Nightshade provou ser altamente eficaz, sendo capaz de corromper completamente o comportamento de modelos avançados como o Stable Diffusion XL com menos de 100 amostras venenosas. Além disso, os efeitos podem se espalhar para conceitos relacionados (efeito *bleed-through*) e até desestabilizar o modelo como um todo, se múltiplos ataques forem realizados em diferentes prompts. (Shan et al., 2024).

Entre os benefícios do Nightshade estão sua alta eficácia, resistência a defesas automatizadas e capacidade de atingir conceitos amplos. No entanto, sua eficácia exige que as imagens envenenadas sejam incluídas no conjunto de dados usado para treinar os modelos, assim punindo a utilização não autorizada de imagens.

Para facilitar a compreensão das principais diferenças entre as ferramentas Glaze e Nightshade, a Tabela 1 apresenta uma comparação que sintetiza seus objetivos, estratégias de alteração de imagens, vantagens, aplicações e requisitos de funcionamento.

Tabela 1 – Comparação entre as abordagens do Glaze e Nightshade

Aspecto	**Glaze**	**Nightshade**
Objetivo principal	Impedir mimetismo de estilo	Sabotar geração de conceitos específicos
Estratégia	Cloaking (alteração de estilo)	Poisoning (alteração semântica no aprendizado)
Visibilidade das alterações	Praticamente invisível	Praticamente invisível
Aplicação prática	Proteção artística	Defesa de conteúdo contra scraping e IA geral
Requisitos	Pré-processamento manual das imagens	Inserção em datasets de treinamento

Fonte: Autor, 2025.

4 MODELAGEM E ARQUITETURA DA PLATAFORMA CLOAKME

O sistema **CloakMe** é uma plataforma desenvolvida para proteger imagens contra o uso indevido por modelos de IA, por meio da aplicação de técnicas de *image poisoning*. A seguir, são apresentados três diagramas fundamentais para a compreensão da arquitetura e funcionamento do sistema: o Diagrama de Casos de Uso, o Diagrama de Sequência e o Diagrama de Classes.

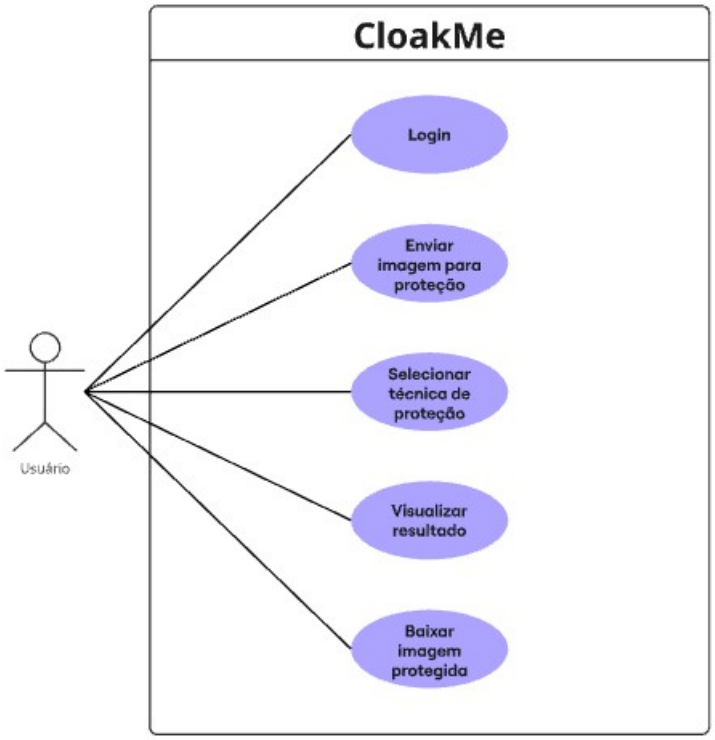
4.1 Diagrama de Casos de Uso

O Diagrama de Casos de Uso ilustra as funcionalidades principais disponíveis ao usuário dentro do sistema CloakMe. Os casos de uso descritos representam as ações essenciais que o sistema oferece, permitindo que o usuário interaja diretamente com a interface web. Os casos incluem:

- **Login:** autenticação do usuário na plataforma.
- **Selecionar técnica de proteção:** o usuário escolhe qual método adversarial deseja aplicar.
- **Enviar imagem para proteção:** o usuário faz o upload da imagem para o método selecionado.
- **Visualizar resultado:** o sistema apresenta a imagem protegida juntamente com informações do ataque.
- **Baixar imagem protegida:** o usuário pode salvar a imagem envenenada em seu dispositivo.

A Figura 4 apresenta a delimitação do escopo funcional do sistema a partir da visão do usuário, evidenciando as ações possíveis e as relações entre os elementos que compõem o caso de uso:

Figura 4 – Diagrama de Casos de Uso



Fonte: Autor, 2025.

4.2 Diagrama de Classes

O Diagrama de Classes representa a estrutura orientada a objetos do sistema, descrevendo os elementos principais que compõem a plataforma CloakMe e suas relações. As classes são organizadas em dois domínios, sendo eles o *Front-end* e *backend*, refletindo a separação entre a interface do usuário e o processamento das técnicas adversariais.

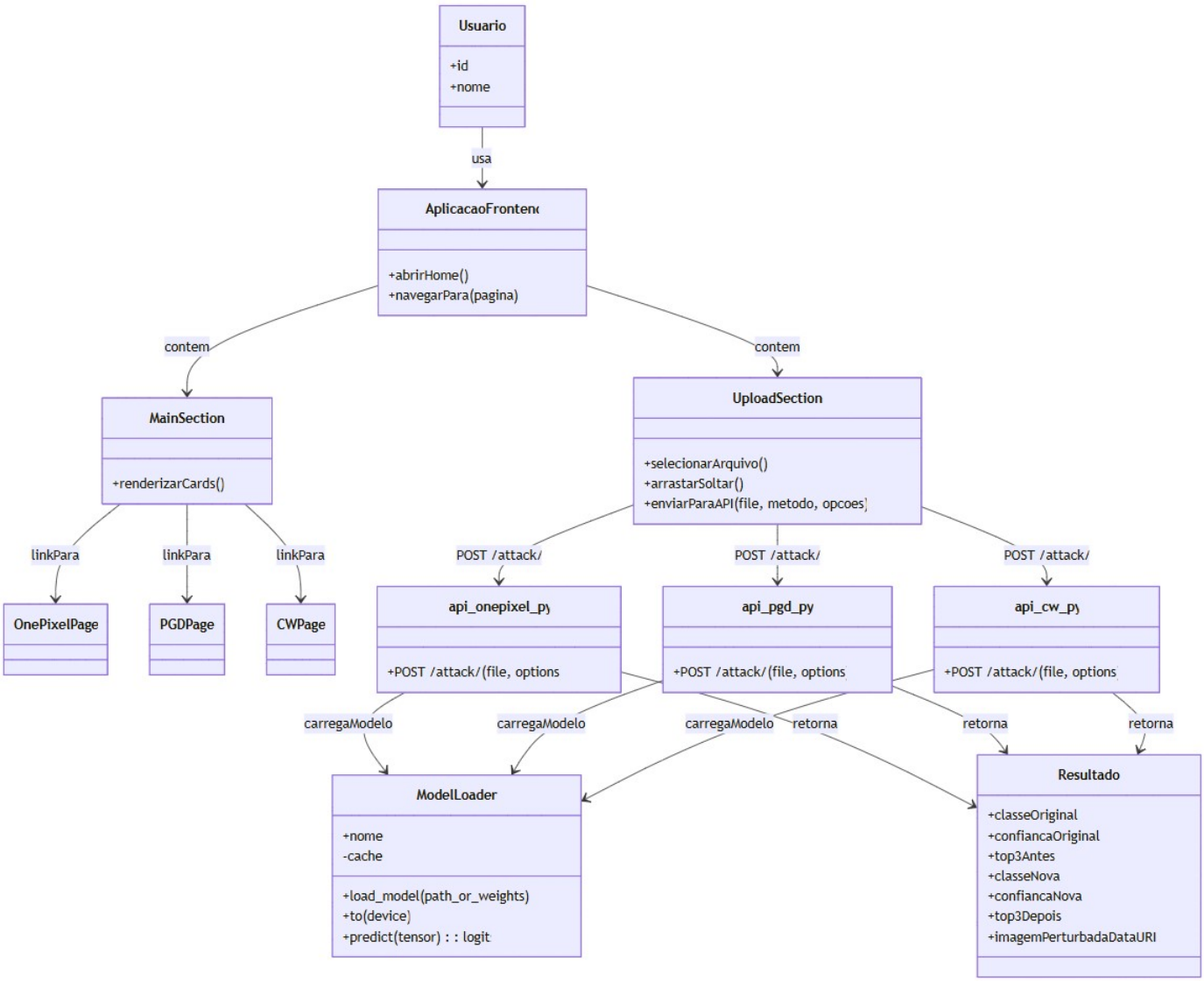
As principais classes são:

- **Usuário:** representa o ator que interage com o sistema, iniciando todas as ações de envio, navegação e seleção dos métodos. É ele quem fornece as imagens e define qual técnica será utilizada. Sua participação dispara todo o fluxo de proteção.
- **AplicacaoFrontend:** centraliza a navegação entre as páginas e organiza a forma como o usuário acessa cada funcionalidade. Atua como o ponto de entrada visual do sistema e coordena a comunicação com os demais componentes do *Front-end*. Garante uma experiência de uso fluida e integrada.

- **MainSection:** seção inicial da aplicação, responsável por apresentar os métodos disponíveis e orientar o usuário para as páginas específicas. Funciona como um painel de escolha que direciona o fluxo principal. Serve também como introdução ao propósito do sistema.
- **UploadSection:** componente que recebe e pré-visualiza a imagem enviada pelo usuário, garantindo que o arquivo seja corretamente encaminhado para o *backend*. Atua como ponte entre o *Front-end* e as APIs. Também valida o envio e prepara a imagem para o processamento.
- **OnePixelPage, PGDPage e CWPage:** páginas dedicadas às três técnicas de proteção, cada uma trazendo uma breve descrição da abordagem e a interface de envio da imagem para a API correspondente. Organizam a experiência conforme o método escolhido. Permitem que o usuário compare e selecione abordagens distintas.
- **ModelLoader:** componente interno das APIs responsável por carregar o modelo ResNet50 utilizado na avaliação das imagens antes e depois da perturbação adversarial. Assegura que o modelo esteja pronto para inferência. É essencial para verificar a mudança de classe após o ataque.
- **API_onepixel, API_pgd e API_cw:** serviços *backend* independentes que executam as respectivas técnicas adversariais de forma isolada. Cada API recebe a imagem, aplica o ataque e retorna o material processado. Juntas, formam o núcleo lógico e computacional da proteção.
- **Resultado:** estrutura que reúne os dados retornados pelas APIs, incluindo informações de classificação, níveis de confiança e a imagem final protegida. Permite que o usuário visualize claramente os efeitos do envenenamento. Serve também como base para análises comparativas dentro do protótipo.

A Figura 5 evidencia a interação entre os módulos e a organização do sistema de forma modular e extensível, permitindo a adição de novos métodos adversariais sem comprometer a arquitetura existente:

Figura 5 – Diagrama de Classes



Fonte: Autor, 2025.

4.3 Diagrama de Sequência

A Figura 6 descreve exclusivamente o fluxo relacionado ao processo de envenenamento da imagem, isto é, a aplicação da técnica adversarial escolhida pelo usuário. Esse diagrama foi reduzido de forma proposital para representar apenas o componente essencial da plataforma: a lógica de proteção da imagem.

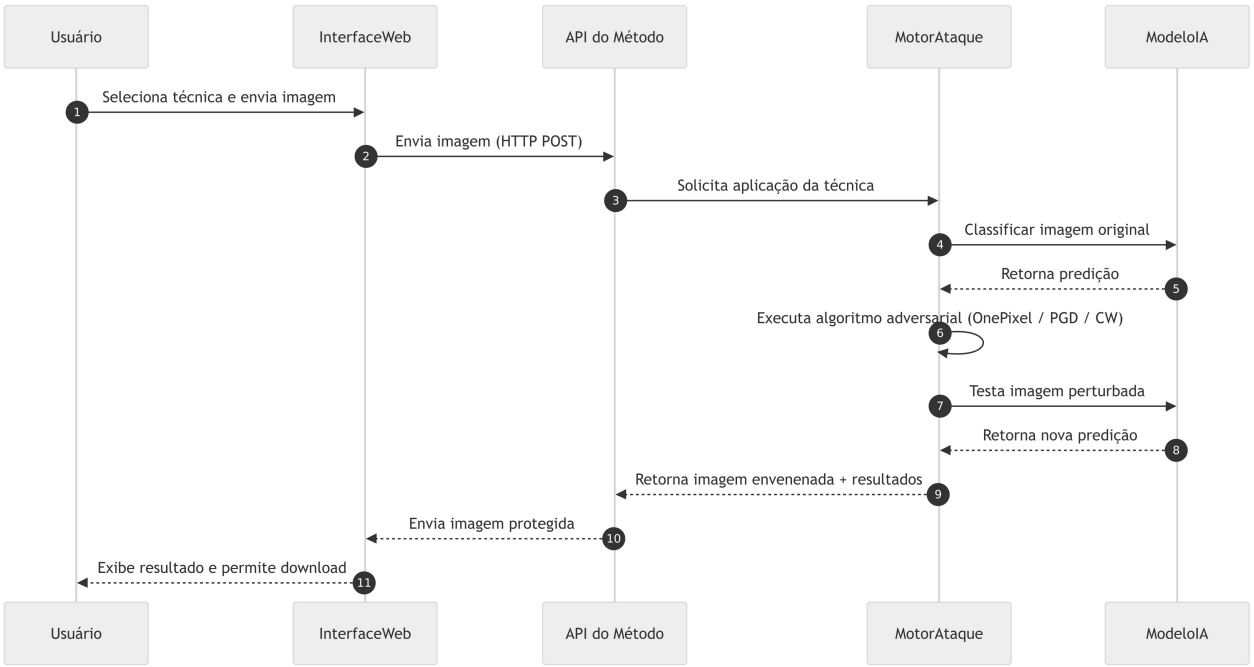
Os principais elementos envolvidos no processo são:

- **Usuário:** inicia o processo ao escolher a técnica de proteção e enviar uma imagem.

- **InterfaceWeb (React):** recebe a imagem selecionada e envia a requisição ao *backend* correspondente ao método escolhido.
- **API do Método:** serviço responsável por processar a imagem, aplicar o ataque adversarial e devolver o resultado.
- **MotorAtaque:** módulo interno que coordena a execução do ataque.
- **Modelo:** rede neural alvo utilizada durante o ataque (ResNet50).

O fluxo representado no diagrama inclui as etapas de envio da imagem, classificação inicial pelo modelo, execução iterativa da técnica adversarial e devolução da imagem envenenada ao usuário, mantendo o foco apenas no processo central de proteção.

Figura 6 – Diagrama de Sequência do Processo de Envenenamento



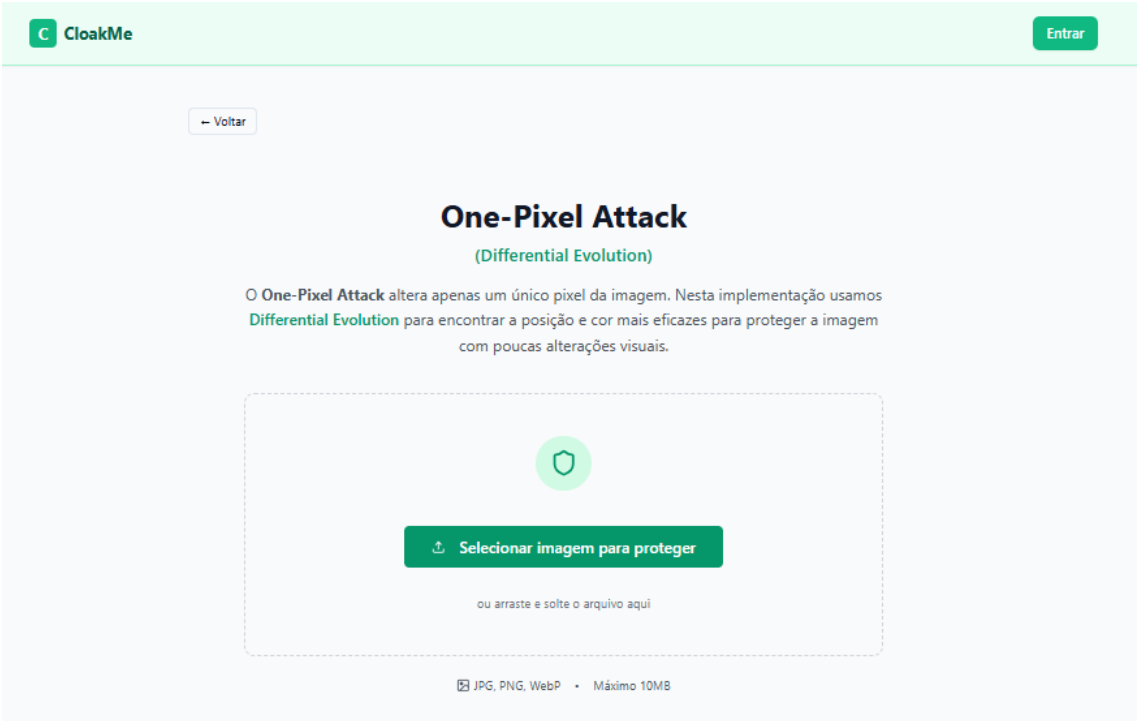
Fonte: Autor, 2025.

Para facilitar a compreensão do diagrama apresentado na Figura 6, as Figuras 7, 8, 9 e 10 ilustram a interface do usuário na plataforma CloakMe, iniciando pela página principal e avançando para as páginas individuais de cada método.

Figura 7 – Página Inicial da Plataforma CloakMe

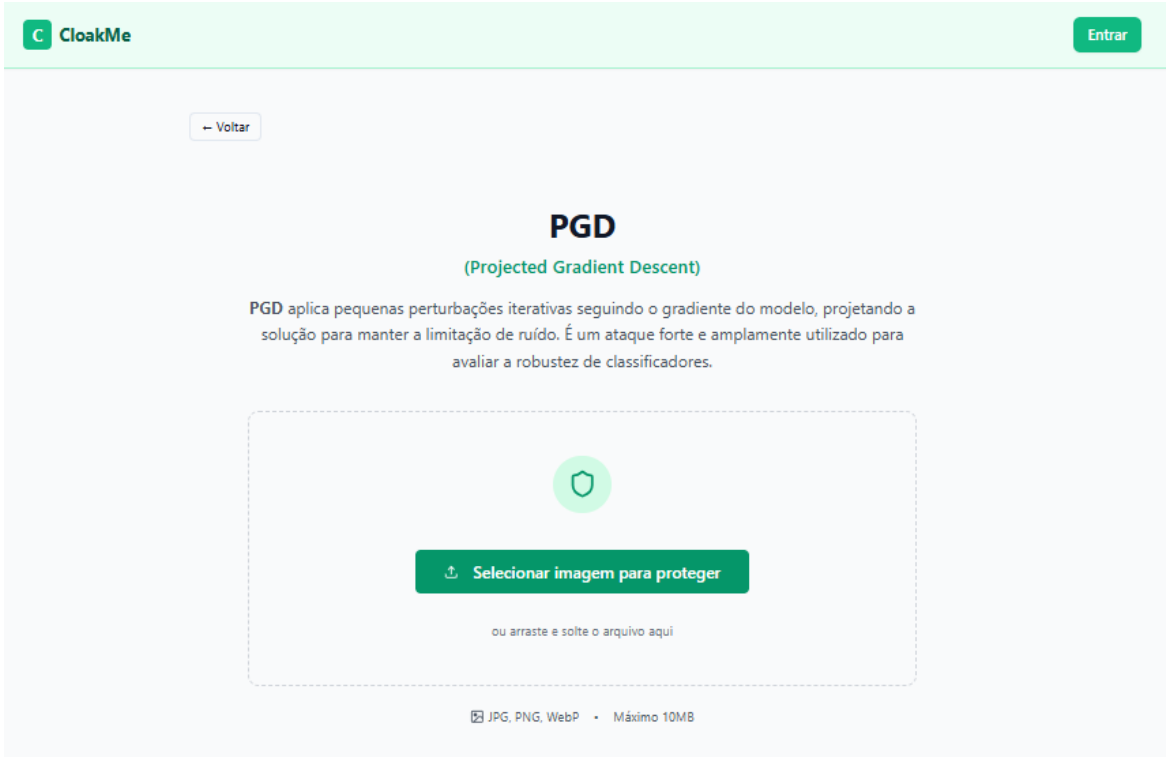
Fonte: Autor, 2025.

Figura 8 – Página do Método *One Pixel Attack*



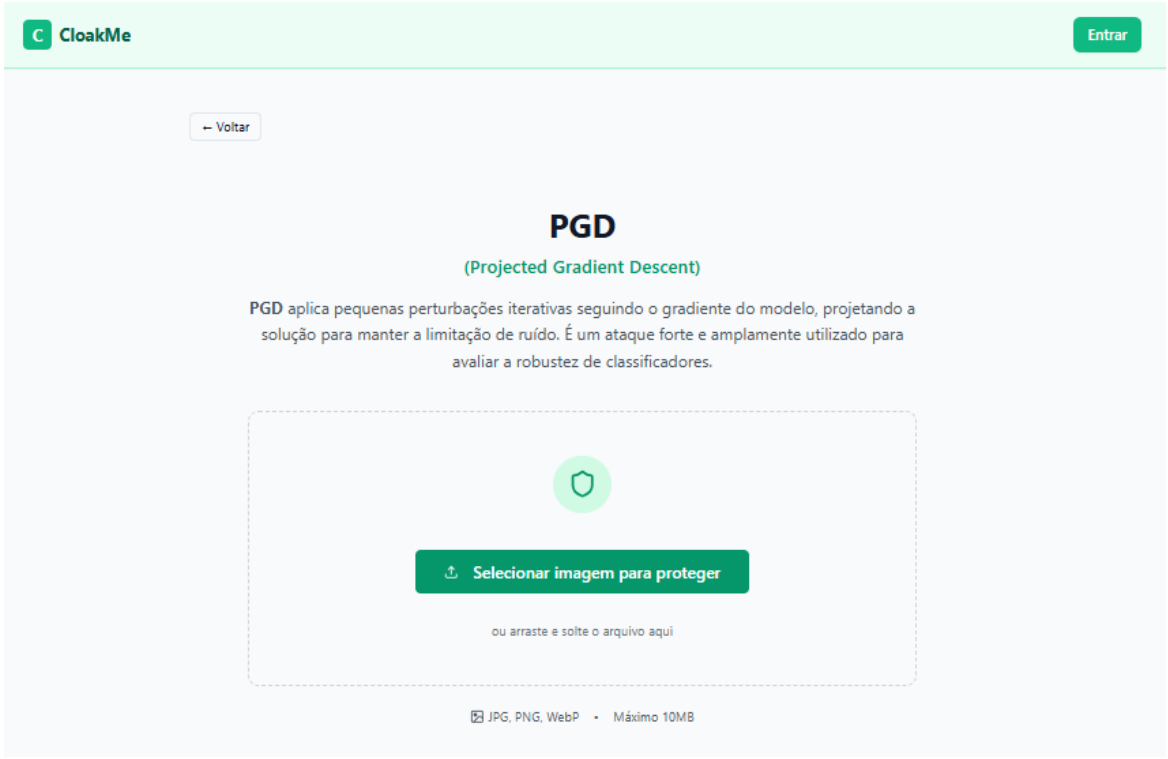
Fonte: Autor, 2025.

Figura 9 – Página do Método *Projected Gradient Descent*



Fonte: Autor, 2025.

Figura 10 – Página do Método *Carlini & Wagner*



Fonte: Autor, 2025.

4.4 Técnicas de Proteção Utilizadas

As técnicas de proteção implementadas baseiam-se em estratégias distintas de perturbação adversarial, com o objetivo de envenenar imagens contra modelos de IA. Cada técnica é representada por uma subclasse da classe abstrata `ProtecaoImagem`:

- **One Pixel Attack:** utiliza evolução diferencial para otimizar a posição e cor de um único pixel, buscando causar erro de classificação. É um ataque extremamente esparso e demonstra como mudanças mínimas podem afetar redes neurais profundas.
- **CW Attack:** utiliza otimização contínua para minimizar a norma da perturbação e, simultaneamente, maximizar a probabilidade de erro do modelo. Gera resultados altamente eficazes e é considerado um dos ataques mais fortes disponíveis.
- **PGD:** ataque iterativo que aplica perturbações sucessivas limitadas por um raio ε. É amplamente utilizado como referência de ataque na literatura por sua robustez e consistência.

Essas técnicas preservam a aparência visual da imagem, mas provocam distorções significativas na interpretação dos modelos de IA. O sistema CloakMe permite que o usuário escolha a técnica desejada, oferecendo uma defesa personalizável e escalável contra usos não autorizados de conteúdo visual.

Os três diagramas apresentados, de casos de uso, sequência e classes, oferecem uma visão integrada do funcionamento do sistema CloakMe. O primeiro destaca as funcionalidades principais do ponto de vista do usuário, o segundo descreve a dinâmica de execução entre os componentes, e o terceiro detalha a estrutura lógica e modular do sistema. Em conjunto, tais diagramas auxiliam na análise, evolução e manutenção da plataforma.

5 MATERIAIS E MÉTODOS

Este capítulo descreve os métodos de geração adversarial empregados, as escolhas de conjuntos de dados e modelos, a arquitetura das implementações, os parâmetros utilizados nos experimentos e o protocolo de avaliação. As técnicas incluídas são: One-Pixel Attack (implementado via *Differential Evolution*), *Projected Gradient Descent* (PGD) e o ataque Carlini & Wagner (CW).

5.1 Visão geral e objetivos

O objetivo é avaliar a eficácia das diferentes estratégias de envenenamento (perturbações adversariais) para reduzir a confiança de modelos de classificação ou provocar a classificação incorreta, preservando ao máximo a aparência visual da imagem. Para tanto selecionamos três métodos adversariais: One-Pixel Attack, um ataque extremamente esparso e visível, PGD, um ataque de primeira ordem iterativo e limitado em norma e CW, um ataque de otimização contínua capaz de encontrar perturbações pequenas e eficazes.

5.2 Conjuntos de dados e modelos

5.2.1 Conjuntos de dados

Os experimentos foram conduzidos com imagens provenientes do banco de dados:

- **ImageNet Large Scale Visual Recognition Challenge (ILSVRC):** conjunto de referência com imagens de resolução original variada; para inferência com modelos pré-treinados redimensionamos para 224×224px. (Deng et al., 2009)

5.2.2 Modelo alvo:

O modelo escolhido foi a ResNet50 é uma rede neural convolucional profunda composta por 50 camadas e amplamente empregada em tarefas de visão computacional, como classificação, detecção e segmentação de imagens. Esse modelo integra a família das *Residual Networks* (ResNet), cuja contribuição central reside na utilização de blocos residuais com conexões de atalho (*skip connections*), que facilitam o fluxo do gradiente durante o treinamento. Essa abordagem mitiga o problema do desvanecimento do gradiente, permitindo o desenvolvimento de arquiteturas mais profundas sem

degradação significativa do desempenho. A ResNet50 utiliza blocos do tipo *bottleneck*, combinando convoluções, normalização e funções de ativação de forma eficiente, o que resulta em elevada acurácia em bases de dados de referência, como o ImageNet, e na sua ampla adoção como *backbone* em modelos modernos de aprendizagem profunda. (He et al., 2015a)

5.3 Implementação e parâmetros

5.3.1 Ferramentas e infraestrutura

A implementação foi desenvolvida em Python, utilizando as bibliotecas: `PyTorch` (Paszke et al., 2019), `torchvision` (modelo e inferência), `SciPy` (Virtanen et al., 2020), `Pillow` e `NumPy` (Harris et al., 2020) para manipulação de imagens, além de `FastAPI` para exposição via API. Para acelerar a inferência e a otimização, utilizamos GPU quando disponível (PyTorch com CUDA); em ambientes sem GPU, os métodos executam em CPU com tempo de execução superior.

5.3.2 Parâmetros Adotados

Os valores apresentados a seguir foram adotados nas execuções relatadas neste trabalho; a escolha priorizou reprodutibilidade e equilíbrio entre taxa de sucesso e tempo de resposta devido ao hardware limitado para realização de testes.

- **One-Pixel (DE)**: `max_iters`=200 (gerações), `popsize`=15 (população efetiva = `popsize` × dim = 15 × 5 = 75), `polish`=True, `restarts`=2.
  Justificativa: população efetiva elevada aumenta cobertura do espaço multimodal; 200 gerações com 2 restarts apresentaram boas taxas de sucesso sem latência excessiva para um serviço interativo.

- **PGD**: $\varepsilon = 0.03$ (escala [0,1]), $\alpha = \varepsilon/4 = 0.0075$, $T = 20$ iterações, `random_start`=True.
  Justificativa: valores comuns na literatura para ataques imperceptíveis em imagens 224×224; equilibram clareza visual e eficácia.

- **CW**: `max_iters`=1000, `lr`=0.01, fator de penalização c testado no seguinte grid $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ com estratégia de busca sequencial (*increasing search*), reparametrização para garantir box constraints.
  Justificativa: CW demanda mais iterações; busca em c evita escolha fixa e melhora taxa de sucesso sem aumentar demasiadamente a magnitude da perturbação.

5.3.3 Sobre normalização e pré-processamento

As imagens são convertidas para tensor e normalizadas com os vetores médias/desvios do respectivo dataset antes de alimentar o modelo:

- ImageNet: $\mu = (0.485, 0.456, 0.406)$, $\sigma = (0.229, 0.224, 0.225)$; resize para 256 e center crop 224.

5.3.4 Métricas coletadas

- **Taxa de Sucesso (Success Rate - SR)**: proporção de amostras em que o ataque ocasionou erro na classificação.
- **Confiança (confidence)**: probabilidade (*softmax*) da classe prevista antes (p_{orig}) e depois (p_{final}) do ataque.
- **Queda de confiança absoluta e relativa**: $\Delta p = p_{orig} - p_{final}$ e $100 \times (p_{orig} - p_{final})/p_{orig}$.
- **Classificação de classes**: mudança no conjunto das k classes mais prováveis (tipicamente k=3).
- **Tempo por amostra**: tempo médio de execução do ataque por imagem (útil para avaliar viabilidade em serviço).
- **Magnitude da perturbação**: quando aplicável, normas ℓ_2 e ℓ_∞ da diferença entre imagens original e perturbada.

5.4 Procedimento de ataque

Para cada imagem escolhida aplica-se o método respectivo até que ocorra sucesso (misclassificação) ou até esgotar os limites de execução:

- **One-Pixel:** executa-se o algoritmo de Differential Evolution com os parâmetros da imagem, interrompendo se algum indivíduo da população gerar erro na classificação. Caso sejam definidos *restarts*, realiza-se novas execuções com população diferente até o limite.
- **PGD:** aplica-se iterações de gradient sign com projeção na norma ℓ_∞ de raio ε, opcionalmente com início aleatório. O ataque termina quando T iterações são completadas ou quando a predição muda.

- **CW:** resolve-se a otimização contínua proposta por Carlini & Wagner com reparametrização e critérios de parada por número de iterações ou por atingimento de sucesso (quando a função objetivo indica que a predição mudou).

5.5 Métricas e definições formais

As métricas calculadas por amostra (índice i) e suas agregações são apresentadas a seguir. O *Success Rate (SR)* é um indicador de sucesso binário por amostra, definido por:

$$succ^{(i)} = \mathbf{1}\{\mathrm{pred}^{(i)}_{final} \neq \mathrm{pred}^{(i)}_{orig}\},$$

Onde $\mathbf{1}\{\cdot\}$ é a função indicadora; A taxa de sucesso sobre N amostras é então:

$$SR = \frac{1}{N}\sum_{i=1}^{N} succ^{(i)}.$$

Para cada amostra registram-se também $p^{(i)}_{orig}$ e $p^{(i)}_{final}$, que correspondem às probabilidades (softmax) atribuídas à classe prevista antes e depois do ataque, respectivamente; as médias dessas probabilidades são:

$$\overline{p_{orig}} = \frac{1}{N}\sum_{i=1}^{N} p^{(i)}_{orig}, \qquad \overline{p_{final}} = \frac{1}{N}\sum_{i=1}^{N} p^{(i)}_{final}.$$

A queda de confiança por amostra é dada por:

$$\Delta p^{(i)} = p^{(i)}_{orig} - p^{(i)}_{final},$$

Para essa queda reportam-se a média e o desvio amostral, respectivamente:

$$\overline{\Delta p} = \frac{1}{N}\sum_{i=1}^{N} \Delta p^{(i)}, \qquad \mathrm{std}(\Delta p) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} (\Delta p^{(i)} - \overline{\Delta p})^2}.$$

Quanto à mudança na classificação, seja $T^{(i)}_k$ o conjunto das k classes mais prováveis antes do ataque e $T^{(i),final}_k$ o conjunto após o ataque; define-se então o indicador de mudança no top-k como:

$$shift^{(i)}_k = \mathbf{1}\{T^{(i)}_k \neq T^{(i),final}_k\},$$

A porcentagem de amostras com mudança no top-k é dada por:

$$Top{-}k\_shift = \frac{1}{N}\sum_{i=1}^{N} shift^{(i)}_k.$$

Medimos também o tempo de execução do ataque por amostra (em segundos), obtido via timestamps imediatamente antes e depois da rotina, do qual reportamos a média e o

desvio-padrão. Finalmente, quando aplicável (por exemplo em PGD e CW) calculamos a magnitude da perturbação $\delta^{(i)} = x'^{(i)} - x^{(i)}$ através das normas:

$$\|\delta^{(i)}\|_\infty = \max_j |\delta_j^{(i)}|, \qquad \|\delta^{(i)}\|_2 = \sqrt{\sum_j (\delta_j^{(i)})^2},$$

Enquanto para o One-Pixel reporta-se a mudança no pixel (valor absoluto) e, se desejado, normas agregadas da perturbação.

5.6 Ética e uso responsável

Os experimentos e as ferramentas desenvolvidas neste trabalho destinam-se exclusivamente a fins de pesquisa, avaliação e proteção de dados pessoais. Em nenhum momento foi cogitado a utilização das técnicas adversariais aqui descritas para finalidades ilícitas ou antiéticas.

6 TECNOLOGIAS UTILIZADAS

Este capítulo apresenta as principais tecnologias empregadas no desenvolvimento da plataforma CloakMe. As ferramentas, linguagens e bibliotecas descritas a seguir foram selecionadas por sua relevância no contexto do desenvolvimento web e de aplicações baseadas em processamento de imagens e inteligência artificial. Cada tecnologia desempenha um papel fundamental na estrutura e no funcionamento do sistema, permitindo a integração entre interface, *backend* e módulo de processamento.

6.1 Interface Web – *Front-end*

A interface da aplicação foi desenvolvida utilizando o React.js, uma biblioteca JavaScript voltada para a criação de interfaces de usuário dinâmicas e responsivas. O React adota o conceito de *componentização*, que permite a construção de elementos reutilizáveis e facilita a manutenção do código. Essa abordagem é especialmente útil em aplicações web que necessitam de atualização constante dos elementos visuais sem recarregar a página. Além disso, o React oferece ampla compatibilidade com bibliotecas auxiliares, como roteadores e gerenciadores de estado, que contribuem para a criação de uma navegação fluida e intuitiva, essencial para o uso acessível da ferramenta. (Meta Platforms, Inc., 2024)

6.2 *backend* e Gerenciamento de Requisições

O *backend* do sistema foi implementado em Python, utilizando o microframework **Flask**. O Flask é uma ferramenta leve e flexível para o desenvolvimento de aplicações web baseadas em APIs, oferecendo suporte à criação de rotas, manipulação de requisições HTTP e integração com módulos externos. No contexto do CloakMe, o Flask atua como intermediário entre a interface web e o módulo de processamento, recebendo as imagens enviadas pelos usuários, realizando a validação dos arquivos e retornando os resultados após a aplicação das técnicas de proteção. Sua arquitetura modular e a simplicidade de uso tornam-no adequado para aplicações que exigem integração com algoritmos de aprendizado de máquina e manipulação de arquivos. (FLASK…, 2024)

6.3 Processamento e Manipulação de Imagens

A etapa de processamento, responsável por aplicar as técnicas de *image poisoning*, foi desenvolvida em Python, com o uso das bibliotecas PyTorch, NumPy e OpenCV. O PyTorch é uma das bibliotecas mais populares para o desenvolvimento e execução de modelos de aprendizado profundo. Ele permite a criação e treinamento de redes neurais, bem como a manipulação de tensores e operações matemáticas complexas necessárias para o cálculo das perturbações adversariais aplicadas às imagens. O NumPy, por sua vez, é uma biblioteca de suporte matemático essencial, utilizada para operações de álgebra linear, manipulação de arrays e vetores, que são a base para o processamento de imagens digitais. Já o OpenCV (Open Source Computer Vision Library) é amplamente utilizado para leitura, escrita, conversão e transformação de imagens. Ele oferece uma grande variedade de funções para manipulação visual, sendo responsável pelas etapas de carregamento e exportação das imagens originais e protegidas.

6.4 Comunicação entre os Módulos

A comunicação entre o *Front-end* e o *backend* ocorre por meio do protocolo HTTP, seguindo o padrão de arquitetura RESTful. Nesse modelo, o sistema é dividido em recursos acessíveis por rotas específicas (como upload, processamento e download), que se comunicam por meio de requisições e respostas padronizadas. Essa estrutura permite que o site funcione de forma integrada e independente, com o *Front-end* enviando as imagens para o *backend* processar e recebendo de volta o arquivo protegido. O uso de APIs REST torna a aplicação mais escalável, interoperável e compatível com diferentes tipos de clientes web.

6.5 Gerenciamento e Armazenamento de Arquivos

Durante o processo de execução, as imagens enviadas pelos usuários são armazenadas temporariamente no servidor, utilizando o sistema de arquivos local. Essa abordagem garante que o processamento ocorra de forma segura e isolada, sem a necessidade de bancos de dados complexos. Após o término da operação, os arquivos podem ser automaticamente removidos, garantindo a privacidade dos dados e evitando acúmulo desnecessário de informações.

6.6 Síntese das Tecnologias

De forma geral, as tecnologias utilizadas no desenvolvimento do CloakMe permitem a integração entre usabilidade, desempenho e segurança. O React.js assegura uma experiência de navegação fluida e moderna, enquanto o Flask, aliado às bibliotecas Python voltadas ao processamento de imagens, fornece a estrutura necessária para o funcionamento do sistema de forma modular e eficiente. A combinação dessas tecnologias viabiliza a implementação de um ambiente web completo, capaz de oferecer ao usuário uma ferramenta acessível para aplicar técnicas de proteção de imagens contra inteligências artificiais, garantindo tanto a preservação da aparência original quanto a inutilização dos dados para fins de treinamento automatizado.

7 RESULTADOS E DISCUSSÃO

Nesta seção descrevemos de forma precisa os critérios adotados para validar os métodos implementados (One-Pixel via DE, PGD e CW), o protocolo experimental aplicado e as métricas calculadas. O objetivo é permitir reprodutibilidade e clareza na interpretação dos resultados.

7.1 Treinamento do Modelo

O modelo utilizado nos experimentos foi a ResNet50 (He et al., 2015b), escolhida por sua ampla adoção em tarefas de visão computacional e por servir como referência sólida na literatura. O treinamento seguiu o paradigma de *transfer learning*: partiu-se dos pesos pré-treinados no conjunto ImageNet (Deng et al., 2009), o que permite ao modelo herdar representações visuais gerais antes de ser ajustado ao domínio específico do experimento. Para isso, a última camada totalmente conectada foi substituída por uma camada compatível com o número de classes do problema, mantendo-se as demais estruturas da rede.

Após essa adaptação, realizou-se o *fine-tuning* do modelo utilizando o otimizador Adam (Kingma; Ba, 2017) e função de perda de entropia cruzada. A taxa inicial de aprendizado foi deliberadamente baixa, garantindo estabilidade durante o ajuste fino dos pesos herdados do ImageNet. Técnicas como *early stopping* e ajustes dinâmicos da taxa de aprendizado foram aplicadas conforme o comportamento da validação. Transformações de aumento de dados e normalização padrão por lote contribuíram para mitigar sobreajuste. Ao final do processo, obtiveram-se pesos estáveis e um desempenho consistente, suficiente para atuar como classificador-base nos experimentos de *image poisoning* desenvolvidos neste trabalho.

7.2 Amostragem e pré-filtragem

Para cada rodada de validação adotou-se o seguinte procedimento de amostragem:

1. Seleção aleatória de um conjunto de 100 imagens do conjunto de teste do dataset ImageNet, garantindo que cada imagem seja inicialmente corretamente classificada pelo modelo alvo. Apenas imagens com predição inicial correta são consideradas para ataque (pré-filtragem).

2. Fixação de sementes pseudoaleatórias (torch.manual_seed(42), numpy.random-.seed(42)) para as etapas não determinísticas (inicialização do DE, embaralhamento, etc.), quando aplicável, visando reprodutibilidade controlada.

7.3 Resultados Obtidos

A Tabela 2 apresenta um conjunto de resultados obtidos com $N = 100$ imagens por método (itens previamente descritos). Os números servem como ilustração do tipo de relatório que deve ser produzido após a execução empírica real.

Tabela 2 – Resultados agregados por método — $N = 100$

Método	SR (%)	$\overline{p_{orig}}$	$\overline{p_{final}}$	$\overline{\Delta p}$	Tempo médio (min)	Top-3 shift (%)
One-Pixel (DE)	42.0	0.86 ± 0.06	0.34 ± 0.18	0.52 ± 0.19	19.3 ± 0.9	48.0
PGD (untargeted)	78.0	0.88 ± 0.05	0.12 ± 0.10	0.76 ± 0.12	7.8 ± 1.1	87.0
CW (optimização)	91.0	0.89 ± 0.05	0.09 ± 0.06	0.80 ± 0.11	9.5 ± 3.4	92.0

Fonte: Autor, 2025.

Legenda: SR = Success Rate. Valores de probabilidade indicam média ± desvio-padrão. Top-3 shift = porcentagem de imagens cuja lista das 3 classes mais prováveis mudou após o ataque. classes mais prováveis mudou após o ataque.

7.4 Análise e Discussão

Os resultados permitem as seguintes observações e conclusões. Em termos de comparação de eficácia entre as técnicas avaliadas, o método *CW* apresenta a maior taxa de sucesso (91%) e grande redução média de confiança, o que está de acordo com a literatura que o caracteriza como um dos ataques mais potentes, especialmente quando se busca perturbações de baixa magnitude. O principal custo associado a esse método é o tempo computacional, que se reflete no tempo médio por amostra mais elevado entre os experimentos.

O método *PGD* alcança boa eficácia (78%) com tempo de execução moderado, o que representa um equilíbrio adequado entre desempenho e custo e explica sua ampla adoção como referência em avaliações de robustez de modelos. O método *One-Pixel* demonstra que uma perturbação extremamente esparsa é capaz de reduzir significativamente a confiança do modelo (média $\Delta p \approx 0.52$) e produzir classificações incorretas em parcela relevante das amostras (42%). Embora apresente a menor taxa de sucesso entre os métodos analisados devido à sua simplicidade, destaca-se como a técnica menos perceptível visualmente, pois altera apenas um único pixel da imagem.

A avaliação realizada, considerando os experimentos e análises anteriores, evidencia que cada família de ataques adversariais envolve um conjunto distinto de compromissos entre eficácia, custo computacional e perceptibilidade visual. Métodos mais sofisticados tendem a

gerar maior taxa de sucesso e maior impacto sobre modelos de inteligência artificial, porém exigem maior tempo de processamento e maior demanda de recursos. Em contraste, técnicas mais simples, como o *One-Pixel*, mesmo apresentando menor taxa de sucesso, são vantajosas em cenários nos quais a simplicidade da implementação e a imperceptibilidade das perturbações são prioridades, o que reforça seu valor estratégico.

No contexto do projeto, o desenvolvimento, a adaptação e a integração dos métodos na plataforma demonstraram-se bem-sucedidos tanto do ponto de vista funcional quanto metodológico. A implementação foi capaz de aplicar as técnicas de envenenamento de maneira consistente, preservando a aparência visual das imagens ao mesmo tempo em que reduziu a capacidade de interpretação dos modelos de inteligência artificial. Dessa forma, os resultados obtidos confirmam que o sistema atende aos requisitos estabelecidos para esta etapa e validam a viabilidade do uso de ataques adversariais como mecanismo de proteção de imagens na plataforma *CloakMe*.

8 CONCLUSÃO

O presente trabalho propôs o desenvolvimento de uma plataforma web capaz de aplicar técnicas de *image poisoning* com o objetivo de proteger imagens pessoais contra a leitura e utilização indevida por sistemas de inteligência artificial. A ferramenta desenvolvida, denominada CloakMe, visa oferecer aos usuários uma forma acessível e prática de dificultar o uso indevido de seus conteúdos visuais em modelos de aprendizado de máquina.

Os ganhos esperados com a implementação da plataforma incluem a democratização do acesso a ferramentas de defesa contra IA, uma vez que seu funcionamento simplificado e acessível pela web permite que qualquer usuário aplique técnicas de *image poisoning* sem depender de conhecimento técnico ou infraestrutura especializada. Após remover essas barreiras de entrada, a plataforma contribui também para a preservação da privacidade digital dos usuários e para a promoção de práticas mais éticas no uso de dados no treinamento de modelos. Além disso, o projeto busca fomentar o debate sobre os limites éticos e técnicos do uso de inteligência artificial em conteúdos visuais disponíveis na internet.

Durante o desenvolvimento, foram identificados desafios significativos, como a complexidade técnica do módulo de processamento, que envolve conceitos avançados de aprendizado de máquina e otimização. Também foi necessário garantir um desempenho satisfatório mesmo em dispositivos com recursos computacionais limitados, além de assegurar a comunicação eficiente entre os componentes do sistema, frontend, *backend* e APIs especializadas. Outro desafio relevante foi a avaliação da eficácia das perturbações geradas, considerando o comportamento dos modelos de IA frente aos métodos aplicados (One Pixel Attack, PGD e Carlini & Wagner).

Com base nos testes realizados e na análise dos resultados obtidos, considera-se que o projeto é viável técnica e operacionalmente. Essa constatação se apoia em três evidências principais:

1. A integração entre os módulos foi validada por meio do funcionamento completo do ciclo de envio, processamento e retorno da imagem protegida;
2. As APIs especializadas demonstraram estabilidade e tempo de resposta adequados, mesmo em execuções locais com hardware limitado;
3. As técnicas adversariais aplicadas produziram resultados consistentes, com redução perceptível na confiança das classificações do modelo alvo (ResNet50), comprovando a efetividade do método de proteção.

O uso de tecnologias consolidadas, como FastAPI, PyTorch, React e Tailwind, aliado à arquitetura modular do sistema, contribuiu diretamente para a robustez e escalabilidade da plataforma. Com as melhorias e testes adicionais planejados, o projeto apresenta potencial

para evoluir como uma ferramenta funcional, de código aberto e com impacto positivo na conscientização sobre privacidade digital e ética no uso da inteligência artificial.

Dessa forma, o CloakMe alcança seu objetivo central: disponibilizar uma solução prática, transparente e tecnicamente sólida para proteção de imagens contra sistemas de IA, promovendo a autonomia e a segurança dos usuários no ambiente digital.

8.1 Trabalhos Futuros

Embora o sistema desenvolvido já cumpra seu objetivo de aplicar técnicas de image poisoning para proteção de imagens, ainda existem caminhos promissores para aprofundamentos futuros. Um dos avanços possíveis é avaliar a eficácia das perturbações em um conjunto mais amplo de modelos de IA, incluindo arquiteturas mais recentes e sistemas multimodais. Essa análise comparativa permitiria compreender como diferentes modelos reagem aos padrões de envenenamento e auxiliar na escolha ou evolução das técnicas aplicadas.

Além dessa expansão técnica, trabalhos futuros também podem considerar a incorporação de mecanismos que tornem o processo de proteção mais adaptativo e personalizado. Isso inclui desde ajustes automáticos da intensidade da perturbação conforme o conteúdo da imagem, até a criação de camadas adicionais de defesa capazes de atuar de forma complementar ao envenenamento adversarial. Investigações que avaliem o comportamento do sistema frente a novos cenários de uso, como proteção de grandes volumes de dados, integração com plataformas externas ou acompanhamento contínuo da evolução dos modelos de IA, podem reforçar a maturidade e a aplicabilidade da solução em contextos reais.

REFERÊNCIAS

CARLINI, N.WAGNER, D. A. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016. Disponível em: <http://arxiv.org/abs/1608.04644>. Citado na página 20.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2009. p. 248–255. Citado 2 vezes nas páginas 33 e 41.

FBSP, F. B. D. S. P. 19º anuário brasileiro de segurança pública. *São Paulo: Fórum Brasileiro de Segurança Pública*, 2025. Disponível em: <https://publicacoes.forumseguranca.org.br/handle/123456789/279>. Citado na página 12.

FLASK – A lightweight WSGI web application framework. 2024. <https://flask.palletsprojects.com/>. Citado 2 vezes nas páginas 38 e 48.

FOUNDATION, P. S. *Python Programming Language – Official Website*. 2024. <https://www.python.org/>. Citado na página 48.

HARRIS, C. R.; MILLMAN, K. J.; WALT, S. J. van der; GOMMERS, R.; VIRTANEN, P.; COURNAPEAU, D.; WIESER, E.; TAYLOR, J.; BERG, S.; SMITH, N. J.; KERN, R.; PICUS, M.; HOYER, S.; KERKWIJK, M. H. van; BRETT, M.; HALDANE, A.; RÍO, J. F. del; WIEBE, M.; PETERSON, P.; GÉRARD-MARCHANT, P.; SHEPPARD, K.; REDDY, T.; WECKESSER, W.; ABBASI, H.; GOHLKE, C.OLIPHANT, T. E. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>. Citado na página 34.

HE, K.; ZHANG, X.; REN, S.SUN, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. Disponível em: <http://arxiv.org/abs/1512.03385>. Citado na página 34.

HE, K.; ZHANG, X.; REN, S.SUN, J. *Deep Residual Learning for Image Recognition*. 2015. Disponível em: <https://arxiv.org/abs/1512.03385>. Citado na página 41.

JULIBONI, M. Crimes cibernéticos com o uso de inteligência artificial avançam em todo mundo. *VEJA Negócios*, v. 9, 2024. Disponível em: <https://veja.abril.com.br/economia/crimes-ciberneticos-com-o-uso-de-inteligencia-artificial-avancam-em-todo-mundo/>. Citado na página 15.

KIETZMANN, J.; LEE, L. W.; MCCARTHY, I. P.KIETZMANN, T. C. Deepfakes: Trick or treat? *Business Horizons, Volume 63, Issue 2*, 2020. Disponível em: <https://doi.org/10.1016/j.bushor.2019.11.006>. Citado 2 vezes nas páginas 15 e 16.

KINGMA, D. P.BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Disponível em: <https://arxiv.org/abs/1412.6980>. Citado na página 41.

MACHADO, S. 'eram meu rosto e minha voz, mas era golpe': como criminosos 'clonam pessoas' com inteligência artificial. *BBC News Brasil*, 2024. Disponível em: <https://www.bbc.com/portuguese/articles/cd1jv45dq3go>. Citado na página 12.

MADRY, A.; MAKELOV, A.; SCHMIDT, L.; TSIPRAS, D.VLADU, A. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. Disponível em: <https://arxiv.org/abs/1706.06083>. Citado na página 19.

Meta Platforms, Inc. *React – A JavaScript library for building user interfaces*. 2024. <https://reactjs.org/>. Citado 2 vezes nas páginas 38 e 48.

MOHANTY, H.; ROUDSARI, A. H.LASHKARI, A. H. Robust stacking ensemble model for darknet traffic classification under adversarial settings. *Computers Security*, v. 120, p. 102830, 2022. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404822002243>. Citado na página 16.

PASZKE, A.; GROSS, S.; MASSA, F.; LERER, A.; BRADBURY, J.; CHANAN, G.; KILLEEN, T.; LIN, Z.; GIMELSHEIN, N.; ANTIGA, L.; DESMAISON, A.; KÖPF, A.; YANG, E. Z.; DEVITO, Z.; RAISON, M.; TEJANI, A.; CHILAMKURTHY, S.; STEINER, B.; FANG, L.; BAI, J.CHINTALA, S. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019. Disponível em: <http://arxiv.org/abs/1912.01703>. Citado na página 34.

SHAN, S.; CRYAN, J.; WENGER, E.; ZHENG, H.; HANOCKA, R.ZHAO, B. Y. *Glaze: Protecting Artists from Style Mimicry by Text-to-Image Models*. 2025. Disponível em: <https://arxiv.org/abs/2302.04222>. Citado na página 22.

SHAN, S.; DING, W.; PASSANANTI, J.; WU, S.; ZHENG, H.ZHAO, B. Y. *Nightshade: Prompt-Specific Poisoning Attacks on Text-to-Image Generative Models*. 2024. Disponível em: <https://arxiv.org/abs/2310.13828>. Citado 2 vezes nas páginas 22 e 23.

VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; HABERLAND, M.; REDDY, T.; COURNAPEAU, D.; BUROVSKI, E.; PETERSON, P.; WECKESSER, W.; BRIGHT, J.; van der Walt, S. J.; BRETT, M.; WILSON, J.; MILLMAN, K. J.; MAYOROV, N.; NELSON, A. R. J.; JONES, E.; KERN, R.; LARSON, E.; CAREY, C. J.; POLAT, İ.; FENG, Y.; MOORE, E. W.; VanderPlas, J.; LAXALDE, D.; PERKTOLD, J.; CIMRMAN, R.; HENRIKSEN, I.; QUINTERO, E. A.; HARRIS, C. R.; ARCHIBALD, A. M.; RIBEIRO, A. H.; PEDREGOSA, F.; van Mulbregt, P.SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, v. 17, p. 261–272, 2020. Citado na página 34.