

Classificação Multimodal de Lesões Orais: Uma Abordagem Híbrida Integrando Redes Neurais Convolucionais e Dados Clínicos Estruturados

Guilherme de Fraga Pires

Resumo: A classificação automatizada de lesões orais apresenta desafios significativos de Visão Computacional, exacerbados pela alta similaridade visual entre classes (inter-class similarity) e pelo severo desbalanceamento de dados em cenários do mundo real. Este trabalho propõe e avalia uma arquitetura de Aprendizado Profundo (*Deep Learning*) multimodal que funde características visuais extraídas via Redes Neurais Convolucionais (CNNs) com dados tabulares clínicos processados por *Perceptrons* Multicamadas (MLPs). Utilizando o *framework* PyTorch e a biblioteca *timm* (*PyTorch Image Models*), foi desenvolvido um modelo que processa simultaneamente o contexto da cavidade oral, o recorte focado da lesão (ROI) e metadados do paciente (idade, sexo e hábitos de risco). O pipeline de treinamento implementou estratégias rigorosas de prevenção de vazamento de dados (*Data Leakage*) através do particionamento baseado em identificadores de pacientes, gerenciado via *Pandas*. Foram realizados experimentos comparativos utilizando backbones modernos como o *ConvNeXt Tiny*, com estratégias distintas de *fine-tuning* e escalonamento de taxa de aprendizado. Os resultados experimentais demonstram que a integração de dados multimodais, aliada a técnicas de amostragem ponderada (*WeightedRandomSampler*), permite que o modelo atinja uma generalização robusta, alcançando um F1-Score macro de aproximadamente 0.76 e Acurácia de 81.57% no conjunto de teste, validando a eficácia da fusão de *features* na distinção de classes críticas como o Carcinoma Oral (OCA) e Desordens Potencialmente Malignas (OPMD).

Abstract: Automated classification of oral lesions poses substantial Computer Vision challenges, amplified by high inter-class visual similarity and severe real-world data imbalance. This work proposes and evaluates a multimodal Deep Learning architecture that fuses visual features extracted by Convolutional Neural Networks (CNNs) with clinical tabular data processed by Multilayer Perceptrons (MLPs). Using the PyTorch framework and the *timm* (PyTorch Image Models) library, we develop a model that simultaneously processes the global oral cavity context, a focused lesion crop (ROI), and patient metadata (age, sex, and risk habits). The training pipeline incorporates strict data-leakage prevention strategies through patient-ID-based partitioning, managed with *Pandas*. Comparative experiments were conducted using modern backbones such as *ConvNeXt-Tiny*, with distinct fine-tuning strategies and learning-rate scheduling. Experimental results show that integrating multimodal data, combined with weighted sampling techniques (*WeightedRandomSampler*), enables the model to achieve robust generalization, reaching a macro F1-Score of approximately 0.76 and an Accuracy of 81.57% on the test set. These findings validate the effectiveness of feature fusion in distinguishing critical classes such as Oral Cancer (OCA) and Oral Potentially Malignant Disorders (OPMD).

Palavras-chave—Deep Learning, Classificação Multimodal, *ConvNeXt*, PyTorch, Data Leakage, Fusão de Dados.

1 Introdução

A aplicação de Inteligência Artificial no domínio de diagnóstico por imagem tem evoluído de abordagens puramente visuais para sistemas mais complexos e holísticos. No contexto específico da classificação de lesões orais, os modelos tradicionais baseados exclusivamente em imagens enfrentam limitações intrínsecas. A aparência visual de lesões benignas frequentemente se sobrepõe à de desordens potencialmente malignas (OPMD), criando uma fronteira de decisão não linear e ruidosa para classificadores padrão. Além disso, a decisão clínica humana raramente é tomada isoladamente; ela é fortemente informada pelo histórico do paciente, incluindo fatores de risco como tabagismo e consumo de álcool.

Neste trabalho, será abordado o problema de classificação multiclasse (Saudável, Benigno, OPMD, OCA) sob uma perspectiva de engenharia de *Deep Learning* multimodal. Em vez de depender unicamente da extração de características visuais, propomos uma arquitetura híbrida que ingere e processa três fluxos de dados distintos: (1) uma visão global da cavidade oral, (2) uma visão focada na Região de Interesse (ROI) da lesão, e (3) dados estruturados do paciente.

1.1 Desafios de Engenharia e Modelagem

A construção de tal sistema impõe desafios técnicos que vão além da simples seleção de uma arquitetura de CNN. O primeiro desafio é a **heterogeneidade dos dados**. Imagens são dados de alta dimensão e não estruturados, enquanto os dados clínicos são tabulares, de baixa dimensão e frequentemente categóricos ou contínuos. A fusão eficaz dessas modalidades requer uma arquitetura que aprenda representações latentes compatíveis (*embeddings*) antes da camada de classificação final. Para solucionar isso, foi utilizado a biblioteca **timm** para instanciar *backbones* de última geração, especificamente a família *ConvNeXt*, que oferece um compromisso otimizado entre custo computacional e capacidade de extração de *features*.

O segundo e talvez mais crítico desafio é a **integridade dos dados e a validação**. Em conjuntos de dados médicos, é comum haver múltiplas imagens pertencentes ao mesmo paciente. Uma divisão aleatória simples (como a função *train_test_split* padrão do **Scikit-learn**) trataria cada imagem como uma amostra independente. Isso inevitavelmente levaria ao fenômeno de *Data Leakage* (vazamento de dados), onde o modelo "memoriza" características irrelevantes do paciente (como a iluminação específica ou anatomia dentária) presentes no treino e as reencontra no teste, inflando artificialmente as métricas de desempenho. Para mitigar esse risco, desenvolvemos algoritmos personalizados de particionamento que garantem o isolamento estrito de pacientes entre os conjuntos de treinamento, validação e teste.

1.2 Contribuições Técnicas

As principais contribuições deste estudo focam na arquitetura de software e na metodologia de treinamento:

1.2.1 Arquitetura de Fusão Tardia: Foi implementado um modelo modular em **PyTorch** que permite a substituição dinâmica do extrator de características visual (*backbone*), facilitando a experimentação com diferentes arquiteturas (e.g., ResNet, EfficientNet, MobileNet) sem refatoração do código de fusão.

1.2.2 Pré-processamento Orientado a ROI: Foram utilizadas anotações no formato COCO para gerar recortes dinâmicos das lesões, permitindo que a rede neural foque em padrões de textura locais sem perder o contexto global da cavidade oral.

1.2.3 Estratégias de Otimização Avançadas: Foi investigado o impacto de diferentes regimes de *fine-tuning*, comparando uma abordagem de descongelamento (*unfreezing*) progressivo baseada em frações de camadas contra janelas de congelamento explícitas combinadas com escalonamento de taxa de aprendizado em duas fases (*Two-phase LR scheduling*).

1.3 Ferramentas e Bibliotecas

O desenvolvimento deste projeto baseou-se em um ecossistema de ferramentas de código aberto. A manipulação e estruturação dos dados tabulares e manifestos de arquivos foram realizadas utilizando **Pandas**, garantindo eficiência na filtragem e agrupamento de metadados. O processamento de imagens, incluindo augmentação de dados (*Data Augmentation*) e transformações tensoais, foi executado via **Torchvision** e **PIL** (Python Imaging Library). Para o cálculo de métricas de avaliação complexas, como curvas ROC (*Receiver Operating Characteristic*) e Precision-Recall, o **Scikit-learn** foi empregado. O núcleo da modelagem e o loop de treinamento foram construídos sobre o **PyTorch**, aproveitando sua capacidade de diferenciação automática e aceleração via GPU (CUDA).

O restante deste artigo está organizado da seguinte forma: A Seção II detalha a metodologia de preparação dos dados, com ênfase na prevenção de vazamento de dados. A Seção III descreve a arquitetura proposta e os mecanismos de fusão. A Seção IV apresenta a configuração experimental, incluindo os hiperparâmetros das execuções (Runs) realizadas. A Seção V discute os resultados obtidos e o comportamento da função de perda, seguida pela Conclusão na Seção VI.

2 Trabalhos Relacionados

A aplicação de *Deep Learning* para o diagnóstico automatizado de câncer oral tem recebido atenção crescente devido à alta taxa de mortalidade associada ao diagnóstico tardio. Embora Redes Neurais Convolucionais (CNNs) tenham demonstrado sucesso em tarefas de imagem médica, como na classificação de tumores cerebrais, a adaptação para a cavidade oral apresenta desafios únicos, como a alta variabilidade intra-classe e a escassez de dados anotados de qualidade.

Recentemente, Piyarathne et al. publicaram um conjunto de dados abrangente de imagens da cavidade oral, abordando a lacuna crítica de falta de dados padronizados na literatura. O estudo detalha a coleta e anotação de imagens para diagnóstico de Câncer Oral (OCA) e Distúrbios Potencialmente Malignos (OPMD), fornecendo uma base sólida validada por especialistas para o treinamento de modelos supervisionados. A disponibilidade deste recurso permitiu avançar além das técnicas de processamento de imagem convencionais, habilitando o uso de arquiteturas profundas que requerem grandes volumes de dados.

No contexto de arquiteturas avançadas, Devindi et al. propuseram um *pipeline* multimodal para a detecção precoce de câncer oral. A abordagem dos autores integra CNNs para extração de características visuais com dados de fatores de risco do paciente, demonstrando que a inclusão de informações clínicas melhora significativamente a performance do classificador em comparação com modelos puramente visuais. O trabalho destaca a importância de considerar o histórico do paciente (tabagismo, consumo de álcool e mascar de betel) como variáveis latentes que influenciam a probabilidade da patologia.

O presente trabalho expande essas iniciativas ao adotar uma arquitetura baseada em *backbones* de última geração (*ConvNeXt*), diferindo dos trabalhos anteriores ao implementar um regime de treinamento focado na mitigação rigorosa de vazamento de dados (*Data Leakage*) através de particionamento estrito por paciente e estratégias de otimização bifásicas. Além disso, refinamos a estratégia de multimodalidade através de um mecanismo de fusão tardia com mascaramento de *features* para lesões ausentes.

3 Metodologia: Preparação e Integridade dos Dados

A confiabilidade de um sistema de auxílio ao diagnóstico médico depende intrinsecamente da qualidade e da integridade dos dados utilizados no seu treinamento. Neste trabalho, a metodologia de manipulação de dados foi desenhada não apenas para formatar as entradas para a rede neural, mas principalmente para garantir uma avaliação estatisticamente honesta e clinicamente relevante. Foi utilizado o conjunto de dados "ORALCANCER_DATASET", composto por imagens fotográficas da cavidade oral e metadados clínicos associados. Dada a

natureza privada deste banco de dados, todo o pipeline de processamento foi implementado localmente, sem exposição pública dos dados brutos.

3.1 Engenharia de Particionamento e Prevenção de Vazamento (Data Leakage)

Um erro metodológico crítico e comum em literatura de aprendizado de máquina médica é a divisão aleatória de amostras (*random split*) baseada puramente no número total de imagens. Como o conjunto de dados utilizado contém múltiplas capturas fotográficas (ângulos e iluminações distintos) de um mesmo paciente, uma divisão aleatória simples resultaria em *Data Leakage* (vazamento de dados). Neste cenário, recortes de uma lesão de um paciente P_x poderiam aparecer tanto no conjunto de treinamento quanto no conjunto de testes. A rede neural, sendo um aproximador de funções universal, tenderia a "memorizar" características irrelevantes do paciente P_x (como a estrutura dentária ou pigmentação específica da pele) em vez de aprender as características generalizáveis da patologia. Isso inflaria artificialmente as métricas de acurácia, resultando em um modelo incapaz de generalizar para novos pacientes (*unseen patients*).

Para mitigar este risco, desenvolvemos um algoritmo de particionamento personalizado, implementado no script *build_manifest_and_split.py*. O algoritmo opera nas seguintes etapas:

3.1.1 **Extração de Identificadores Únicos:** O script processa os nomes dos arquivos brutos (ex: C-41-7-2.jpg) para extrair o identificador único do paciente (*patient_id*, ex: C-41). Esta etapa é fundamental para agrupar todas as instâncias visuais de um indivíduo antes de qualquer operação de divisão.

3.1.2 **Agregação por Paciente:** Utilizando a biblioteca **Pandas**, uma tabela pivô (*by_patient*) que contabiliza o número total de imagens por paciente e suas respectivas categorias diagnósticas foi gerada.

3.1.3 **Divisão Estratificada Orientada a Pacientes:** A divisão dos conjuntos de Treino (60%), Validação (20%) e Teste (20%) é realizada sobre a lista de *IDs de pacientes*, e não sobre a lista de imagens. Isso garante o isolamento estrito: se o paciente P_x é alocado para o treinamento, nenhuma imagem dele jamais será vista pelo modelo durante a validação ou teste. Além disso, uma pequena lógica é aplicada para priorizar pacientes com maior quantidade de imagens total para o conjunto de teste, permitindo que classes com menor porcentagem de imagens tenham uma maior quantidade de pacientes.

3.1.4 **Algoritmo de Troca para Cobertura de Classes:** Dada a escassez de casos de Carcinoma Oral (OCA) em relação às classes "Saudável" ou "Benigno", uma divisão puramente aleatória de pacientes poderia resultar em conjuntos de teste sem nenhum exemplo positivo de câncer, inviabilizando o cálculo de métricas como Sensibilidade ou F1-Score para esta classe crítica. Foi implementada uma heurística iterativa que monitora a presença da classe OCA em cada *split*. Caso um conjunto (ex: Validação) não atinja o mínimo de pacientes OCA definidos (configurado como `MIN_OCA_PATIENTS_BY_SPLIT = 1`), o algoritmo identifica um paciente com OCA no conjunto de origem (ex: Teste ou Treino) e realiza uma troca (*swap*) com um paciente de classe não-crítica, garantindo a representatividade de todas as patologias em todas as fases do experimento.

3.1.5 **Distribuição de Imagens por Classe e Conjunto de Dados:** A distribuição quantitativa das imagens resultante do processo de particionamento é detalhada na Tabela I. O conjunto de dados apresenta um desbalanceamento natural severo, com a classe 'OPMD' representando a maioria das amostras, enquanto a classe crítica 'Carcinoma (OCA)' compreende apenas uma pequena fração do total (aprox. 4,3%). Além disso, a variação no número de imagens por paciente reforça a necessidade da estratégia de amostragem ponderada (*WeightedRandomSampler*) adotada durante o treinamento para evitar que o modelo enviesasse suas previsões em direção às classes majoritárias.

Tabela I. Distribuição de Imagens por Classe e Conjunto de Dados

Classe / Categoria	Treino (n)	Validação (n)	Teste (n)	Total (n)
Saudável (Healthy)	438	151	140	729
Benigno (Benign)	392	180	176	748
OPMD	912	233	249	1394
Carcinoma (OCA)	61	36	32	129
TOTAL	1803	600	597	3000

3.2 Extração de Regiões de Interesse (ROI) Baseada em Anotações COCO

As imagens originais da cavidade oral contêm uma quantidade significativa de informação visual não relacionada à patologia (ruído de fundo, lábios, instrumentos médicos). Para focar a atenção da rede neural nas características relevantes, foi criado um pipeline de pré-processamento focado em Regiões de Interesse (ROIs), automatizado pelo script *build_roi_manifest.py*.

Este script consome um arquivo de anotações no formato padronizado **COCO** (*Annotation.json*), que contém as coordenadas das *bounding boxes* (caixas delimitadoras) desenhadas por especialistas. O processo de recorte segue a lógica:

3.2.1 Identificação Semântica: O script mapeia dinamicamente os IDs das categorias "Oral Cavity" (contexto anatômico) e "Lesion" (alvo patológico) presentes no JSON.

3.2.2 Recorte de Contexto (Obrigatório): Para cada imagem, extraímos o recorte da "Oral Cavity". Aplicamos um *padding* (margem de segurança) de 5% a 10% ao redor da caixa delimitadora anotada. Isso previne que bordas importantes da anatomia sejam cortadas inadvertidamente e fornece contexto espacial para o modelo.

3.2.3 Recorte de Lesão (Condicional): Se a imagem contém anotações de lesão, geramos um segundo recorte focado especificamente nela. Nos casos onde múltiplas lesões estão presentes na mesma imagem, implementamos uma função de união geométrica (*union_bboxes*) que calcula o menor retângulo capaz de englobar todas as lesões simultaneamente. Este recorte também recebe *padding* proporcional para capturar a transição entre o tecido saudável e o patológico (borda da lesão). Inclusive, esta abordagem ainda requer validação de eficácia ou embasamento científico. Talvez fosse melhor escolher uma das lesões em caso de múltiplas lesões em uma imagem.

3.2.4 Tratamento de Ausências: Caso a imagem seja de um paciente saudável (classe "Healthy"), o recorte de lesão não é gerado. No pipeline de carregamento (descrito a seguir), esta ausência é tratada inserindo-se um tensor de zeros (imagem preta), sinalizando explicitamente à rede a ausência de patologia focal.

O resultado deste processo é um novo manifesto (*roi_manifest.csv*) que mapeia cada entrada original para seus respectivos recortes processados e metadados, servindo como a "fonte da verdade" para o treinamento.

3.3 Pré-processamento de Dados Tabulares e Clínicos

Paralelamente ao processamento visual, os dados clínicos brutos (Idade, Sexo, Hábitos) foram submetidos a um processo de engenharia de *features* para torná-los compatíveis com redes neurais baseadas em gradiente descendente.

Normalização Numérica: A variável contínua "Idade" foi normalizada utilizando a técnica de *Z-score* $\left(\frac{x-\mu}{\sigma}\right)$, centralizando os dados em zero e escalando-os pela variância. Isso previne que a magnitude numérica da idade (ex: 60 anos) domine os gradientes em relação a variáveis binárias (0 ou 1) nas primeiras camadas da rede.

Codificação Categórica: Variáveis categóricas nominais (Gênero) e binárias (Fumo, Álcool, Mascar Betel) foram convertidas utilizando *One-Hot Encoding*. Por exemplo, a variável "Fumo" foi desdobrada em dois canais de entrada: *smoking_Yes* e *smoking_No*. Embora redundante para árvores de decisão, essa representação densa facilita a convergência em MLPs (*Multilayer Perceptrons*) por fornecer caminhos de ativação ortogonais para cada estado.

3.4. Pipeline de Carregamento e Amostragem Ponderada

A ingestão dos dados para a GPU foi gerenciada pela classe personalizada *OralLesionMultimodalDataset* e orquestrada pelo *DataLoader* do PyTorch.

Um desafio final abordado nesta etapa foi o severo desbalanceamento de classes, onde a quantidade de imagens "Saudáveis" supera largamente a de "OCA" ou "OPMD". O treinamento em dados desbalanceados tipicamente resulta em um modelo enviesado que atinge alta acurácia global simplesmente prevendo a classe majoritária para todos os exemplos. Para combater isso, não foi utilizada amostragem aleatória uniforme padrão. Em vez disso, foi implementado um *WeightedRandomSampler* (Amostrador Aleatório Ponderado).

Este amostrador atribui a cada amostra de treinamento um peso inversamente proporcional à frequência da sua classe no conjunto de dados:

$$W_{classe} = \frac{N_{total}}{N_{classe}}$$

Durante a formação de cada "minibatch", a probabilidade de uma imagem ser selecionada é ditada por esse peso. Consequentemente, o modelo é exposto a exemplos de classes raras (como Câncer) com muito mais frequência do que sua ocorrência natural, simulando um conjunto de dados balanceado estatisticamente durante o cálculo do gradiente, sem a necessidade de descartar dados da classe majoritária (undersampling) ou gerar dados sintéticos (oversampling físico).

4 Arquitetura Multimodal Proposta

Para capturar a complexidade multifacetada do diagnóstico de câncer oral, foi projetado uma Rede Neural Profunda híbrida que opera simultaneamente em domínios visuais e tabulares. A arquitetura, definida na classe *MultiModalMobileNetV3Large* (uma nomenclatura herdada que encapsula uma estrutura agnóstica ao *backbone*), foi implementada utilizando o *framework* PyTorch. O modelo é composto por três ramos (*branches*) de processamento paralelo que convergem para um módulo de fusão tardia (*late fusion*).

4.1 Extração de Características Visuais (Visual Backbones)

A extração de *features* visuais é realizada por dois *backbones* convolucionais independentes: um dedicado ao contexto global da cavidade oral e outro focado no recorte da lesão.

Para garantir flexibilidade e acesso ao estado da arte em visão computacional, foi integrado a biblioteca *timm* (*PyTorch Image Models*) ao pipeline. Foi desenvolvido um módulo invólucro, denominado *_BackboneWrapper*, que automatiza a instanciação de modelos pré-treinados. Este módulo realiza três operações na inicialização da rede:

4.1.1 Remoção do Cabeçalho de Classificação: O parâmetro *num_classes=0* é passado ao construtor do *timm*, descartando as camadas densas finais originais (usadas para classificar as 1000 classes do ImageNet).

4.1.2 Pooling Global: Ativamos o *global_pool='avg'*, que aplica uma operação de *Global Average Pooling* sobre os mapas de características finais. Isso transforma o tensor de saída 4D (B, C, H, W) em um vetor de características contíguo 2D (B, C), onde B é o tamanho do lote (*batch size*) e C é a dimensão do canal de *features* (ex: 768 para o ConvNeXt Tiny).

4.1.3 Inferência Dinâmica de Dimensão: Para evitar a codificação rígida (*hardcoding*) das dimensões de entrada do classificador, implementamos um método *_infer_out_dim* que

executa uma passagem direta (*forward pass*) com um tensor "dummy" (falso) durante a inicialização, capturando automaticamente o tamanho do vetor de saída do *backbone* selecionado.

Nos experimentos relatados neste trabalho, o *backbone* selecionado foi o ConvNeXt Tiny, uma arquitetura convolucional moderna que incorpora princípios de design de *Vision Transformers* (ViTs), oferecendo robustez superior a variações de textura em comparação a CNNs clássicas como a ResNet.

Vale ressaltar que a modularidade do componente *_BackboneWrapper* foi arquitetada para suportar a troca agnóstica de extratores de características. Embora os resultados finais apresentados neste artigo se concentrem na performance do *ConvNeXt Tiny*, a infraestrutura de código suporta nativamente a substituição imediata por qualquer modelo do catálogo do timm através de um único argumento de linha de comando. Durante as etapas preliminares de desenvolvimento, arquiteturas leves como a MobileNetV3 e a EfficientNet-B0 foram integradas e testadas para validar o fluxo de dados multimodal. Embora esses modelos não tenham sido selecionados para as execuções finais (Run 1 e Run 2) devido à superioridade do *ConvNeXt* na captura de texturas finas de lesões, a sua implementação bem-sucedida comprova a flexibilidade do *framework* proposto para cenários onde a eficiência computacional (inferência em dispositivos móveis/edge) seja prioritária sobre a maximização absoluta de métricas.

Um desafio arquitetural específico deste domínio é a variabilidade na entrada: pacientes saudáveis não possuem uma imagem de "lesão". Nesses casos, o dataloader fornece um tensor de zeros (imagem preta). No entanto, passar um tensor de zeros por uma CNN profunda gera um vetor de características ruidoso (ativações não nulas devido aos vieses/biases das convoluções), o que poderia confundir o classificador.

Para resolver isso, foi implementado um mecanismo de "gating" (portão) no método forward. O modelo recebe um tensor auxiliar binário *has_lesion*. Multiplicamos elementarmente o vetor de características da lesão por este tensor:

$$F_{lesion} = F_{lesion} \otimes M_{has_lesion}$$

Dessa forma, se $M_{has_lesion} = 0$, o vetor de características visual da lesão é forçado a ser matematicamente zero, eliminando qualquer ruído processual antes da etapa de fusão.

4.2 Processamento de Dados Clínicos (Ramo Tabular)

Os dados clínicos pré-processados (vetor de dimensão $N_{tab} = 9$, contendo idade normalizada e variáveis *one-hot*) são processados por um *Perceptron* Multicamadas (MLP) dedicado. Este ramo, definido como *self.tab_mlp*, consiste em uma projeção linear que mapeia as *features* de entrada para um espaço latente de dimensão 64, seguida por uma camada de Batch Normalization unidimensional e uma ativação não-linear ReLU (*Rectified Linear Unit*). A normalização em lote (*Batch Norm*) neste estágio é crucial para estabilizar a distribuição das ativações tabulares, garantindo que elas tenham uma escala comparável às características visuais extraídas pelos *backbones*.

4.3 Fusão Multimodal e Classificação

A integração das modalidades ocorre através de uma estratégia de concatenação de vetores (*feature concatenation*). Os vetores de características da cavidade oral (D_{oral}), da lesão (D_{lesion}) e dos dados tabulares (D_{tab}) são unidos ao longo da dimensão do canal, resultando em um vetor de fusão V_{fusion} com dimensão total $D_{total} = D_{oral} + D_{lesion} + D_{tab}$.

Este vetor combinado alimenta o módulo *self.fusion_mlp*, uma rede densa projetada para aprender as correlações não-lineares cruzadas entre a aparência visual e o perfil de risco do paciente. A estrutura deste módulo segue o padrão:

4.3.1 Linear ($D_{total} \rightarrow 256$) \rightarrow BatchNorm \rightarrow ReLU \rightarrow Dropout

4.3.2 Linear (256 \rightarrow 256) \rightarrow BatchNorm \rightarrow ReLU \rightarrow Dropout

O uso de Dropout (com probabilidade p configurável, tipicamente 0.3, 0.4 ou 0.5) nestas camadas é fundamental para prevenir o sobreajuste (*overfitting*), forçando a rede a não depender excessivamente de uma única modalidade de entrada. Finalmente, uma camada linear de projeção (*self.classifier*) mapeia o vetor de 256 dimensões para os 4 *logits* correspondentes às classes de saída (Saudável, Benigno, OPMD, OCA).

5 Configuração Experimental

Para avaliar a eficácia da arquitetura multimodal proposta, conduzimos uma série de experimentos controlados. O objetivo central não foi apenas maximizar as métricas de desempenho, mas investigar como diferentes regimes de ajuste fino (*fine-tuning*) e escalonamento de taxa de aprendizado impactam a convergência do modelo em um cenário de dados médicos escassos e desbalanceados.

5.1 Protocolo de Treinamento Comum

Todos os experimentos foram executados num ambiente computacional padronizado baseado em linux para garantir a reprodutibilidade. O código foi desenvolvido em Python 3.10 sobre o *framework* PyTorch 2.0, com aceleração via GPU (CUDA). Para assegurar a determinística dos resultados, fixamos a semente aleatória (*seed*) em 42 para todas as bibliotecas estocásticas (*numpy*, *random*, *torch*).

5.1.1 **Otimizador:** Foi utilizado o algoritmo AdamW (*Adam with Weight Decay*), escolhido pela sua capacidade de desacoplar o decaimento de peso da otimização do gradiente, para evitar o sobreajuste em modelos profundos.

5.1.2 **Função de Perda (Loss Function):** Dada a natureza crítica dos erros de classificação em oncologia, empregamos a CrossEntropyLoss Ponderada. Calculamos um vetor de pesos W_c para as classes, onde o peso de cada classe é inversamente proporcional à sua frequência no conjunto de treino. Na Run 1, os pesos manuais foram definidos como [2.11, 4.60, 4.98, 2.0] para as classes *Healthy*, *Benign*, *OPMD* e *OCA*, respetivamente. Isso penaliza o modelo severamente por erros nas classes minoritárias (Benigno e OPMD).

5.1.3 **Amostragem Estratégica:** No nível do *DataLoader*, integramos um *WeightedRandomSampler*. Ao contrário da iteração sequencial ou aleatória simples, este amostrador consulta os metadados de frequência de classes e constrói cada *minibatch* (tamanho 16) garantindo que, estatisticamente, o modelo veja uma distribuição uniforme de patologias, mitigando o viés natural do *dataset*.

5.1.4 **Critério de Seleção de Modelo (Checkpointing):** É bom salientar que a avaliação final no conjunto de teste não utilizou os pesos da última época de treinamento (época 20), mas sim um *checkpoint* do "Melhor Modelo" (*best_model.pt*) salvo automaticamente. O algoritmo de treino monitora o F1-Score Macro no conjunto de validação ao final de cada época; sempre que este valor supera o recorde histórico, o estado atual dos pesos é serializado em disco. Esta estratégia atua como um mecanismo de *Early Stopping* virtual, assegurando que o modelo implantado seja aquele com maior capacidade de generalização, evitando o uso de versões que possam ter sofrido sobreajuste (*overfitting*) nos estágios finais do treinamento. Especificamente, a análise dos *logs* revela que os resultados finais da Run 1 foram gerados pelos pesos da época 14 (F1 de Validação: 0.7368), enquanto a performance superior da Run 2 derivou, curiosamente, da época 7 (F1 de Validação: 0.7527), indicando que a melhor fronteira de decisão foi encontrada antes mesmo da transição agressiva da segunda fase de otimização.

5.2 Experimento 1 (Run 1): Baseline com Descongelamento Progressivo

O primeiro cenário experimental (Run 1) foi desenhado para estabelecer uma linha de base robusta, seguindo a prática padrão de *Transfer Learning* progressivo.

Hiperparâmetros: O treino decorreu por 20 épocas com uma taxa de aprendizado (*Learning Rate* - LR) fixa e conservadora de 1×10^{-5} e decaimento de peso de 1×10^{-3} .

Estratégia de Congelamento (Freeze Schedule): Adotou-se uma abordagem de "Aquecimento do Classificador". Nas primeiras 10 épocas, o *backbone* visual (ConvNeXt Tiny) permaneceu totalmente congelado (pesos imutáveis), permitindo que apenas os módulos MLP (tabular e fusão) e o classificador final ajustassem os seus pesos aleatórios iniciais.

Descongelamento Parcial: A partir da época 11, aplicou-se a técnica de *backbone_unfreeze_mode='partial'* com fração 0.3. Isso significa que apenas os últimos 30% das camadas do *backbone* foram descongelados para treino, mantendo as camadas iniciais (responsáveis por detectar arestas e texturas básicas) intactas. Esta estratégia visa adaptar as representações semânticas de alto nível ao domínio oral sem destruir os filtros genéricos aprendidos na ImageNet.

5.3 Experimento 2 (Run 2): Otimização Bifásica com Congelamento Explícito

O segundo cenário (**Run 2**) explorou uma estratégia de otimização mais agressiva e não-linear, dividindo o treino em duas fases distintas controladas por janelas de congelamento explícitas e LRs dinâmicos.

Fase 1 (Épocas 1-10): Refinamento de Features.

Nesta fase inicial, o modelo operou com uma taxa de aprendizado baixa (1×10^{-5}) mas com o backbone parcialmente descongelado desde o início (*unfreeze_fraction=0.7*). Ao permitir que 70% da rede visual fosse treinada imediatamente, objetivou-se uma adaptação rápida das features profundas ao domínio específico das lesões.

Fase 2 (Épocas 11-20): Consolidação do Classificador.

A partir da época 11, introduziu-se uma mudança de regime radical. O backbone foi forçado a um estado congelado (*freeze_epoch_start=11, freeze_epoch_end=20*). Simultaneamente, a taxa de aprendizado foi aumentada em uma ordem de grandeza para 1×10^{-4} .

A hipótese técnica por trás desta configuração "invertida" é permitir que, uma vez que as features visuais estejam estavelmente adaptadas (Fase 1), o otimizador possa focar exclusivamente na fronteira de decisão do classificador (MLP) com passos de gradiente maiores (Fase 2), sem o risco de degradar a qualidade da extração de características visuais devido a oscilações. O *Dropout* foi ajustado para 0.4 e os pesos de classe foram ligeiramente recalibrados para [2.0, 6.0, 6.0, 4.0], com um decaimento de peso ajustado para $1e-4$ para permitir um ajuste mais fino dos parâmetros, aumentando a penalidade para erros em OPMD e Benignos.

5.4 Regularização L2 (Weight Decay):

Para mitigar o risco de *overfitting* inerente a modelos profundos treinados em conjuntos de dados de tamanho moderado, aplicou-se Regularização L2 em ambas as execuções (*runs*). Implementada através do parâmetro de decaimento de pesos (*weight decay*) do otimizador AdamW, esta técnica adiciona um termo de penalidade à função de perda proporcional ao quadrado da magnitude dos pesos da rede ($\|w\|^2$). Ao forçar os pesos a permanecerem pequenos, a regularização L2 restringe a complexidade efetiva do modelo e impede que a rede dependa excessivamente de um pequeno conjunto de neurônios para tomar decisões, promovendo o aprendizado de padrões mais robustos e distribuídos. Os coeficientes específicos utilizados () foram ajustados individualmente para cada experimento (detalhados nas seções seguintes).

5.5. Recursos Computacionais e Reprodutibilidade

Para garantir a reprodutibilidade dos resultados e estabelecer uma base de custo computacional para trabalhos futuros, todos os experimentos foram conduzidos em um ambiente linux de hardware padronizado. O treinamento dos modelos foi acelerado utilizando uma única GPU NVIDIA QUADRO RTX 5000 com 16384 MiB de memória dedicada. O ambiente de software baseou-se no framework PyTorch versão 2.0 com suporte a CUDA versão 12.6.

Em termos de eficiência temporal, o tempo médio de treinamento por época para a arquitetura multimodal proposta foi de aproximadamente 3 minutos, utilizando 8 *workers* paralelos no *DataLoader* para mitigar gargalos de I/O durante a leitura das imagens e recortes. O código fonte completo, incluindo os *scripts* de particionamento e os pesos dos modelos treinados, foi estruturado para permitir a re-execução determinística dos experimentos mediante a fixação das sementes aleatórias (*seed*=42).

6 Resultados e Discussão

A avaliação quantitativa dos modelos propostos baseou-se num conjunto de teste independente, rigorosamente isolado (conforme descrito na Seção II), contendo pacientes nunca vistos durante as fases de treino ou validação. As métricas primárias selecionadas para a análise foram a Acurácia Global e, mais criticamente, o F1-Score Macro. Dada a natureza desbalanceada do problema (onde a classe "Saudável" é predominante), a Acurácia isolada pode ser enganosa; o F1-Macro, sendo a média harmônica não ponderada da precisão e revocação de todas as classes, oferece uma visão mais honesta da capacidade do modelo em detectar patologias raras como o Carcinoma Oral (OCA).

6.1 Análise do Experimento 1: Baseline com Descongelamento Progressivo

O primeiro cenário experimental (Run 1) demonstrou um comportamento de convergência estável e consistente, típico de abordagens de *Transfer Learning* bem comportadas. O modelo iniciou o treinamento com uma Acurácia de Validação de aproximadamente 43% na época 1, evoluindo rapidamente para a casa dos 70% na época 5, à medida que o classificador MLP se adaptava às *features* estáticas do *backbone* congelado.

Um ponto de inflexão notável ocorreu na época 11, momento programado para o descongelamento parcial (*unfreeze_fraction*=0.3) do *backbone* ConvNeXt Tiny. Observamos um salto imediato nas métricas de treinamento (Acurácia subindo de 72% para 77%), indicando que a libertação dos pesos das camadas profundas permitiu ao modelo refinar as suas representações semânticas para as especificidades das lesões orais.

Ao final das 20 épocas, a Run 1 atingiu no conjunto de Teste uma Acurácia de 80.74% e um F1-Macro de 0.7402. Estes valores estabelecem uma linha de base sólida, validando que a arquitetura multimodal é capaz de generalizar eficazmente mesmo com uma estratégia de otimização linear e conservadora.

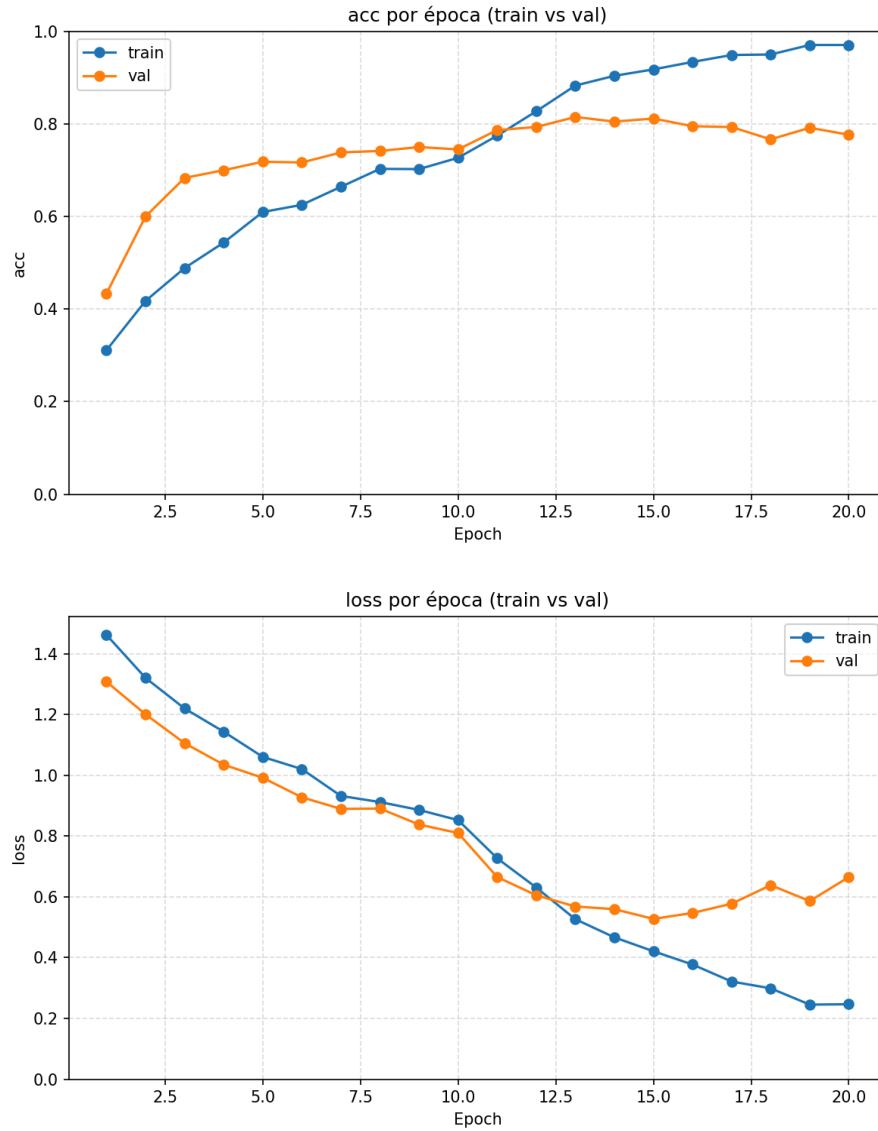


Fig. 1. Evolução da Acurácia (esq./cima) e Perda (dir./baixo) ao longo de 20 épocas. Observa-se uma convergência estável, mas lenta, característica da taxa de aprendizado fixa. O descongelamento parcial na época 11 gera um ganho visível de desempenho.

6.2 Análise do Experimento 2: Otimização Bifásica e Congelamento Explícito

O segundo experimento (Run 2) apresentou uma dinâmica de aprendizado distinta, caracterizada pela intervenção agressiva nos hiperparâmetros na metade do ciclo de treino. Durante a Fase 1 (Épocas 1-10), onde o *backbone* operou parcialmente descongelado com uma taxa de aprendizado (LR) baixa ($1e - 5$), o modelo focou na extração de *features* finas. É notável que, já na época 9, o modelo atingiu uma acurácia de treino de 98.67%, sugerindo uma rápida adaptação visual.

A transição para a Fase 2 (Épocas 11-20) introduziu um "choque" de otimização: o congelamento forçado do *backbone* simultâneo a um aumento de 10x na taxa de aprendizado ($1e - 4$). O objetivo desta estratégia era consolidar as fronteiras de decisão do MLP sem arriscar a degradação das *features* visuais já aprendidas. Os resultados confirmam a eficácia desta abordagem: a Run 2 superou a baseline, atingindo no Teste uma Acurácia de 81.57% e um F1-Macro de 0.7695. Este ganho de aproximadamente 3 pontos percentuais no F1-Score é interessante em contextos médicos, indicando uma melhor discriminação entre classes difíceis.

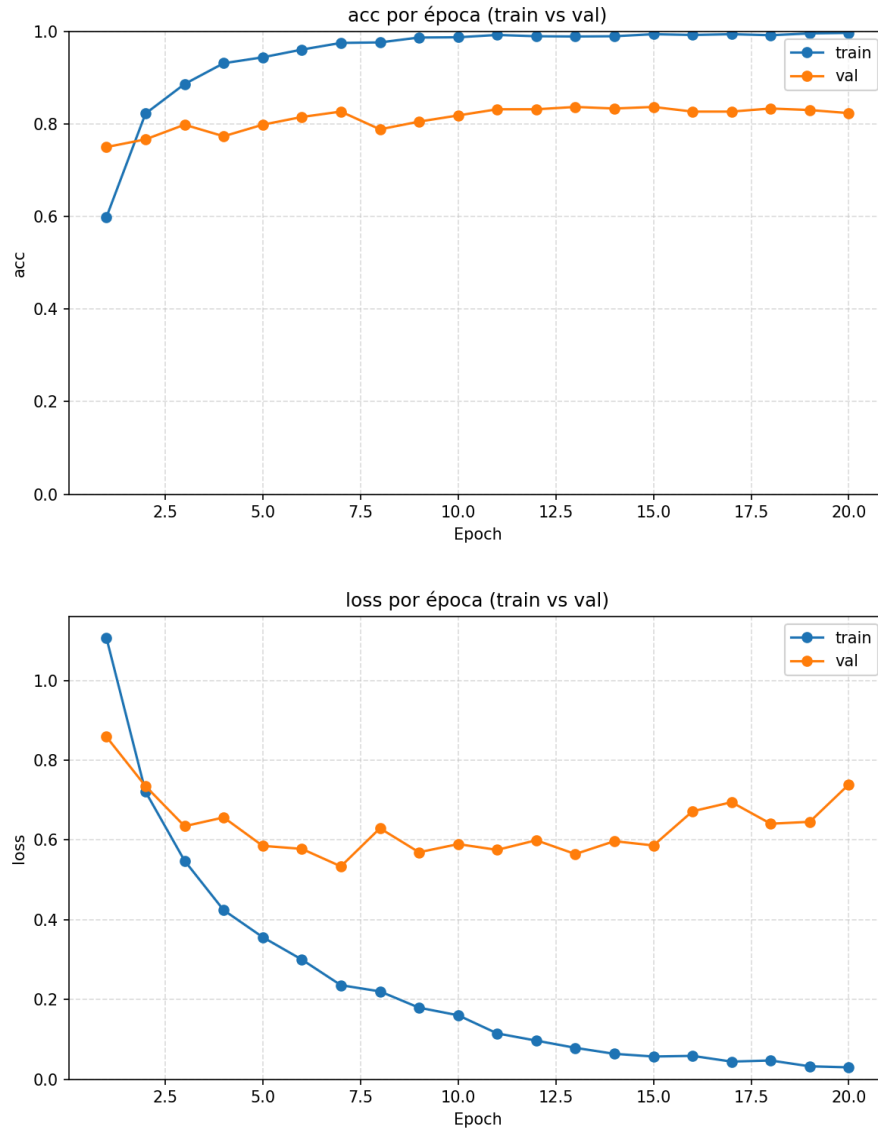


Fig. 2. A estratégia de congelamento explícito combinada com aumento de LR na época 11 (Fase 2) resultou em uma redução da Loss de treino e estabilização da Acurácia de validação em patamares superiores à baseline.

6.3 Análise Detalhada das Confusões e Classes Críticas

A análise das Matrizes de Confusão de ambos os experimentos revela padrões consistentes que corroboram a robustez da arquitetura proposta, independentemente da estratégia de treino.

6.3.1 Classe Saudável: O modelo demonstrou um desempenho perfeito na identificação de tecido saudável. Na Run 2, por exemplo, de 140 amostras de teste da classe "Healthy", o modelo classificou corretamente todas as 140, sem nenhum falso positivo ou negativo para esta classe. Isso é bom para um sistema de triagem, garantindo que pacientes saudáveis não sejam submetidos a procedimentos desnecessários, mas o importante é sua capacidade de identificar casos de lesões malignas acima dos saudáveis.

6.3.2 A Fronteira Benigno vs. OPMD: A maior fonte de erro do sistema reside na distinção entre lesões "Benignas" e "OPMD" (Desordens Potencialmente Malignas). Visualmente, estas classes podem compartilhar características morfológicas muito similares (e.g., leucoplasias sutis vs. hiperqueratose friccional). Na Run 2, o modelo confundiu 45 casos de Benigno como OPMD e 44 casos de OPMD como Benigno. Este comportamento é

esperado em sistemas de visão computacional que não dispõem de informações histopatológicas (biópsia), mas é preocupantemente indesejado no domínio da saúde.

6.3.3 Carcinoma Oral (OCA): Para a classe mais crítica, o Câncer, o uso do *WeightedRandomSampler* e da *CrossEntropy* Ponderada mostrou-se fundamental. Apesar de ser a classe com menor número de exemplos absolutos, o modelo conseguiu manter uma taxa de detecção relevante, com os erros confinados majoritariamente à confusão com OPMD (o que é clinicamente menos grave do que confundir com Saudável, pois ambos exigem acompanhamento).

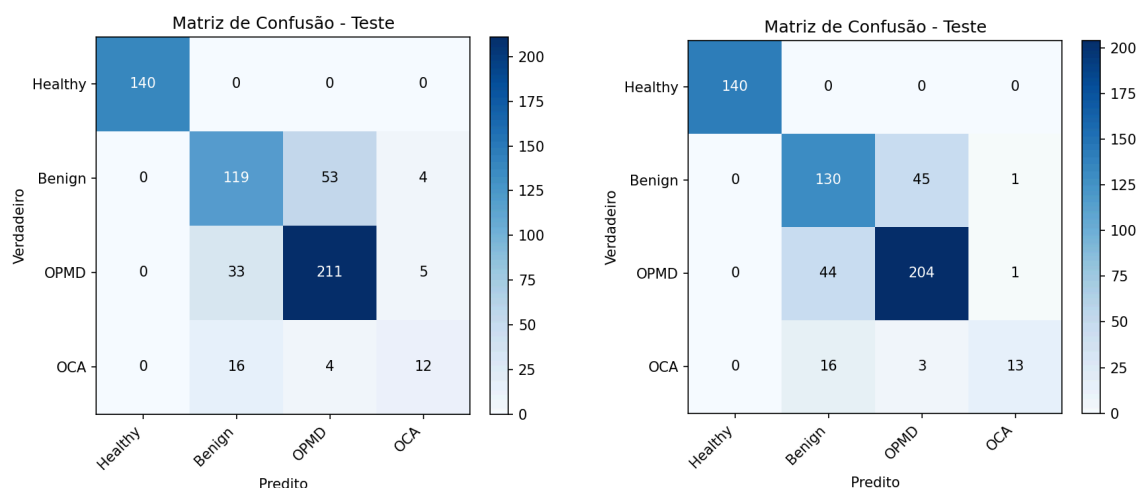


Fig. 3. Detalhamento dos acertos e erros por classe. Destaca-se a perfeição na classificação de "Saudável" (140/140) e a concentração de erros na distinção visual entre "Benigno" e "OPMD".

6.4 Divergência entre Perda (Loss) e Acurácia: Uma Perspectiva de Otimização

Uma observação recorrente nos gráficos de treinamento de ambas as execuções é o comportamento da Validation Loss (Perda de Validação). Enquanto a Acurácia de Validação se mantém estável ou sobe (atingindo ~83% na Run 2), a Loss de Validação não acompanha essa queda na mesma proporção, estabilizando-se em valores absolutos altos (em torno de 0.6 - 0.7).

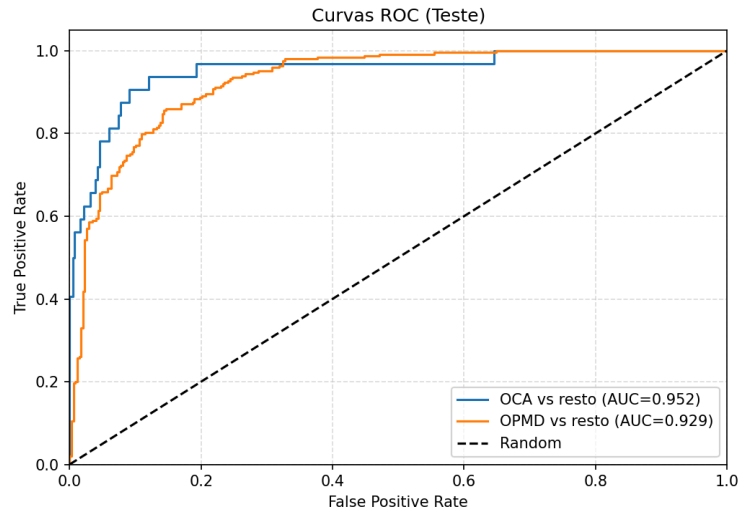
Não necessariamente este fator indica má qualidade dos dados ou rotulagem incorreta no conjunto de validação. Parece que este comportamento é intrínseco à natureza da função de custo CrossEntropyLoss quando aplicada a redes neurais profundas em regimes de *fine-tuning*, mas mais testes precisam ser feitos para ter certeza, demarca inclusive uma limitação deste projeto.

6.5 A métrica de Acurácia:

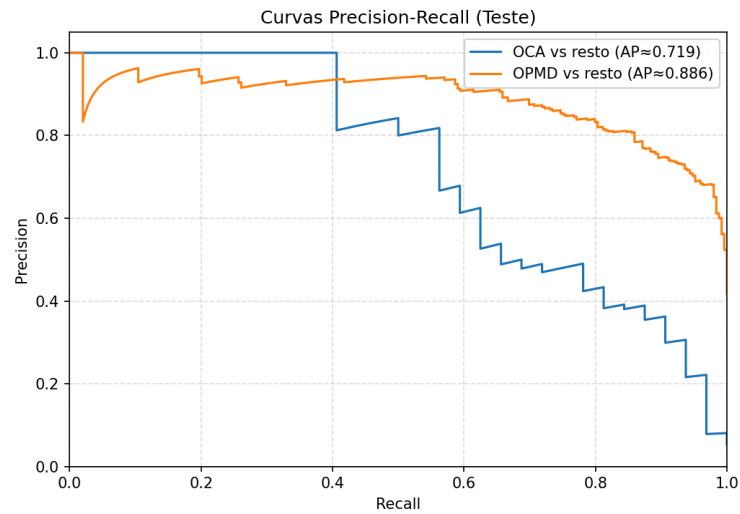
Para métrica de acurácia importa apenas se a classe com a maior probabilidade (argmax) é a correta. Já a *Loss* é contínua e penaliza a incerteza. Durante o treinamento, a rede é incentivada a aumentar a magnitude dos *logits* para a classe correta, levando a probabilidades próximas de 1.0 (100% de confiança) e uma Training Loss próxima de zero (e.g., 0.0298 na época 20 da Run 2). No entanto, no conjunto de validação, mesmo quando a rede acerta a classe (contribuindo para a acurácia), ela pode fazê-lo com uma confiança menor (e.g., 70% de certeza em vez de 99%). Matematicamente, a CrossEntropy penaliza logaritmicamente essa "falta de confiança extrema", mantendo a Loss elevada. Além disso, os poucos erros cometidos (como as confusões Benigno/OPMD) são penalizados severamente se o modelo estiver "superconfiante" (*overconfident*) na classe errada, o que é um efeito colateral da otimização agressiva usada na Fase 2. Portanto, a Loss alta pode refletir a calibração de probabilidade do modelo, e não a sua capacidade discriminativa ou a qualidade do *dataset*.

fornecido, e poderia ser melhorada com mais tempo de trabalho e ajustes, além do conhecimento do engenheiro.

Além das métricas pontuais, foi avaliado a robustez das probabilidades preditas através das Curvas ROC (*Receiver Operating Characteristic*) e Precision-Recall (PR) para as classes críticas.



A)



B)

Fig. 4. (A) Curvas ROC ilustrando a relação Sensibilidade vs. Especificidade. (B) Curvas Precision-Recall, fundamentais para validar a performance em classes desbalanceadas (OCA), mostrando a manutenção da precisão mesmo em altas taxas de recuperação.

As curvas ROC geradas para a **Run 2** demonstram a capacidade do modelo em manter altas taxas de verdadeiro positivo (Sensibilidade) mesmo sob limiares de decisão rigorosos, evidenciando uma separabilidade linear robusta no espaço de *features* multimodal. Paralelamente, as curvas Precision-Recall são importantes neste contexto desbalanceado; elas confirmam que o alto F1-Score obtido para a classe OCA não é fruto do acaso, mas sim de uma precisão sustentada à medida que o recall aumenta.

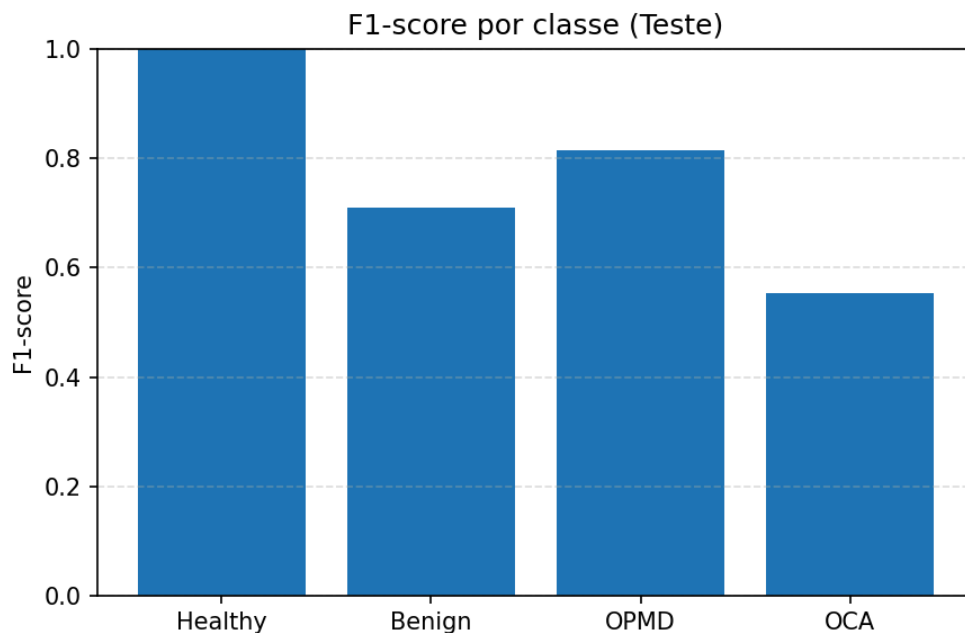


Fig. 5. Comparativo visual do desempenho harmônico. A barra inferior na classe OCA visualiza a dificuldade do modelo em generalizar características visuais ambíguas desta categoria específica.

Por fim, a análise visual do F1-Score por classe isola visualmente a performance superior nas classes extremas ("Saudável" e "Benigno"), enquanto quantifica graficamente o desafio persistente na classe intermediária "OPMD", orientando futuros esforços de ajustes para focar especificamente nesta categoria limítrofe.

6.6 Dinâmica Temporal de Convergência e o Fenômeno da "Fase 1"

Uma análise cronológica detalhada dos logs de treinamento da Run 2 revela um comportamento contra-intuitivo, porém instrutivo, sobre a dinâmica de otimização em duas fases. Embora o experimento tenha sido desenhado com uma segunda fase agressiva (Épocas 11-20, com alta taxa de aprendizado e congelamento do backbone) visando refinar a fronteira de decisão do classificador, os dados demonstram que o pico de generalização foi atingido precocemente.

O checkpoint de "Melhor Modelo", que gerou os resultados finais no conjunto de teste, foi salvo na época 7, registrando um F1-Macro de validação de 0.7527. Este momento pertence à Fase 1 do protocolo, onde o modelo operava com uma taxa de aprendizado conservadora ($1e-5$) e o backbone parcialmente descongelado.

Este achado pode sugerir que a adaptação das features visuais profundas (permitida pelo descongelamento parcial) foi o fator determinante para a performance do modelo, ocorrendo de forma rápida e eficiente nas primeiras iterações. A transição para a Fase 2 (a partir da época 11), embora tenha sido eficaz em minimizar a Perda de Treinamento (reduzindo-a de 0.11 para 0.02) e elevar a Acurácia de Treino para níveis próximos de 100% (overfitting no treino), não se traduziu em ganhos adicionais de generalização na validação. Pelo contrário, observou-se uma leve degradação no F1-Macro de validação nas épocas finais (caindo para 0.69 na época 20), indicando que a "fronteira de decisão ótima" já havia sido estabelecida antes da intervenção agressiva do otimizador. Isso valida a importância do mecanismo de Early Stopping (via checkpointing) e evidencia que, para este domínio, a qualidade da extração de características visuais supera a necessidade de um ajuste fino exaustivo do classificador final.

7 Conclusão

Este trabalho apresentou o desenvolvimento e a avaliação de uma arquitetura de *Deep Learning* multimodal para a classificação de lesões orais, integrando dados visuais (contexto e ROI) e tabulares (clínicos). A abordagem proposta pode superar as limitações de classificadores visuais puros ao incorporar metadados do paciente no processo de decisão, atingindo um F1-Score Macro de aproximadamente 0.76 e Acurácia de 81.57% no conjunto de teste.

A comparação entre as duas estratégias de treinamento revelou que o controle fino sobre a dinâmica de otimização é tão importante quanto a escolha da arquitetura. A Run 2, ao implementar uma estratégia de "congelamento explícito" (*explicit freeze window*) combinada com um escalonamento de taxa de aprendizado bifásico, permitiu que o classificador (MLP) consolidasse suas fronteiras de decisão sem desestabilizar as *features* visuais aprendidas, resultando em uma generalização levemente superior à abordagem linear da Run 1, mas com uma loss excessivamente alta.

7.1 Desafios de Implementação e Escolha de Framework

A implementação deste complexo pipeline evidenciou o *trade-off* entre controle e simplicidade no desenvolvimento de Inteligência Artificial. A opção pelo uso do PyTorch em detrimento de frameworks de mais alto nível, como Keras/TensorFlow, introduziu uma curva de complexidade e aprendizagem significativa. A necessidade de escrever manualmente os loops de treinamento (*run_one_epoch*), gerenciar o estado dos dispositivos (CUDA/CPU) e implementar a lógica de retropropagação (*loss.backward()*, *optimizer.step()*) exigiu um esforço de engenharia substancialmente maior.

No entanto, essa verbosidade foi indispensável para implementar as lógicas customizadas deste projeto, especificamente:

Controle Dinâmico de Congelamento: Talvez a lógica de congelar/descongelar o *backbone* baseada em janelas de épocas específicas (*freeze_epoch_start*, *freeze_epoch_end*) seria complexa de implementar via *callbacks* padrão do Keras.

Amostragem Ponderada: A integração nativa do *WeightedRandomSampler* com o *DataLoader* ofereceu uma solução robusta para o desbalanceamento de classes, operando no nível do *minibatch* antes mesmo dos dados chegarem à GPU.

7.2 Limitações e Trabalhos Futuros

Apesar dos resultados promissores em acurácia e F1-Score, foi observado uma divergência persistente na *Validation Loss*. Embora matematicamente esperada em regimes de *CrossEntropy*, valores absolutos elevados de perda sugerem que a calibração de probabilidade do modelo pode ser refinada. Com maior disponibilidade de tempo computacional e cronograma estendido para experimentação, ajustes adicionais focariam em técnicas de regularização mais fortes (como *Label Smoothing* ou *MixUp*) e na exploração de *schedulers* de taxa de aprendizado cíclicos (*Cosine Annealing with Warm Restarts*) entre outras abordagens para tentar reduzir o hiato entre a confiança do treino e da validação.

7.3 Contribuição de Engenharia: Um Esqueleto Reutilizável

Mais do que um classificador específico para câncer oral, o código desenvolvido neste projeto foi arquitetado como um *framework* genérico e modular para problemas multimodais. A classe *_BackboneWrapper* abstrai a complexidade de instanciar diferentes modelos visuais via biblioteca *timm*, enquanto a classe *MultiModalMobileNetV3Large* desacopla a lógica de fusão da extração de características bem como possibilita a utilização de outros modelos pre-treinados como o *MobileNetV3-Large*, que também foi testado neste projeto, mas não contemplado nas análises finais. Esta estrutura flexível permite que o mesmo esqueleto de

código seja reutilizado em futuros projetos que envolvam a combinação de imagens e dados tabulares, bastando ajustar as dimensões de entrada e o manifesto de dados, consolidando uma ferramenta robusta para pesquisa e desenvolvimento em Inteligência Artificial aplicada.

Referências

1. Piyrathne NS, Liyanage SN, Rasnayaka RMSGK, Hettiarachchi PVKS, Devindi GAI, Francis FBAH, Dissanayake DMDR, Ranasinghe RANS, Pavithya MBD, Nawinne I, Ragel RG, Jayasinghe RD (2024) A comprehensive dataset of annotated oral cavity images for diagnosis of oral cancer and oral potentially malignant disorders. *Oral Oncol* 156:106946. <https://doi.org/10.1016/j.oraloncology.2024.106946>
2. Devindi GAI, Dissanayake DMDR, Liyanage SN, Francis FBAH, Pavithya MBD, Piyrathne NS, Hettiarachchi PVKS, Rasnayaka RMSGK, Jayasinghe RD, Ragel RG, Nawinne I (2024) Multimodal Deep Convolutional Neural Network Pipeline for AI-Assisted Early Detection of Oral Cancer. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2024.3454338>
3. Schiavon DEB, Botelho VR, Pianoski TA, Becker CDL (2023) Interpreting Convolutional Neural Networks for Brain Tumor Classification: An Explainable Artificial Intelligence Approach. In: *Intelligent Systems: 12th Brazilian Conference, BRACIS 2023, Montes Claros, Brazil, September 25–29, 2023, Proceedings, Part I*. Springer, Cham, pp 1–15
4. Felipe CS, Alva TAP, Winck AT, Becker CDL (2023) An Approach in Brain Tumor Classification: The Development of a New Convolutional Neural Network Model. In: *Intelligent Systems: 12th Brazilian Conference, BRACIS 2023, Montes Claros, Brazil, September 25–29, 2023, Proceedings, Part I*. Springer, Cham, pp 16–30
5. Liu Z, Mao H, Wu C-Y, Feichtenhofer C, Darrell T, Xie S (2022) A ConvNeXt for the 2020s. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp 11976–11986
6. Wightman R (2019) *PyTorch Image Models*. <https://github.com/rwightman/pytorch-image-models>. Accessed 24 Nov 2025
7. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp 8024–8035