UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
BACHARELADO EM INTELIGÊNCIA ARTIFICIAL
INTRODUÇÃO A INTELIGÊNCIA ARTIFICIAL - IMD3001

ARTHUR ANDRADE GALVÍNCIO RODRIGUES
RAFAELA ARAÚJO DE ALBUQUERQUE

CLASSIFICAÇÃO DE IMAGENS DE MUCOSA GÁSTRICA PARA DIAGNÓSTICO DE *HELICOBACTER PYLORI*

NATAL
2025

Resumo

Este relatório detalha o desenvolvimento e a avaliação de um sistema modular de Deep Learning para a classificação binária da infecção por *Helicobacter pylori* em imagens histopatológicas da mucosa gástrica. O dataset **DeepHP**, proveniente da Universidade Federal do Pará (UFPA - "DeepHP: A New Gastric Mucosa Histopathology Dataset for Helicobacter pylori Infection Diagnosis"), foi o foco central. A metodologia empregou o modelo Inception V3, otimizado por um fine-tuning em duas fases (camadas congeladas e descongeladas), que alcançou uma acurácia excepcional. Discutem-se as ferramentas de IA utilizadas, as métricas de desempenho e as decisões de refinamento do modelo. O sistema foi encapsulado em uma API (FastAPI) e uma interface gráfica (Streamlit). Aborda-se criticamente a natureza da alta acurácia, considerando a possibilidade de super otimização relacionada à origem dos "tiles" e à importância de ferramentas de interpretabilidade, como o SHAP, para validação das decisões do modelo.

Disponível em: https://github.com/ArthurGalvincio

Palavras-chave: Deep Learning, Classificação de Imagens, *Helicobacter pylori*, Histopatologia, DeepHP, Inception V3, FastAPI, Streamlit, Interpretabilidade, SHAP.

1. INTRODUÇÃO

O diagnóstico ágil e preciso da infecção por *Helicobacter pylori* é fundamental para a saúde gastrointestinal. A análise histopatológica de biópsias gástricas é um método diagnóstico padrão, mas sua natureza manual pode ser demorada e sujeita à variabilidade interobservadora. Este projeto explora o potencial do Deep Learning para automatizar e otimizar esse processo, visando aumentar a eficiência e a consistência diagnóstica.

1.1 Contextualização do Problema e Objetivos

A *Helicobacter pylori* é uma bactéria amplamente reconhecida por seu papel na etiologia de diversas afecções gástricas, desde gastrite crônica e úlceras pépticas até um risco aumentado de câncer gástrico. A detecção precoce e precisa é, portanto, de grande relevância clínica.

O objetivo principal deste projeto foi construir e avaliar um sistema robusto de classificação binária para *H. pylori*, utilizando o dataset DeepHP(Coloração H&E). Especificamente, buscou-se:

- Desenvolver um pipeline de treinamento otimizado para modelos pré-treinados em imagens histopatológicas.
- Avaliar a capacidade de generalização dos modelos em diferentes escalas do dataset.
- Encapsular o modelo preditivo em uma interface acessível e intuitiva.
- Explorar a interpretabilidade do modelo para aumentar a confiança em suas decisões.

2. REFERENCIAL TEÓRICO E ABORDAGEM DE INTELIGÊNCIA ARTIFICIAL

2.1 Redes Neurais Convolucionais (CNNs)

Redes Neurais Convolucionais (CNNs) são uma classe de redes neurais artificiais amplamente utilizadas para análise de imagens. Inspiradas no córtex visual, as CNNs são capazes de aprender hierarquias de características diretamente dos dados, desde bordas e texturas em camadas iniciais até objetos complexos em camadas mais profundas. Elas são a base de muitos sistemas modernos de visão computacional.

2.2 Algoritmos de IA e Modelagem de Agentes (no Contexto de CNNs)

No contexto deste projeto de classificação de imagens, os "algoritmos de IA" referem-se às arquiteturas de redes neurais e aos métodos de otimização que as governam, e a "modelagem de agentes" pode ser reinterpretada como a forma como o sistema percebe e age.

2.2.1 Modelos de Rede (Algoritmos de IA)

Foram utilizadas três arquiteturas de CNN pré-treinadas no ImageNet, servindo como "algoritmos de IA" para extração de características e classificação:

- **Inception V3:** Escolhido por sua eficácia na extração de características multi-escala através de "módulos Inception" e sua eficiência computacional. Destaca-se por sua capacidade de aprender representações complexas.
- **VGG16:** Uma arquitetura mais clássica, conhecida por sua simplicidade e profundidade alcançada através de camadas convolucionais 3x3 empilhadas.
- **ResNet50:** Utiliza "conexões residuais" que permitem o treinamento de redes muito mais profundas, mitigando o problema do vanishing gradient.

2.2.2 Métodos de Otimização

Para o treinamento desses modelos, foram empregados algoritmos de otimização de IA:

- **Otimizador Adam:** Um algoritmo de otimização adaptativo, amplamente utilizado por sua eficiência e robustez na convergência.
- **Agendadores de Taxa de Aprendizagem (Schedulers):** Como ReduceLROnPlateau e CosineAnnealingLR, que ajustam a taxa de aprendizado dinamicamente durante o treinamento para melhorar a convergência e evitar platôs.

2.2.3 Representação de Conhecimento

Em CNNs, a "representação de conhecimento" não é explícita (como em sistemas baseados em regras), mas sim implícita nas hierarquias de características aprendidas pelas camadas convolucionais. O modelo "aprende" a representar a presença de *H. pylori* através de padrões visuais progressivamente complexos.

2.3 Arquitetura do Sistema e Pipeline de Processamento (PEAS Reinterpretado)

Embora a modelagem PEAS (Percepção, Ação, Ambiente, Sensores, Atuadores) seja mais comum para agentes de IA clássicos, podemos reinterpretá-la no contexto de uma CNN para classificação de imagens:

- **Ambiente:** O conjunto de imagens histopatológicas da mucosa gástrica (o dataset

DeepHP).

- **Sensores:** A entrada de imagem processada (pixels, normalização, redimensionamento), fornecida ao modelo.
- **Agente (o Modelo de DL):** A rede neural convolucional treinada (Inception V3, VGG16, ResNet50).
  - **Percepção:** As camadas convolucionais do modelo "percebem" padrões e características visuais na imagem.
  - **Ação:** A saída da camada final (classificador) do modelo, que é a predição binária (positive ou negative).
- **Atuadores:** O sistema que recebe a predição e a comunica ao usuário ou a outro sistema (a API FastAPI, a interface Streamlit).

A arquitetura do sistema é modular, com componentes distintos para cada etapa da pipeline (treinamento, avaliação, predição, interface).

3. MATERIAIS E MÉTODOS

3.1 Aquisição e Preparação do Dataset DeepHP

O dataset **DeepHP** é a pedra angular deste projeto. Foi obtido em parceria com a Universidade Federal do Pará (UFPA) e detalhado no artigo científico específico, sendo um conjunto diversificado de imagens de mucosa gástrica. O dataset possui aproximadamente 395.000 imagens, categorizadas como positive (com a bactéria) e negative (sem a bactéria).

3.1.1 Criação do Subconjunto deepHP_50k

Para otimizar a velocidade de treinamento e experimentação sem comprometer a representatividade, foi criado um subconjunto **deepHP_50k** usando o script create_deepHP_subset.py. Este script seleciona aleatoriamente 50.000 imagens da classe negative e 50.000 da classe positive do dataset deepHP total, totalizando 100.000 imagens. Este subconjunto foi então dividido em proporções de 70% treino, 15% validação e 15% teste, com estratificação para manter o balanceamento de classes.

3.2 Ferramentas e Tecnologias Utilizadas

- **PyTorch:** Framework principal para o desenvolvimento e treinamento dos modelos de Deep Learning.
- **FastAPI:** Utilizado para construir uma API RESTful de alta performance para a inferência do modelo.
- **Streamlit:** Empregado para criar uma interface gráfica de usuário web intuitiva para interação com a API e visualização de resultados.
- **Scikit-learn:** Usada para cálculo de métricas de classificação robustas.
- **SHAP (SHapley Additive exPlanations):** Biblioteca para análise de interpretabilidade de modelos, permitindo visualizar a contribuição de cada pixel na decisão final.
- **Hardware:** Todos os treinamentos e avaliações primárias foram conduzidos em um PC equipado com NVIDIA GeForce GTX 1650 (4GB VRAM).

3.3 Metodologia de Treinamento e Avaliação

3.3.1 Estrutura do Sistema de Treinamento (Scripts)

O sistema é modular, composto por scripts Python bem definidos:

- **train.py:** Orquestra o treinamento do modelo, incluindo carregamento de configurações, preparação de dados, loop de treino/validação, monitoramento de hardware, e salvamento de checkpoints. Foi aprimorado para retomar treinos interrompidos e aumentar a estabilidade em Windows.
- **evaluate.py:** Realiza a avaliação detalhada de modelos em datasets específicos, permitindo controle de dispositivo (GPU/CPU) e salvamento completo de métricas em JSON para análise.
- **predict.py:** Oferece funcionalidade de predição para imagens ou diretórios, com carregamento modular de modelos.
- **metrics.py:** Contém a classe BinaryClassificationMetrics, responsável por calcular acurácia, precisão, recall, F1-score e gerar matrizes de confusão/curvas ROC de forma robusta.

3.3.2 Critérios de Avaliação e Métricas

O desempenho dos modelos foi sistematicamente avaliado utilizando:

- **Métricas de Classificação:** Acurácia, Precisão, Recall, F1-Score (geral e por classe).
- **Métricas de Discriminação:** Área Sob a Curva ROC (AUC).
- **Métricas de Eficiência:** Tempo médio de inferência por imagem (ms/img), throughput (imagens/segundo) e uso de recursos de hardware (GPU, VRAM, RAM, CPU).

4. RESULTADOS E DISCUSSÃO

Os experimentos focaram na otimização do modelo Inception V3 no dataset deepHP, abordando a generalização e a eficiência.

4.1 Treinamento e Evolução do Inception V3 no deepHP_50k

O Inception V3 foi otimizado em duas fases, partindo de um modelo pré-treinado.

4.1.1 Fase 1: Treinamento com Camadas Congeladas (Baseline)

Esta fase inicial adaptou o classificador do Inception V3 ao deepHP_50k, mantendo o backbone congelado.

- **Configuração:** epochs: 50, learning_rate: 0.0005, augmentation: light, dropout: 0.4.
- **Tempo Total de Treino:** 13h 29min 48s.
- **Uso de Hardware (Médio):** GPU: 57.1%, VRAM: 47.7%, RAM: 84.9%, CPU: 63.9%. (Evidenciou gargalo de CPU devido a num_workers: 0 no Windows).
- **Melhor Acurácia Validação (deepHP_50k):** 95.61%.
- **Avaliação no Conjunto de Teste (deepHP Total - GPU):**
  - Acurácia: 95.38%.
  - Capacidade de Discriminação (AUC): 0.99.
- **Conclusão da Fase 1:** O modelo já obteve uma alta acurácia, demonstrando a eficácia do modelo pré-treinado e da light augmentation como ponto de partida.

4.1.2 Fase 2: Fine-tuning com Camadas Descongeladas (freeze_layers: false)

Esta fase buscou o refinamento máximo, permitindo o ajuste de todas as camadas do modelo (do backbone ao classificador), partindo do melhor modelo da Fase 1 e usando uma taxa de aprendizado muito fina.

- **Configuração:** epochs: 100, learning_rate: 0.00001, weight_decay: 0.0005, dropout: 0.5.
- **Tempo Total de Treino:** 21h 00min 15s. (Mais longo, como esperado para ajustar mais parâmetros).
- **Uso de Hardware (Médio):** GPU: 78.6%, VRAM: 84.2%, RAM: 85.4%, CPU: 50.3%. (Melhor aproveitamento da GPU).
- **Avaliação no Conjunto de Teste (deepHP Total - GPU):**
  - Acurácia: 99.17%.
  - Capacidade de Discriminação (AUC): 1.00.
- **Conclusão da Fase 2:** O fine-tuning profundo resultou em um **aumento notável na acurácia (quase 4% no teste)** e na capacidade de discriminação, atingindo performance próxima à perfeição no deepHP Total.

4.2 Discussão Crítica: Natureza da Alta Acurácia e Overfitting

A acurácia de 99.17% no teste (deepHP Total) é **alta para um dataset de imagens médicas**. Este resultado, embora impressionante, levanta discussões acadêmicas importantes:

- **Superotimização ao Dataset:** É possível que o modelo tenha se superotimizado para os padrões específicos presentes no deepHP_50k e deepHP Total. Isso pode ocorrer se o dataset, apesar do volume, possuir uma variabilidade limitada ou características muito distintivas entre as classes que o modelo memoriza.
- **Natureza dos "Tiles" e Coerência da Imagem:** A preocupação de que a alta acurácia possa ser influenciada pelo modelo "completando" a imagem (ou seja, detectando a ausência/presença da bactéria em um tile com base em informações de outros tiles da mesma imagem maior, se houver sobreposição ou contexto) é válida. Se os "tiles" (pequenas imagens) não são totalmente independentes em termos de informação visual, a métrica de acurácia por tile pode ser inflacionada para a imagem completa, em vez de refletir a capacidade de processar um tile isoladamente.

4.3 Implantação do Modelo: API e Interface Gráfica

Para demonstrar a aplicação prática do modelo, ele foi encapsulado em uma arquitetura cliente-servidor.

- **API com FastAPI:** Uma API RESTful (api/main.py) foi desenvolvida para servir o modelo. Ela aceita arquivos ZIP contendo múltiplas imagens (tiles), realiza a inferência na GPU e retorna os resultados de predição para cada tile em formato JSON (seja diretamente na resposta ou encapsulados em um ZIP para download).
- **Interface Gráfica com Streamlit:** Um frontend (streamlit_client_app.py) foi criado para interagir com a API. Ele permite que usuários façam upload de arquivos ZIP de

imagens, enviem para a API, visualizem as predições JSON diretamente no navegador e/ou baixem os resultados.

5. LIMITAÇÕES E MELHORIAS

5.1 Limitações e Dificuldades na Construção do Projeto

- **Erros de Memória e Carregamento:** Enfrentamento e depuração de RuntimeErrors, CUDA out of memory (resolvido por batch_size), IndexError (resolvido por metrics.py robusto).
- **Incompatibilidade SHAP e Modelos torchvision:** Dificuldades em fazer o DeepExplainer do SHAP funcionar devido a operações inplace=True internas do torchvision.models.inception_v3 e incompatibilidade com a tupla de saída do forward do modelo.

5.2 Limitações da Arquitetura Inception V3

- **Complexidade de Debug:** A arquitetura Inception, embora eficiente, pode ser mais complexa de depurar e adaptar (forward com saídas em tupla, otimização de inplace).
- **Custo Computacional do Treino:** O fine-tuning profundo de todas as camadas do Inception V3 exige tempo e recursos consideráveis (21 horas de treino).

5.3 Sugestões de Melhorias Futuras

- **Validação Externa Rigorosa:** A prioridade é validar o modelo em um **conjunto de dados de teste totalmente independente e nunca visto**, proveniente de uma fonte diferente e representativa de cenários de uso real, para confirmar a alta taxa de generalização.
- **Otimização do Pipeline de Treinamento:** Explorar o uso de um ambiente Linux para permitir num_workers > 0, o que melhoraria drasticamente a utilização da GPU e reduziria o tempo de treino.
- **Análise de Erros Qualitativa:** Inspecionar as poucas imagens mal classificadas para identificar padrões de erro e refinar o augmentation ou o pré-processamento.
- **Interpretabilidade Aprofundada:** Concluir a implementação e análise com a ferramenta SHAP para validar visualmente as regiões da imagem que mais influenciam a decisão do modelo, especialmente para os casos de fronteira ou onde a acurácia é muito alta.

6. CONCLUSÃO

O projeto resultou no desenvolvimento de um sistema robusto para a classificação de *Helicobacter pylori* em imagens com coloração H&E. O modelo Inception V3, submetido a um fine-tuning profundo, alcançou uma acurácia de 99.17% no dataset deepHP Total. Essa performance notável, embora necessite de validação em dados independentes, destaca o potencial do Deep Learning no diagnóstico histopatológico. A infraestrutura modular, a API (FastAPI) e a interface (Streamlit) garantem a aplicabilidade prática do sistema. O projeto também abordou desafios de otimização de treinamento e iniciou a exploração da interpretabilidade do modelo, pavimentando o caminho para futuras melhorias e validações em cenários clínicos reais.