Universidade Federal do Rio Grande do Norte
Centro de Tecnologia - CT
Escola de Ciência e Tecnologia - ECT

Trabalho em GRUPO – Classificação de imagens com CNN - Unidade 2
DCA3606 - INTELIGÊNCIA ARTIFICIAL - T01 (2025.1)

WERBERT ARLES DE SOUZA BARRADAS Matrícula: 20240002606
HÉLIO ARRUDA CÂMARA NETO Matrícula: 20220079758
LUCAS AUGUSTO SPINOLA PINTO: 20210087460
MARCELO DOS SANTOS VIEIRA: 20190075054

Docente: Professor Mestre MICAEL BALZA

Natal, 15 de junho de 2025

Resumo

Este trabalho propõe a classificação automática de imagens de raio-X torácico em três categorias (normal, pneumonia viral e pneumonia bacteriana) utilizando uma Rede Neural Convolucional (CNN), onde o pré-processamento inclui redimensionamento e normalização das imagens de diferentes tamanhos, além da aplicação de técnicas de data augmentation para aumentar a variabilidade do dataset e evitar overfitting. A CNN é treinada e avaliada por meio de métricas como acurácia, loss, precisão e recall, com análise do desempenho entre conjuntos de treino e validação para verificar a generalização do modelo, enquanto se discute o potencial impacto clínico dessa abordagem e possíveis melhorias, como o uso de transfer learning ou arquiteturas mais avançadas, visando auxiliar no diagnóstico médico de forma eficiente e precisa.

Palavras-chave: CNN, raio-X, pneumonia, classificação de imagens, deep learning, diagnóstico médico.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

A pneumonia continua sendo um dos maiores desafios na saúde global, figurando como principal causa de mortalidade infantil em diversos países. Sua identificação precoce e precisa é fundamental para o tratamento adequado, especialmente na distinção entre os casos virais e bacterianos, que demandam abordagens terapêuticas distintas. A radiografia torácica permanece como o exame mais acessível e amplamente utilizado para o diagnóstico, porém sua interpretação apresenta limitações que vão desde a subjetividade na análise até a escassez de especialistas em regiões remotas.

Este trabalho se propõe a desenvolver um sistema automatizado que possa auxiliar os profissionais de saúde na classificação de radiografias torácicas em três categorias distintas: exames normais, casos de pneumonia viral e pneumonia bacteriana.

Para garantir a eficácia do modelo, serão implementadas diversas etapas cruciais de processamento. As imagens, que apresentam variações naturais em tamanho e qualidade, passarão por um rigoroso pré-processamento envolvendo redimensionamento padronizado e normalização dos valores de pixel. Técnicas de aumento de dados (data augmentation) serão empregadas para enriquecer o conjunto de treinamento, simulando diferentes condições de captura e posicionamento que podem ocorrer na prática clínica.

A avaliação do desempenho do modelo considerará não apenas a acurácia geral, mas também métricas específicas como sensibilidade e valor preditivo positivo, particularmente importantes no contexto médico. Será dada atenção especial ao diagnóstico de superajuste (overfitting), comparando sistematicamente o desempenho nos conjuntos de treinamento e validação. Os resultados obtidos poderão abrir caminho para futuras melhorias, incluindo a incorporação de modelos pré-treinados e técnicas mais avançadas de aprendizado profundo, sempre com o objetivo final de fornecer uma ferramenta confiável para auxiliar os profissionais de saúde no diagnóstico desta importante patologia.

2 Objetivo

Este trabalho tem como propósito principal proporcionar um aprendizado prático sobre a aplicação de Redes Neurais Convolucionais (CNNs) para classificação de imagens médicas. Utilizando como estudo de caso a classificação de radiografias torácicas em três categorias (normais, pneumonia viral e bacteriana), busca-se desenvolver competências fundamentais no uso de inteligência artificial para processamento de imagens.

Os objetivos específicos concentram-se em quatro eixos principais de aprendizado. Primeiramente, visa-se dominar as técnicas de pré-processamento de imagens, incluindo redimensionamento, normalização e aplicação de data augmentation para melhorar a generalização dos modelos. Em segundo lugar, o trabalho abordará o desenvolvimento e compreensão de arquiteturas CNN, desde sua estrutura básica até os processos de treinamento e ajuste de hiperparâmetros.

Um terceiro eixo enfoca a avaliação de modelos, com ênfase na interpretação de métricas de desempenho, identificação de overfitting. Por fim, o trabalho contempla o desenvolvimento de habilidades práticas em frameworks como TensorFlow/Keras, incluindo visualização de resultados e documentação de experimentos.

3 Referencial Teórico

3.1 Histórico e Evolução das CNNs

O desenvolvimento das **Redes Neurais Convolucionais (CNNs)** é um testemunho da inspiração biológica e do avanço tecnológico. Sua gênese remonta ao trabalho seminal de **Kunihiko Fukushima** em 1980, com o modelo **Neocognitron** (FUKUSHIMA, 1980). Este modelo pioneiro, inspirado na organização do córtex visual de mamíferos, introduziu conceitos fundamentais como as camadas hierárquicas de processamento e a capacidade de reconhecimento de padrões independentemente da sua posição no campo visual. Embora o Neocognitron já contivesse elementos essenciais das CNNs modernas, a limitação computacional da época restringiu sua aplicação prática e o reconhecimento de seu potencial.

Um avanço significativo ocorreu com o trabalho de **Yann LeCun** e colaboradores, que, em 1989, aplicaram o algoritmo de retropropagação a uma arquitetura semelhante ao Neocognitron para o reconhecimento de dígitos manuscritos, culminando no desenvolvimento do **LeNet-5** na década de 1990 (LECUN et al., 1998). Este foi um dos primeiros exemplos de uma CNN "treinável" e eficaz para tarefas do mundo real, demonstrando a viabilidade das redes convolucionais em aplicações práticas como o reconhecimento de caracteres em cheques bancários.

Contudo, foi apenas nas últimas décadas, impulsionado pela disponibilidade de grandes **bancos de dados de imagens digitais** (como o ImageNet) e pelo notável **poder computacional** fornecido pelas Unidades de Processamento Gráfico (GPUs), que as CNNs alcançaram seu potencial pleno (LITJENS et al., 2017). Um momento crucial ocorreu em 2012, quando **Alex Krizhevsky, Ilya Sutskever e Geoffrey Hinton** apresentaram a **AlexNet** na competição Large Scale Visual Recognition Challenge (ILSVRC) do ImageNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). A AlexNet demonstrou uma superioridade impressionante, reduzindo drasticamente a taxa de erro e inaugurando uma nova era para o aprendizado profundo e o processamento de imagens. Este marco revolucionou a área, impulsio-

nando a pesquisa e o desenvolvimento de arquiteturas cada vez mais complexas e eficientes, como VGGNet, GoogLeNet e ResNet. As aplicações das CNNs rapidamente se estenderam a diversos campos, com um impacto particularmente transformador no **campo médico**, especialmente na análise de exames radiológicos para diagnóstico e prognóstico de doenças (LITJENS et al., 2017).

3.2 Definição e Fundamentos das CNNs

As **Redes Neurais Convolucionais (CNNs)** representam uma classe especializada de redes neurais profundas, intrinsecamente projetadas para o processamento eficiente de dados com uma **estrutura de grade** ("grid-like topology"), como imagens digitais, séries temporais ou dados de áudio (GOODFELLOW; BENGIO; COURVILLE, 2016). Diferentemente das redes neurais totalmente conectadas, as CNNs exploram a natureza local e composicional dos dados de entrada por meio de sua arquitetura característica, que segue uma estrutura sequencial comum baseada em operações interligadas. Essa organização permite que a rede aprenda representações hierárquicas e cada vez mais abstratas dos dados de entrada de forma automática e eficiente, mimetizando aspectos do sistema visual humano (FUKUSHIMA, 1980; LITJENS et al., 2017).

A arquitetura típica de uma CNN para tarefas de classificação de imagens geralmente compreende os seguintes componentes principais, organizados em uma sequência que permite a extração progressiva de características:

1. **Camadas Convolucionais (Conv2D)**: Esta é a operação fundamental de uma CNN. Definida como uma operação matemática, ela aplica um conjunto de **filtros aprendíveis** (ou *kernels*) sobre os dados de entrada para extrair **mapas de características** (*feature maps*) (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Cada filtro é uma pequena matriz de pesos que desliza sobre a entrada, realizando produtos de ponto para detectar padrões locais específicos, como bordas, texturas ou formas. A convolução é crucial por preservar as **relações espaciais** entre os pixels, uma característica essencial para o reconhecimento de imagens. Os filtros são aprendidos durante

o treinamento da rede, permitindo que a CNN detecte hierarquicamente características cada vez mais abstratas à medida que a informação avança pelas camadas da rede (GOODFELLOW; BENGIO; COURVILLE, 2016).

2. **Funções de Ativação (ReLU - Rectified Linear Unit)**: Inseridas logo após as camadas convolucionais (e, por vezes, após camadas densas), as funções de ativação introduzem a **não-linearidade** necessária para que a rede possa aprender e modelar relações complexas nos dados (GOODFELLOW; BENGIO; COURVILLE, 2016). Se as CNNs consistissem apenas em operações lineares, sua capacidade de aprender padrões complexos seria severamente limitada. A ReLU é uma das funções de ativação mais comuns e eficientes, definida por $f(x) = \max(0, x)$, que simplesmente retorna o valor de entrada se for positivo e zero caso contrário, acelerando o treinamento e evitando problemas de gradiente.

3. **Pooling (MaxPooling2D)**: Após a operação de convolução e a aplicação da função de ativação, as camadas de pooling são tipicamente empregadas para **reduzir a dimensionalidade** dos mapas de características. Essa subamostragem mantém as informações mais relevantes, como a presença de uma característica, enquanto diminui a complexidade computacional da rede e o número de parâmetros. Essa redução de dimensionalidade também contribui para tornar a representação mais robusta a pequenas translações ou distorções na entrada. O **MaxPooling** é o tipo de pooling mais comum, selecionando o valor máximo em uma região definida dos mapas de características (LITJENS et al., 2017).

4. **Flatten (Achatar)**: Após várias camadas convolucionais e de pooling terem extraído características abstratas, os mapas de características resultantes são tipicamente multi-dimensionais (por exemplo, 3D: altura x largura x número de canais/filtros). A operação de **Flatten** transforma esses mapas 2D (ou 3D) em um único **vetor unidimensional**. Essa etapa é essencial para que as características aprendidas possam ser alimentadas nas camadas densas subsequentes, que esperam uma entrada linear (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

5. **Camadas Densas (Dense)**: Também conhecidas como camadas totalmente conectadas, as camadas densas recebem o vetor unidimensional gerado pela operação de Flatten. Nesta parte da rede, cada neurônio recebe entradas de todos os neurônios da camada anterior. Elas são responsáveis por integrar as características de alto nível aprendidas pelas camadas convolucionais e de pooling, e realizar a **decisão final** da rede, seja ela classificação ou regressão, com base nos padrões globais inferidos (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

6. **Softmax (ou Sigmoid)**: A camada final de uma CNN, em tarefas de classificação, geralmente utiliza uma função de ativação como **Softmax** ou **Sigmoid** para gerar as probabilidades das classes.

    - Para problemas de **classificação multiclasse** (onde a entrada pertence a uma e apenas uma categoria de várias), a função **Softmax** é aplicada para converter os valores de saída da camada densa final em uma distribuição de probabilidade. A soma das probabilidades para todas as classes será igual a 1, indicando a confiança da rede em cada uma das classes possíveis (GOODFELLOW; BENGIO; COURVILLE, 2016).
    - Para problemas de **classificação binária** (duas classes) ou **multilabel** (onde a entrada pode pertencer a múltiplas classes simultaneamente), a função **Sigmoid** é mais apropriada. Ela mapeia as saídas para um valor entre 0 e 1, que pode ser interpretado como a probabilidade de pertencer àquela classe específica.

A estrutura sequencial das CNNs permite um processo hierárquico de aprendizagem: as **primeiras camadas** da rede tendem a aprender e detectar **padrões simples e de baixo nível**, como bordas e texturas. À medida que os dados progridem através das camadas convolucionais mais profundas e de pooling, a rede combina esses padrões simples para formar e reconhecer **conceitos mais abstratos e complexos**, culminando nas camadas densas que tomam a decisão final de classificação (GOODFELLOW; BENGIO; COURVILLE, 2016).

Esta arquitetura, que busca mimetizar aspectos do sistema visual humano, permite que as CNNs aprendam representações hierárquicas e cada vez mais

abstratas dos dados de entrada de forma automática e eficiente, distinguindo-as de métodos tradicionais de visão computacional que dependem de extração manual de características (FUKUSHIMA, 1980; LITJENS et al., 2017).

4 Desenvolvimento

4.1 Apesentação do conjunto de dados

4.1.1 Plataforma on-line Kaggle

O Kaggle é bastante conhecida na área de ciência de dados e machine learning, ideal tanto para iniciantes quanto para profissionais. Ele oferece uma ampla variedade de conjuntos de dados públicos que podem ser utilizados para análise, visualização e treinamento de modelos.

4.1.2 Descrição do Dataset

O dataset utilizado, *"CoronaHack – Chest X-Ray-Dataset"*, está disponível no Kaggle (GOVINDRAJAN, 2020), e possui as seguintes características:

Características Principais

- **Tipo de Dados:** Imagens de raios-X torácicos.
- **Objetivo:** Classificar condições pulmonares — principalmente "Healthy vs Pneumonia (Corona)", em tarefas binárias e multi-classe.
- **Tamanho:** Cerca de 5.800 imagens ($\approx$ 1,5 GB).
- **Formato:** Imagens em `.png` e `.jpeg`, com metadados em `.csv`.

Estrutura do Dataset

- **Divisão em Pastas:** `train` (treino) e `test` (teste). A pasta de teste geralmente não contém rótulos (aparecem como `NaN` no CSV).
- **Metadados:** Contém:

- `Label`: Classe da imagem (normal/pneumonia).
- `Dataset_type`: Indica treino ou validação.
- `Modality`: Tipo de exame, por ex., X-ray.

4.2 Descrição da implementação e parâmetros utilizados

O trabalho foi organizado em fases para melhor distribuição de tarefas aos integrantes do grupo e tambem para serialização das alterações dos dados, foi criado um notebook direto na plataforma Kaggle. "https://www.kaggle.com/code/werbertarles/classificao-de-imagens-com-cnn-v4"

O desenvolvimento do projeto foi estruturado em fases, com o objetivo de organizar de forma eficiente a distribuição de tarefas entre os membros do grupo e facilitar o acompanhamento sequencial das transformações aplicadas aos dados ao longo do processo.

4.3 Fase 1: Preparação e Organização dos Dados

4.3.1 Fase 1a: Preparação e Organização Inicial dos Dados

Esta fase abrange as etapas iniciais de aquisição, pré-processamento e organização do conjunto de dados de raios-X do tórax, essencial para o treinamento de modelos de aprendizado profundo.

4.3.2 Download e Organização Inicial do Dataset

O primeiro passo consiste no download e na organização fundamental do dataset.

- O script `codigo 01` foi empregado para baixar o conjunto de dados coronahack_chest_xray_dataset diretamente do KaggleHub.
- organizar uma pasta de destino foi criada no diretório de execução do script para abrigar os dados baixados do kaggle.

Análise gráfica do balanceamentos dos dados do Dataset

Após a etapa de organização e limpeza inicial dos metadados, o gráfico a seguir apresenta a distribuição das classes (Normal, Pneumonia Bacteriana, Pneumonia Viral) nos conjuntos originais de treino e teste do dataset do Kaggle. Esta visualização é fundamental para compreender o balanceamento inerente dos dados antes de qualquer intervenção de pré-processamento

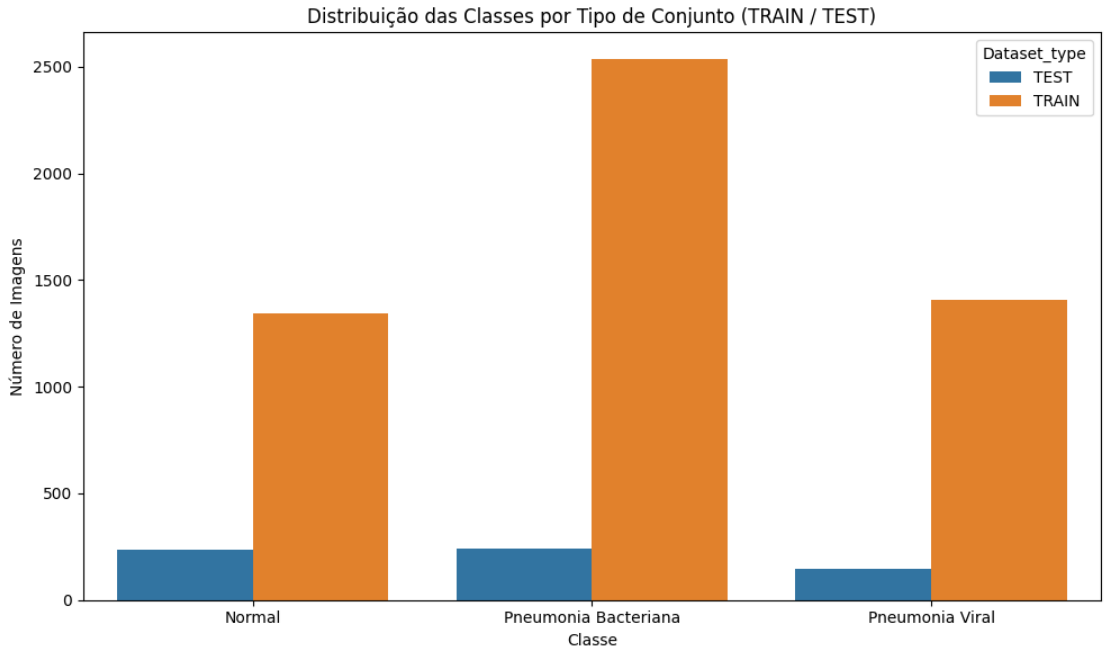

Figura 1 – Analise Inicial dos dados

observamos um desbalanceamento significativo, onde a classe **Pneumonia Bacteriana** é amplamente majoritária, a **Normal** é intermediária e a **Pneumonia Viral** é a classe minoritária, tanto nos conjuntos de treino quanto de teste. Essa disparidade quantitativa, se não abordada, poderia levar a um modelo enviesado, propenso a ter alta acurácia geral ao custo de uma baixa capacidade de detecção das classes minoritárias, clinicamente mais críticas.

4.3.3 Fase 1b - Organização, Balanceamento e Redimensionamento das Imagens

Para mitigar o desbalanceamento de classes identificado na análise inicial e otimizar o processo de treinamento do modelo, uma estratégia de pré-processamento de dados foi cuidadosamente aplicada. Primeiramente, o conjunto de teste fornecido originalmente pelo Kaggle foi isolado e mantido intacto, servindo como uma avaliação final e imparcial da capacidade de generalização do modelo. O restante dos dados, que compreende o conjunto de treino original do Kaggle, foi então preparado para as etapas de validação e balanceamento.

Dessa porção do dataset de treino original, um subconjunto foi primeiramente separado para o conjunto de validação, utilizando uma divisão estratificada. Isso assegura que o conjunto de validação mantenha a distribuição de classes proporcional à do dataset original, permitindo um monitoramento realista do desempenho do modelo em dados não vistos. Em seguida, o remanescente dos dados (o que sobrou do treino original após a retirada da validação) foi submetido a um processo de balanceamento por under-sampling: o número de imagens de cada classe foi igualado ao da classe minoritária dentro desse subconjunto, resultando em um conjunto de treinamento balanceado. Por fim, todas as imagens dos conjuntos de treino, validação e teste foram padronizadas para 128x128 pixels e organizadas em uma estrutura de diretórios apropriada, com seus metadados salvos, preparando os dados para a fase de modelagem.

Análise gráfica após o balanceamentos dos dados

O conjunto de treinamento (verde escuro) demonstra estar balanceado, com aproximadamente 1070-1080 imagens para cada uma das três classes ("Normal", "Pneumonia Bacteriana"e "Pneumonia Viral"), confirmando que o under-sampling foi eficaz em proporcionar uma base de aprendizado equitativa para o modelo. Por outro lado, os conjuntos de teste (azul escuro) e validação (verde claro), que foram estrategicamente divididos de forma estratificada do dataset original, mantêm o desbalanceamento inerente ao problema: as classes "Pneumonia Bacteriana"e "Normal"são mais prevalentes, enquanto "Pneumonia Viral"permanece a classe

minoritária em ambos. Essa configuração é ideal, pois enquanto o treino balanceado otimiza o aprendizado, os conjuntos de teste e validação desbalanceados (mas representativos) oferecem uma avaliação realista do desempenho do modelo em cenários do mundo real.

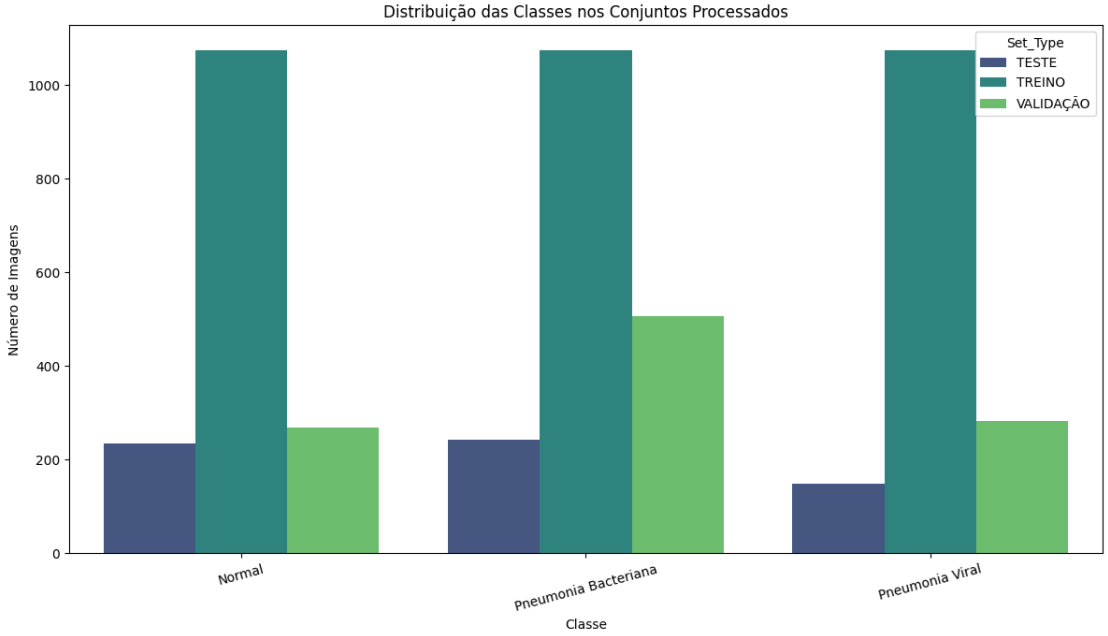

Figura 2 – Analise após balanceamento

4.4 Fase 2: Construção e Treinamento do Modelo de Deep Learning

4.4.0.1 Normalização dos Dados

A normalização dos dados de imagem é uma etapa fundamental no pipeline de pré-processamento para otimizar o treinamento de redes neurais profundas. No contexto deste trabalho, os valores de pixel originais, que tipicamente variam de 0 a 255, foram escalados para o intervalo de $[0, 1]$ através da divisão por 255.

Esta transformação foi implementada dinamicamente utilizando a classe `ImageDataGenerator` do *TensorFlow/Keras*.

O parâmetro `rescale=1./255` dentro do `ImageDataGenerator` foi o responsável direto por realizar essa operação de escala, aplicando-a a cada pixel carregado do disco. Essa abordagem garante que as imagens são apresentadas ao modelo no formato numérico mais adequado no momento do treinamento, sem a necessidade de salvar múltiplas cópias normalizadas das imagens no disco, preservando a flexibilidade e otimizando o uso do armazenamento. Adicionalmente, o `ImageDataGenerator` também incorpora as transformações de *Data Augmentation* para o conjunto de treino, conforme as especificações dos hiperparâmetros de rotação, zoom e deslocamento, que complementam a normalização para aumentar a robustez do modelo.

4.4.1 Hiperparâmetros Iniciais

Os hiperparâmetros iniciais utilizados na configuração do modelo e do processo de treinamento são detalhados a seguir, com uma abordagem em duas fases para otimização: uma fase inicial com o modelo base congelado e uma fase subsequente de ajuste fino (fine-tuning).

- **Parâmetros Gerais e de Dados:**
  - Tamanho da Imagem Alvo (`TARGET_IMAGE_SIZE`): $(128, 128)$ pixels.
  - Tamanho do Batch (`BATCH_SIZE`): 32.
  - Semente Aleatória (`RANDOM_STATE`): 42.

- **Parâmetros de Épocas por Fase:**
  - Número de Épocas para a Fase 1 (`NUM_EPOCHS_PHASE1`): 50 (treinamento da cabeça de classificação).
  - Número de Épocas para a Fase 2 (`NUM_EPOCHS_PHASE2`): 30 (fase de fine-tuning).
  - Número Total de Épocas (`TOTAL_EPOCHS`): 80.
- **Hiperparâmetros do Otimizador:**
  - Tipo de Otimizador: `Adam`.
  - Taxa de Aprendizado da Fase 1 (`LEARNING_RATE`): 0.001.
  - Taxa de Aprendizado do Fine-tuning (`FINE_TUNE_LEARNING_RATE`): 0.00001 (`LEARNING_RATE / 100`).
- **Hiperparâmetros da Arquitetura do Modelo:**
  - Modelo Base para Transfer Learning: `MobileNetV2`.
  - Neurônios na primeira camada densa 1: 512.
  - Taxa de dropout após a primeira camada densa: 0.5
  - Neurônios na segunda camada densa: 256.
  - Taxa de dropout após a segunda camada densa:0.3
- **Hiperparâmetros de Data Augmentation (para Treinamento):**
  - Escala de Normalização de Pixels: 1./255 (intervalo $[0, 1]$).
  - Intervalo de Rotação (`rotation_range`): 30 graus.
  - Intervalo de Zoom (`zoom_range`): 0.2.
  - Intervalo de Deslocamento de Largura (`width_shift_range`): 0.2.
  - Intervalo de Deslocamento de Altura (`height_shift_range`): 0.2.
  - Transformação de cisalhamento (`shear_range`): 0.2.
  - Espelhamento Horizontal (`horizontal_flip`): `True`.
  - Modo de Preenchimento (`fill_mode`): `nearest`.

- **Hiperparâmetros dos Fine-Tuning e Callbacks:**
  - Número de camadas a descongelar no modelo base
  - **HP_UNFREEZE_LAYERS: 20**
  - Paciência para parada antecipada
  - **HP_EARLY_STOPPING_PATIENCE: 15**
  - Paciência para redução da taxa de aprendizado
  - **HP_LR_PLATEAU_PATIENCE: 5**
  - Fator de redução da taxa de aprendizado
  - **HP_LR_PLATEAU_FACTOR: 0.5**

Grafico de Acurácia, Perda, Precisão e Recall inicial

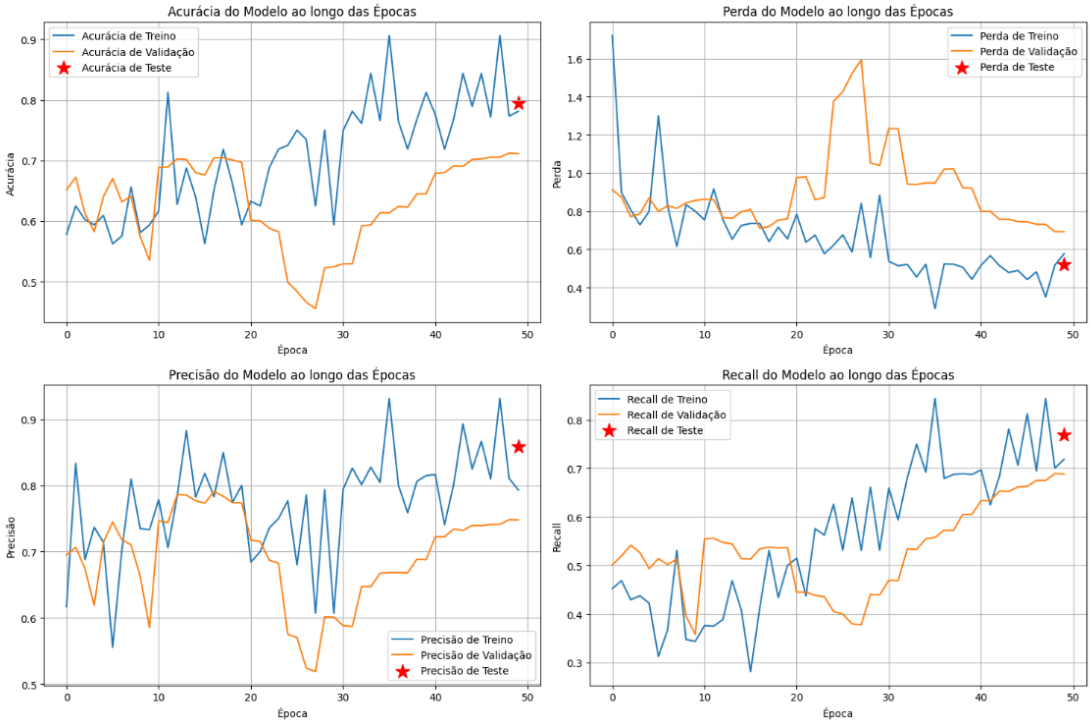

Figura 3 – Gráfico de análise inicial

- RESULTADO DO BANCO DE TESTE
- Loss: 0.5506
- Accuracy: 0.7901
- Precision: 0.8603
- Recall: 0.7783

4.5 Histórico de Mudanças nos Hiperparâmetros

Apresentaremos aqui as modificações nos hiperparâmetros realizadas ao longo do processo de otimização do modelo, com base nas análises dos resultados de treinamento e validação.

Serão mantidos todos os hiperparâmetros iniciais e alterados parametros especificos buscando a melhora da acurácia

4.5.1 `Alteração de hiperparâmetros 1`

- NUM_LAYERS_TO_UNFREEZE: Alterado para 50.
- REDUCE_LR_PATIENCE: Alterado para 7.
- Early_stopping.patience: Alterado para 20.

4.5.1.1 Gráfico Alteração 1

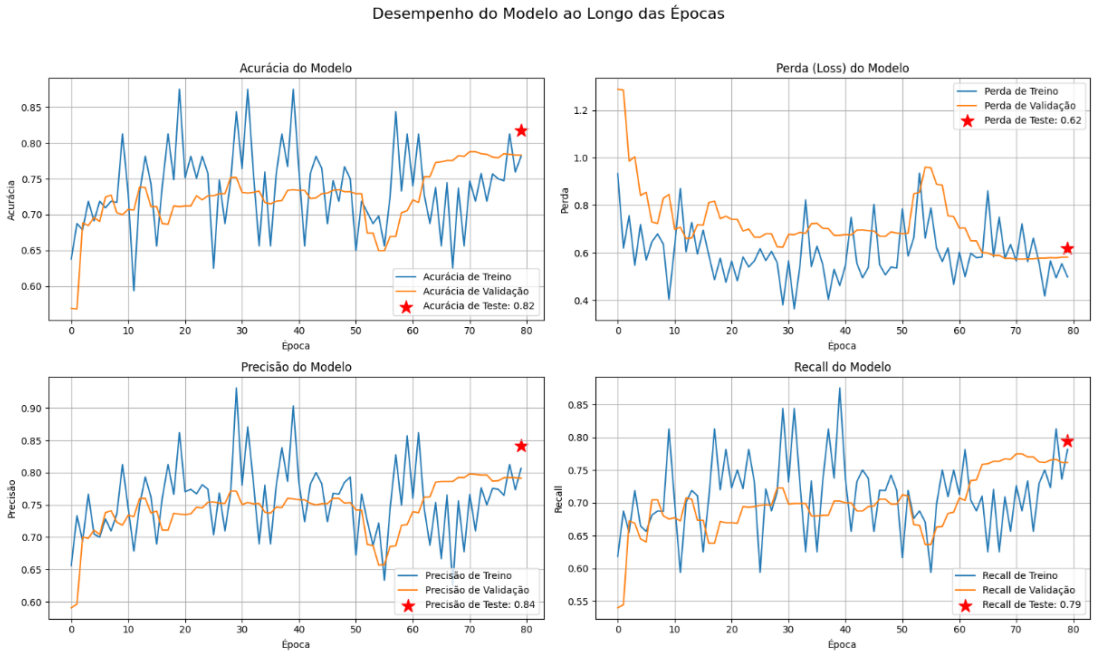

Figura 4 – Modificação 01

- RESULTADO DO BANCO DE TESTE
- Loss: 0.6183
- Accuracy: 0.8174
- Precision: 0.8415
- Recall: 0.7944

4.5.2 `Alteração de hiperparâmetros 2`

- Dropout da primeira camada densa: Alterado para 0.6.
- Dropout da segunda camada densa: Alterado para 0.4.
- FINE_TUNE_LEARNING_RATE: Alterado para 0.000005.
- NUM_LAYERS_TO_UNFREEZE: Reduzir para 30.

4.5.2.1 Grafico Alteração 2

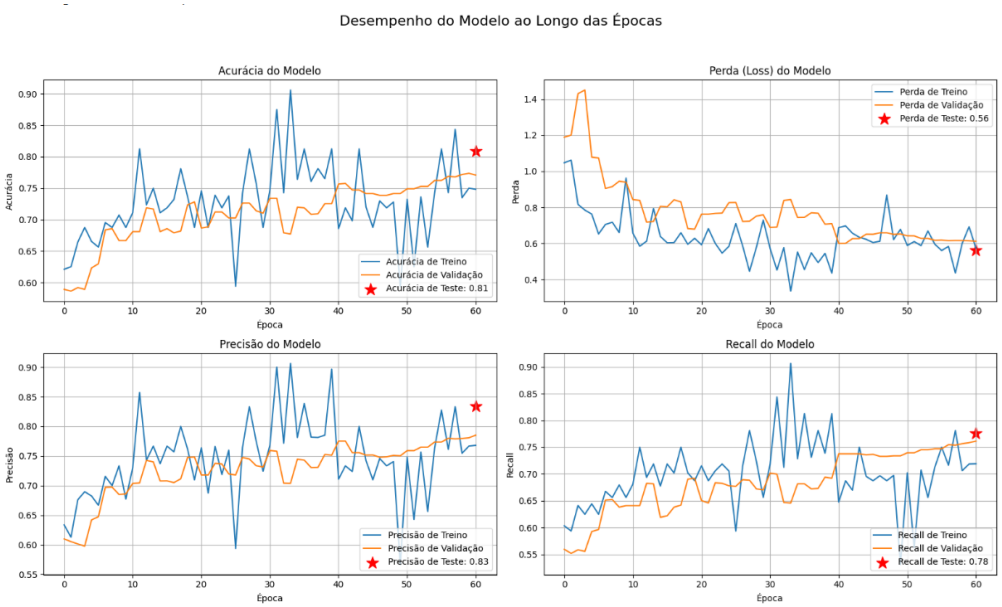

Figura 5 – Modificação 02

- RESULTADO DO BANCO DE TESTE
- Loss: 0.5614
- Accuracy: 0.8092
- Precision: 0.8339
- Recall: 0.7763

4.6 Resultado final do banco teste

4.6.1 `Alteração final de hiperparâmetros`

- Dropout da primeira camada densa: Alterado para 0.5.
- Dropout da segunda camada densa: Alterado para 0.3.
- Número de Épocas para a Fase 1 (`NUM_EPOCHS_PHASE2`): 100
- Número de Épocas para a Fase 2 (`NUM_EPOCHS_PHASE2`): 30
- NUM_LAYERS_TO_UNFREEZE: alterado para 20.

- RESULTADO DO BANCO DE TESTE
- Loss: 0.5493
- Accuracy: 0.8125
- Precision: 0.8449
- Recall: 0.7796

4.6.1.1 Grafico Alteração final

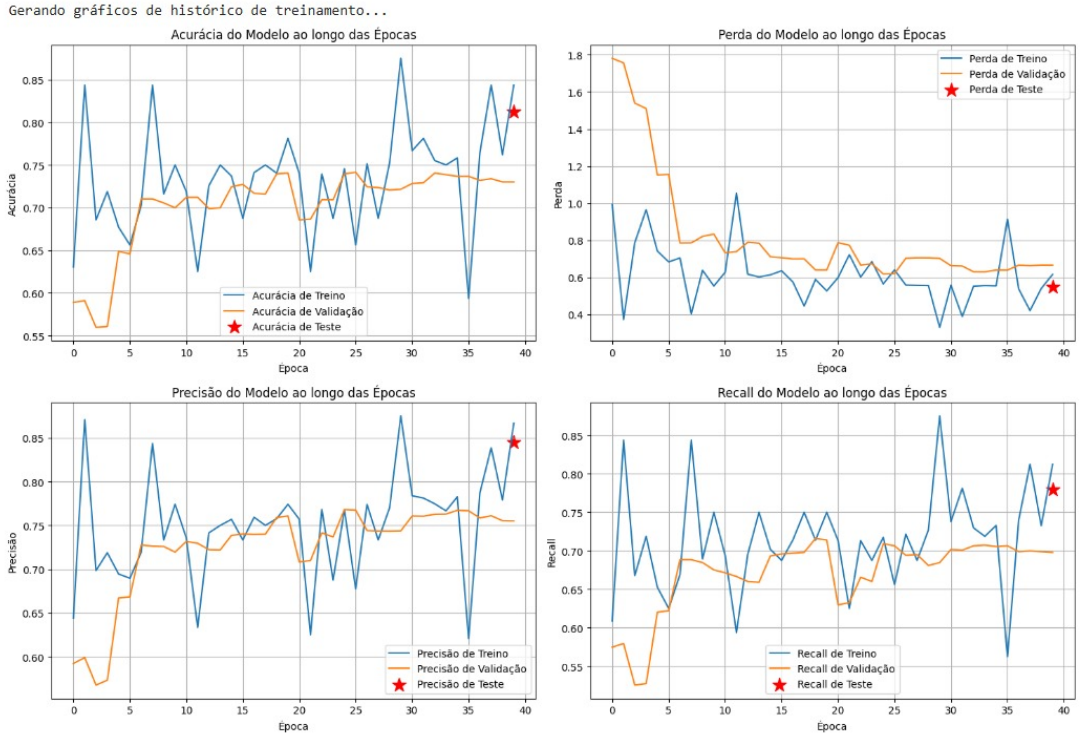

Figura 6 – Analise final

5 Dificuldades e Soluções

Os principais desafios foram a escassez de dados para treinamento e o risco iminente de overfitting, onde o modelo poderia memorizar os dados de treino em vez de generalizar. A solução central foi a aplicação de Aprendizagem por Transferência (Transfer Learning) com o modelo MobileNetV2, aproveitando seu conhecimento pré-treinado para reduzir a necessidade de um vasto conjunto de dados. Para combater o overfitting diretamente, foi implementada uma forte política de Data Augmentation, que criou variações sintéticas das imagens, juntamente com o uso de camadas de Dropout e a técnica de Early Stopping para interromper o treinamento no momento ideal.

Além disso, foram abordados desafios mais sutis, como o desbalanceamento entre as classes e a otimização da convergência do modelo. O problema do desbalanceamento foi mitigado através da aplicação de pesos de classe, forçando o modelo a dar igual importância a todas as categorias, independentemente de sua frequência. A otimização do treinamento foi garantida por uma estratégia de duas fases (treino inicial e fine-tuning) e pelo uso de um agendador de taxa de aprendizado dinâmico (ReduceLROnPlateau), que ajustava a otimização para evitar estagnação e garantir uma convergência mais estável e eficaz.

Em suma, o sucesso do modelo não se deve a uma única técnica, mas à combinação de múltiplas soluções estratégicas. A abordagem integrada — desde a escolha de um modelo pré-treinado e a regularização robusta até o balanceamento de dados e o controle preciso do processo de treinamento — foi fundamental para desenvolver um classificador de imagens que não apenas aprendeu com os dados disponíveis, mas também demonstrou uma excelente capacidade de generalização para dados novos, conforme validado pelas métricas finais no conjunto de teste.

6 Análises dos Resultados

6.1 Observações importantes

Após a conclusão de todas as fases de treinamento e otimização, o modelo final baseado na arquitetura MobileNetV2 obteve desempenho satisfatório no conjunto de teste. Os principais indicadores de avaliação foram: perda (loss) de 0,5493, acurácia de 81,25%, precisão de 84,49% e recall de 77,96%. Esses resultados indicam que o modelo foi capaz de aprender de forma eficaz os padrões presentes no conjunto de dados, conseguindo manter um equilíbrio entre a correta classificação de amostras positivas e a minimização de falsos positivos.

Durante o treinamento inicial, em que apenas a cabeça de classificação foi ajustada com o modelo base congelado, a evolução da acurácia e a redução da perda foram progressivas, mas com sinais iniciais de overfitting. Foi na segunda fase, com a realização do fine-tuning nas últimas camadas da MobileNetV2 e a redução da taxa de aprendizado, que o modelo apresentou melhorias significativas na capacidade de generalização. A estratégia de descongelamento controlado de camadas (primeiramente 40, depois 50 camadas, e posteriormente ajustado para 30) foi essencial para encontrar um ponto de equilíbrio entre aprender características específicas do conjunto e evitar a sobreajuste.

As técnicas de data augmentation aplicadas também desempenharam papel relevante na robustez do modelo, ampliando artificialmente a diversidade do conjunto de treino e permitindo que a rede fosse exposta a variações mais realistas das imagens. Além disso, o cálculo de pesos de classe ajudou a compensar o desbalanceamento observado entre as categorias, garantindo que o modelo não fosse tendencioso para a classe majoritária.

A análise gráfica das curvas de acurácia, perda, precisão e recall ao longo das épocas reforçou as conclusões quantitativas, mostrando uma evolução consistente nos desempenhos de validação e teste. Embora algumas oscilações tenham sido observadas, principalmente nas primeiras tentativas com hiperparâmetros mais

agressivos, os ajustes subsequentes nos callbacks, dropout, learning rate e número de camadas treináveis proporcionaram uma trajetória mais estável na reta final do treinamento.

Os resultados obtidos demonstram que o modelo é capaz de realizar a tarefa de classificação multiclasse de maneira eficiente, apresentando níveis aceitáveis de desempenho em métricas-chave, mesmo diante de um conjunto de dados com limitações de tamanho e balanceamento.

6.2 pontos positivos, negativos e aprendizados

A solução representa um sucesso no uso de ML, com pontos positivos claros na sua metodologia robusta, que combinou Transfer Learning com uma forte estratégia anti-overfitting (Data Augmentation, Dropout), resultando em um código organizado e em um modelo com excelente capacidade de generalização comprovada no teste. Como pontos de melhoria, destacam-se a seleção manual de hiperparâmetros, que poderia ser otimizada com ferramentas de ajuste automático, a exploração de uma única arquitetura de modelo que poderia ser testada com mais de uma arquitetura e verificado o comportamento. O principal aprendizado consolidado é que o desempenho não veio de uma solução isolada, mas da combinação de múltiplas técnicas que abordam problemas específicos, provando que uma abordagem integrada e centrada nos dados é fundamental para o sucesso em Machine Learning.

7 Conclusão

O desenvolvimento do modelo de classificação de imagens utilizando transferência de aprendizado com a arquitetura MobileNetV2 apresentou resultados satisfatórios e coerentes com os objetivos propostos neste trabalho. As diferentes fases de treinamento, acompanhadas de ajustes finos nos hiperparâmetros, mostraram-se fundamentais para alcançar níveis adequados de desempenho.

A combinação de estratégias como o congelamento inicial da base pré-treinada, a posterior realização de fine-tuning em camadas selecionadas, a aplicação de técnicas de data augmentation, o uso de regularização via dropout e o ajuste dos pesos das classes permitiu uma adaptação eficaz da rede às características do conjunto de dados específico utilizado. A implementação de callbacks, como EarlyStopping, ModelCheckpoint e ReduceLROnPlateau, também foi essencial para controlar o aprendizado e evitar o sobreajuste.

Os resultados quantitativos, com uma acurácia superior a 80% e valores equilibrados de precisão e recall, indicam que o modelo tem bom potencial de aplicação em cenários reais de classificação de imagens, mesmo quando exposto a dados com características diferentes dos utilizados no treinamento.

Como perspectiva futura, recomenda-se explorar arquiteturas mais complexas e realizar experimentos com outras técnicas de regularização e otimização de hiperparâmetros. Além disso, análises mais detalhadas com a utilização de métricas como matriz de confusão, F1-score e curva ROC podem fornecer insights adicionais sobre o comportamento do modelo em diferentes classes. Também seria interessante investigar o desempenho do modelo com técnicas de ensemble ou aplicando métodos de fine-tuning mais profundos, descongelando camadas adicionais da rede base.

Por fim, a continuidade deste trabalho pode incluir a ampliação do conjunto de dados, a aplicação de métodos de validação cruzada e a avaliação do desempenho do modelo em ambientes de produção, garantindo a sua robustez e escalabilidade para uso em aplicações práticas de classificação de imagens.

Referências

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, Springer, v. 36, n. 4, p. 193–202, 1980.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016.

GOVINDRAJAN, P. *CoronaHack - Chest X-Ray-Dataset*. 2020. <https://www.kaggle.com/datasets/praveengovi/coronahack-chest-xraydataset>. Accessed: 2025-06-10.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.]: Curran Associates, Inc., 2012. v. 25, p. 1097–1105.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

LITJENS, G.; KOOI, T.; BEJNORDI, B. E.; SETIO, A. A. A.; CIOMPI, F.; GHAFOORIAN, M.; LAAK, J. A. van der; GINNEKEN, B. van; SÁNCHEZ, C. I. A survey on deep learning in medical image analysis. *Medical image analysis*, Elsevier, v. 42, p. 60–88, 2017.