UNISINOS
UNIVERSIDADE DO RIO DOS SINOS

Curso de Especialização em Engenharia e Ciências de Dados
Deep Learning

Atividade Avaliativa

Aluno: Andressa Magnus Friedrichs **Data da entrega**: 07/dez/2025
Vídeo de apresentação: https://youtu.be/3gl--k27qto

Detecção de COVID-19, Pneumonia e Normal em Radiografias de Tórax usando CNN do zero e ResNet50

1. Introdução

A pandemia da COVID-19 evidenciou a necessidade de ferramentas rápidas e eficazes para auxiliar profissionais de saúde no diagnóstico inicial e na triagem de pacientes. Embora o teste RT-PCR permaneça como principal método de detecção do SARS-CoV-2, sua disponibilidade limitada em regiões vulneráveis, custos elevados e tempo de processamento motivaram a busca por métodos auxiliares.

Radiografias de tórax surgem como alternativa prática, rápida e de baixo custo. Entretanto, sua interpretação é subjetiva e depende do nível de expertise do radiologista. Modelos de Deep Learning, especialmente Redes Neurais Convolucionais (CNNs), têm se mostrado eficazes para identificar padrões associados a COVID-19, pneumonia e pulmões saudáveis.

Este trabalho apresenta o desenvolvimento completo, em estilo literate programming, de dois modelos:

- **CNN construída do zero**, totalmente aprendida a partir do dataset.
- **ResNet50 com Transfer Learning e Fine-Tuning**, adaptada ao domínio médico.

O relatório integra explicações, código, tabelas, experimentos e resultados, formando um documento técnico reprodutível.

Hipóteses do Estudo

Hipótese 1 — Radiografias de pacientes com COVID-19 apresentam padrões característicos (opacidades em vidro fosco, infiltrados bilaterais) detectáveis por CNNs.

Hipótese 2 — Data augmentation, padronização e balanceamento adequado permitem que CNNs aprendam representações discriminativas entre COVID, Pneumonia e Normal.

Hipótese 3 — O uso de Grad-CAM permitirá identificar regiões relevantes do pulmão na decisão dos modelos, reforçando sua interpretabilidade.

Hipótese 4 — Modelos pré-treinados (ResNet50) apresentarão desempenho superior à CNN do zero, devido ao aprendizado prévio de filtros mais ricos.

Dataset e Estruturação dos Dados

3.1 Fonte dos Dados

Foram combinados quatro datasets públicos do Kaggle:

- **Chest X-ray (COVID-19 & Pneumonia).**
- **COVID-19 Radiography Dataset.**
- **Chest X-ray Pneumonia, COVID-19, Tuberculosis.**
- **Chest Xray for COVID-19 Detection.**

3.2 Classes Unificadas

Após limpeza e reorganização, todas as imagens foram agrupadas em três classes principais:

- **COVID-19**
- **Pneumonia** (incluindo Viral Pneumonia e Lung Opacity)
- **Normal**

3.3 Balanceamento por Amostragem

Para minimizar viés, o número total foi **equalizado para 5.473 imagens por classe**, totalizando:

$$\text{Total} = 3 \times 5473 = 16.419 \text{ imagens.}$$

3.4 Divisão Estratificada

A base foi dividida em:

Conjunto	Proporção	Total	Por Classe
Treino	70%	11.493	3.831
Validação	15%	2.460	820
Teste	15%	2.466	822

Código utilizado para estruturar as pastas:

```
BASE_DIR = "data"
OUTPUT_DIR = "data_split"
CLASSES = ["covid", "normal", "pneumonia"]

N_IMAGENS_POR_CLASSE = 5473
TRAIN_RATIO, VAL_RATIO, TEST_RATIO = 0.7, 0.15, 0.15

def limpar_e_criar_pastas():
    if os.path.exists(OUTPUT_DIR):
        shutil.rmtree(OUTPUT_DIR)
    for subset in ["train", "val", "test"]:
        for classe in CLASSES:
            os.makedirs(os.path.join(OUTPUT_DIR, subset, classe), exist_ok=True)
```

2. EDA - Análise Exploratória Detalhada

A Análise Exploratória de Dados (EDA) tem como objetivo examinar a qualidade, estrutura, consistência e características visuais das radiografias que alimentam os modelos. Essa etapa é essencial para garantir o pipeline de processamento e que os modelos de deep learning recebam dados adequados e representativos.

4.1 Distribuição das Imagens por Classe e Conjunto

O primeiro passo da EDA é avaliar a distribuição do dataset após o processo de balanceamento e separação em *train*, *val* e *test*.

Código utilizado

```
class_counts = {}
sets = ["train", "val", "test"]
classes = ["covid", "normal", "pneumonia"]

for s in sets:
    class_counts[s] = {c: len(os.listdir(os.path.join(base_dir, s, c))) for c in classes}
```

Tabela de distribuição

Conjunto	COVID-19	3831	Pneumonia	Total
Train	3831	3831	3831	11.493

Val	820	820	820	2.460
Test	822	822	822	2.466

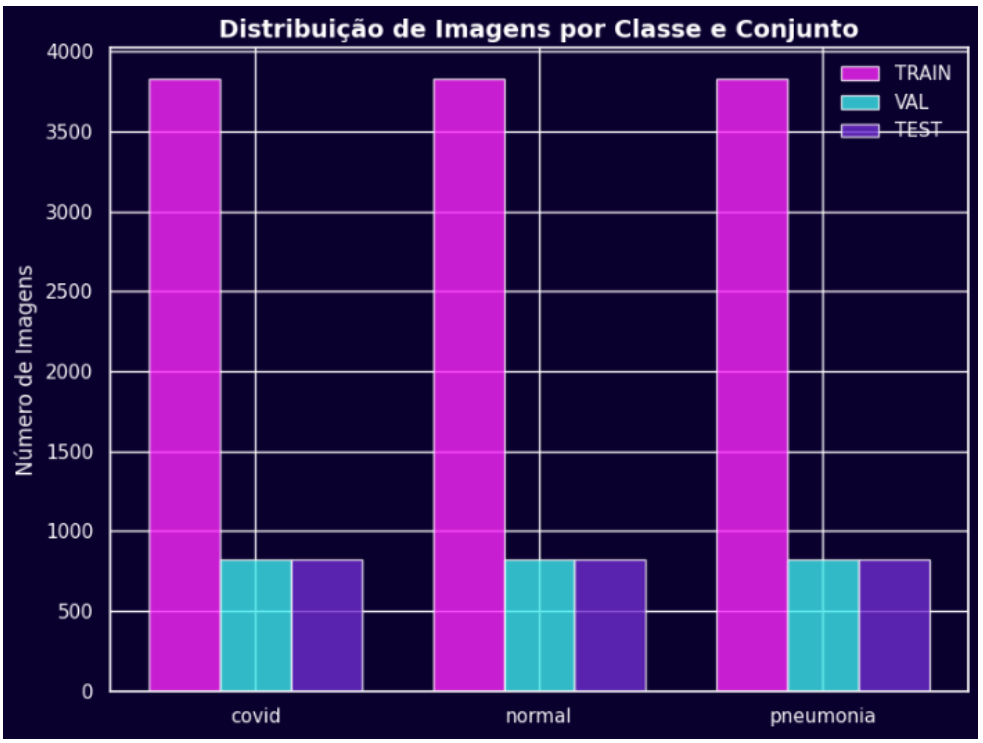
Conclusão:

O dataset está perfeitamente balanceado.

A razão entre classes é 1.00x, evitando vieses de aprendizagem.

4.2 Visualização da Distribuição por Classe

Figura 1 - Distribuição por Classe nos Conjuntos (Train/Val/Test)



Interpretação:

- A distribuição é homogênea em todos os subconjuntos.
- Nenhuma classe domina o treino — algo crítico para evitar vieses de predição.
- Isso confirma a eficácia do script de balanceamento.

4.3 Propriedades Estruturais das Imagens

As radiografias foram examinadas em termos de:

- resolução
- altura e largura
- *aspect ratio*
- modo de cor

Código utilizado

```
widths, heights, ratios, modes = analyze_image_properties(base_dir, classes)
```

Figura 2 - Distribuição das Resoluções

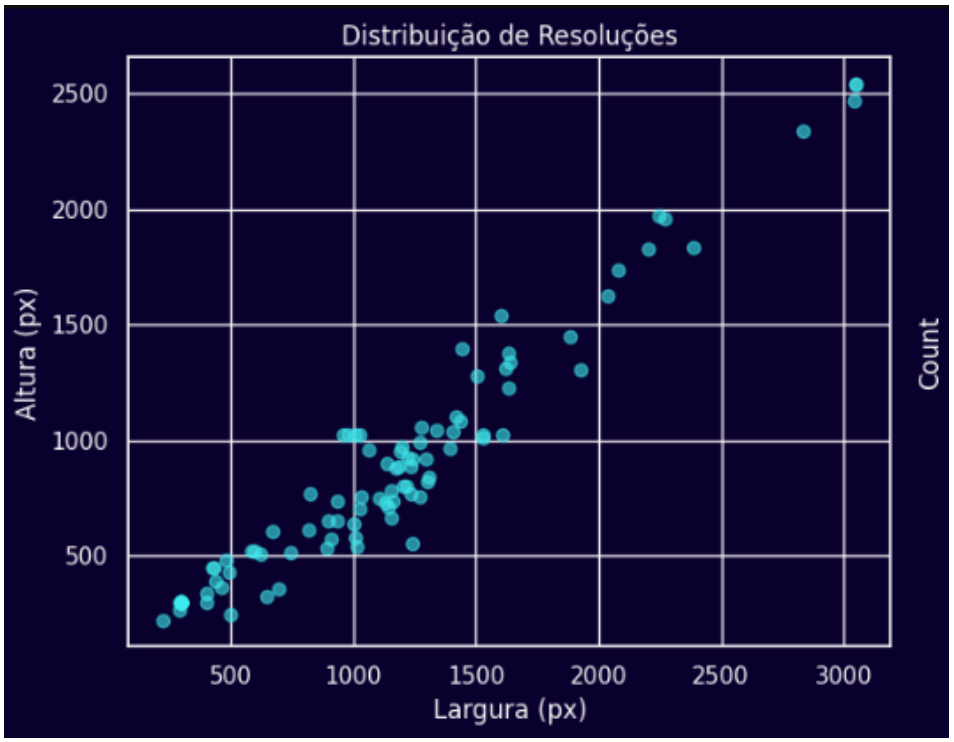
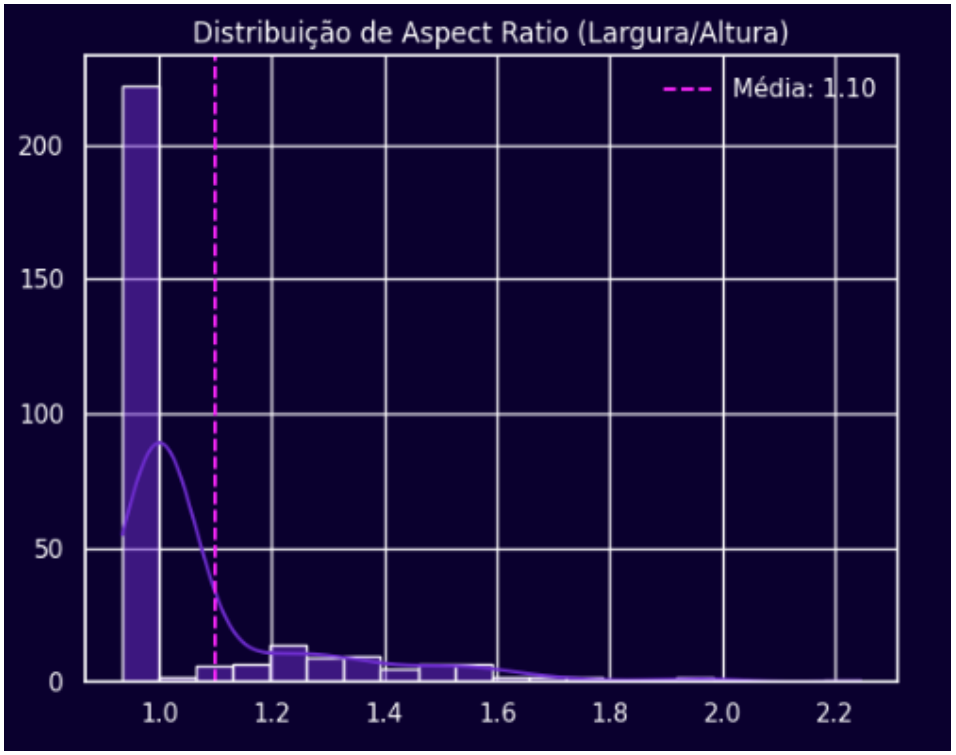


Figura 2 - Distribuição das Resoluções



Interpretação

- A maioria das imagens possui resoluções entre 400–2000 pixels, com média aproximada de 567×485.
- O *aspect ratio* médio é 1.10 → imagens levemente mais largas do que altas.
- Há heterogeneidade de tamanho, justificando o uso de resize para 224×224 no pipeline.

4.4 Análise de Intensidade e Textura (Entropia)

Este passo analisa brilho, contraste e complexidade da textura.

Código utilizado

```
fig, axes = plt.subplots(1, 2, figsize=(14, 5))
for c in classes:
    sns.kdeplot(stats[c]["intensity_values"], label=c)
```

Figura 3 - Distribuição da Intensidade por Classe

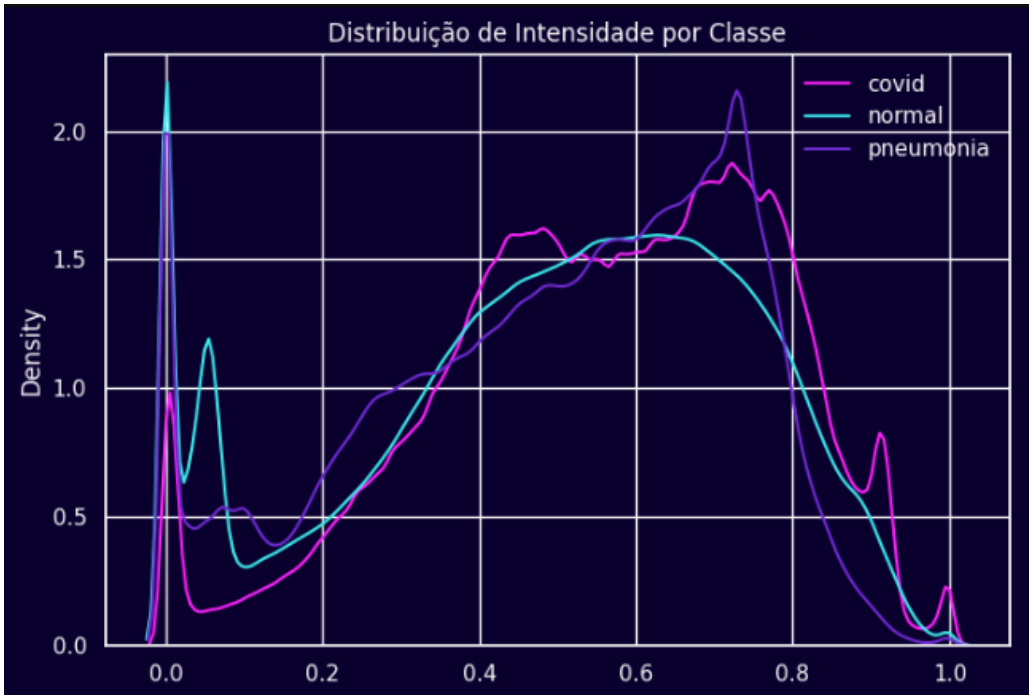
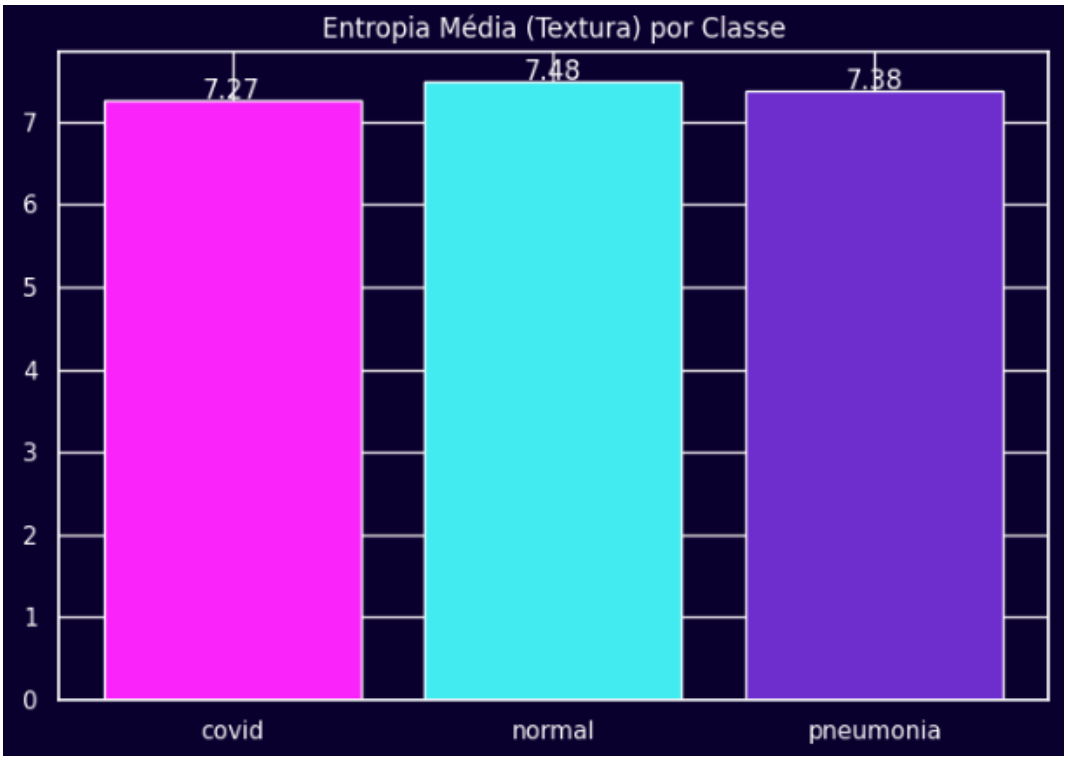


Figura 4 - Entropia Média por Classe



Interpretação

- Intensidade média global: 0.516.
- Entropia média global: 7.376.
- COVID-19 e Pneumonia tendem a apresentar maior entropia, devido a áreas irregulares e opacificações.
- Normais possuem intensidade mais estável, refletindo menor variabilidade estrutural.

Essas diferenças são benéficas para a CNN aprender padrões discriminativos.

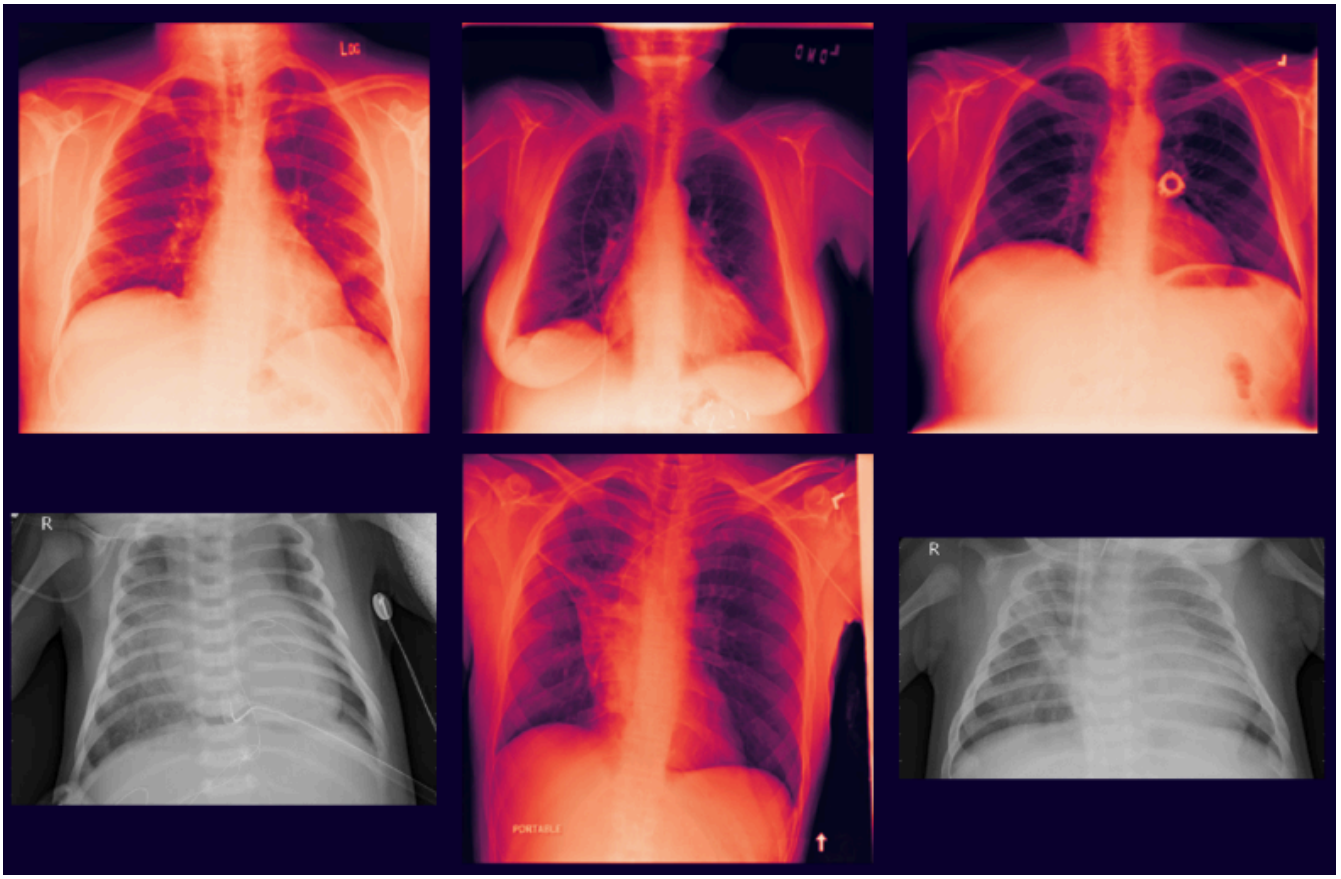
4.5 Amostras Visuais das Classes

Código utilizado

```
display_samples(base_dir, classes, n=3)
```

Interpretação das amostras

- As imagens são variadas, com diferentes máquinas de raio-X e padrões anatômicos.
- Há diferenças de rotação e intensidades, justificando **data augmentation**.
- Não há imagens claramente corrompidas no conjunto de treino.



4.6 Resumo Estatístico Final

Métrica	Valor
Resolução média	567×485 px
Aspect Ratio médio	1.10
Intensidade média	0.516
Entropia média	7.376
Balanceamento	perfeito (1:1:1)

Conclusão Geral da EDA

A base é de alta qualidade, bem estruturada, diversificada e com características visuais adequadas para treinamento de CNNs.

O EDA confirma que o dataset é adequado, representativo e seguro contra problemas comuns como:

- Desbalanceamento.
- Inconsistência visual.
- Ruído excessivo.

3. Preparação dos Dados para os Modelos

A preparação dos dados é uma das etapas mais críticas em um pipeline de deep learning aplicado à classificação de imagens médicas.
O objetivo desta seção é mostrar como os dados foram padronizados, balanceados e transformados antes de serem alimentados nos modelos CNN do Zero e ResNet50.

Esta etapa garante:

- Consistência entre amostras.
- Remoção de variações indesejadas.
- Melhor capacidade de generalização.
- Prevenção de vieses estruturais.
- Controle de sobremodelagem (*overfitting*).

5.1 Objetivos da Preparação dos Dados

A preparação implementada busca atender aos seguintes objetivos fundamentais:

Padronizar entrada:

- tamanho único (224×224),
- modo de cor consistente (grayscale ou RGB),
- normalização dos valores de pixel.

Aumentar variabilidade:

Com data augmentation controlada para evitar superdependência das amostras originais.

Ajustar o pré-processamento ao modelo:

- CNN do zero - escala de cinza + normalização em [0,1]
- ResNet50 - 3 canais RGB + pré-processamento padrão da rede ImageNet

Balancear classes:

- cálculo automático de **class weights**
- essencial para evitar viés no aprendizado

5.2 Configurações Básicas dos Geradores

Abaixo está o código que define os parâmetros principais da preparação:

```
# CLASSIFICAÇÃO DE IMAGENS
plt.style.use('seaborn-v0_8')

base_dir = "data_split"
IMG_SIZE = (224, 224)
BATCH_SIZE = 16
SEED = 42
NUM_CLASSES = 3
```

Justificativa das escolhas

Parâmetro	Justificativa
224×224	Padrão de entrada para ResNet50 e leve para treinar CNNs
Batch 16	Bom equilíbrio entre performance e uso de GPU
Seed 42	Reprodutibilidade dos experimentos
3 classes	COVID-19, Normal, Pneumonia

5.3 Data Augmentation para a CNN do Zero

```
train_datagen_cnn = ImageDataGenerator(
    rescale=1./255,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Explicação dos hiperparâmetros e impacto

Hiperparâmetro	Valor	Impacto
rotation_range	15°	Simula variações naturais de posição
shift (x,y)	10%	Melhora robustez a deslocamentos

zoom_range	0.1	Permite aproximações/afastamentos
horizontal_flip	True	Essencial em imagens simétricas como tórax
rescale	1/255	Normaliza pixels entre 0 e 1

Conclusão: Esses parâmetros ajudam o modelo a generalizar melhor, evitando que aprenda padrões fixos ou ruídos.

5.4 Data Augmentation para a ResNet50

```
train_datagen_resnet = ImageDataGenerator(
    preprocessing_function=resnet_preprocess,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Pré-processamento específico

ResNet50 utiliza a função:

$resnet\_preprocess(img) = img - mean_{ImageNet}$

Esse passo aproxima as radiografias do domínio das imagens naturais da ImageNet, onde a rede foi pré-treinada.

5.5 Função Geral de Geração dos Conjuntos

```
def get_generators(model_type="cnn"):
    if model_type == "resnet":
        train_datagen = train_datagen_resnet
        val_test_datagen = val_test_datagen_resnet
        color_mode = 'rgb'
    else:
        train_datagen = train_datagen_cnn
        val_test_datagen = val_test_datagen_cnn
        color_mode = 'grayscale'
```

Interpretação

Tipo	Modo de cor	Pré-processamento
CNN do Zero	Grayscale (1 canal)	Normalização
ResNet50	RGB (3 canais)	Pré-processamento da ImageNet

5.6 Cálculo dos Pesos de Classe

```
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(temp_gen.classes),
    y=temp_gen.classes
)
```

Por que usar class weights se a base está balanceada?

Mesmo com 1:1:1, pequenas variações estruturais podem levar a:

- dominância de padrões comuns em uma classe
- aprendizado enviesado
- instabilidade em classes com maior entropia

O uso de **class_weight** reforça equilíbrio e garante que cada classe contribua igualmente para o gradiente.

5.7 Tabela dos Hiperparâmetros de Pré-processamento

Hiperparâmetro	CNN do Zero	ResNet50
Modo de Cor	Grayscale	RGB
Normalização	1/255	resnet_preprocess
Tamanho	224×224	224×224
Augmentation	Sim	Sim
Shuffle	Sim	Sim
Seed	42	42

5.8 Conclusão da Preparação dos Dados

A preparação dos dados garante:

- Generalização adequada
- Consistência entre os modelos
- Pré-processamento alinhado com a arquitetura
- Controle sobre variância e ruído

Essa etapa estabelece bases sólidas para os experimentos posteriores.

6. Modelo CNN do Zero

6.1 Objetivo da Arquitetura CNN do Zero

O objetivo dessa CNN é criar uma baseline totalmente construída do zero, permitindo:

- avaliar a capacidade do modelo em aprender padrões diretamente do dataset,
- comparar com modelos pré-treinados (como ResNet50),
- observar como o desempenho evolui sem transferência de conhecimento externo,
- analisar o impacto de cada camada na classificação de COVID-19, Normal e Pneumonia.

6.2 Arquitetura da CNN

A arquitetura foi projetada segundo princípios clássicos para processamento de imagens médicas:

- convoluções progressivas: 32 → 64 → 128 filtros
- batch normalization: estabiliza aprendizado e acelera convergência
- pooling: reduz dimensionalidade preservando padrões
- rede densa moderada: 256 → 128 neurônios
- dropout: evita overfitting
- camada de saída softmax: classificação multiclasse

6.3 Código da Arquitetura

```python
# 1. MODELO PRINCIPAL PARA TREINO
cnn_model_train = tf.keras.Sequential([
    # 🔵 Primeira camada convolucional
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(224,224,1)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2,2),

    # 🟢 Segunda camada convolucional
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2,2),

    # 🔴 Terceira camada convolucional
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2,2),

    # 🔷 Camada densa
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.4),

    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.3),

    # 🟣 Camada de saída
    tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')
])
```

6.4 Justificativas das principais escolhas

Convoluções 3×3

Capturam padrões locais importantes (texturas pulmonares, opacidades, consolidações).

Profundidade progressiva

32 → 64 → 128 filtros simulam aprendizado hierárquico:

- primeiras camadas: bordas e estruturas básicas
- intermediárias: padrões anatômicos
- últimas: patologias e padrões complexos

Batch Normalization

Reduz covariate shift e acelera convergência mesmo com dataset relativamente pequeno.

Dropout 0.4 e 0.3

Evita que o modelo memorize padrões específicos de imagens.

Softmax

Indicado para classificação em 3 classes mutuamente exclusivas.

6.5 Compilação e Hiperparâmetros de Treinamento

```
cnn_model_train.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Justificativas

Hiperparâmetro	CNN do Zero	ResNet50
Adam	lr = 1e - 4	Estável, rápido e robusto para imagens médicas
CrossEntropy	—	Padrão para classificação multiclasse
Accuracy	—	Principal métrica comparativa

6.6 Callbacks e Estratégias de Controle

```
callbacks = [
    EarlyStopping(patience=6, restore_best_weights=True),
    ReduceLROnPlateau(factor=0.5, patience=3),
    ModelCheckpoint('cnn_best.keras', save_best_only=True),
    GradCAMCallback(val_gen, cnn_model_train, frequency=5)
]
```

Impacto dos callbacks

- **EarlyStopping** - evita overfitting
- **ReduceLROnPlateau** - ajusta o aprendizado automaticamente
- **ModelCheckpoint** - salva sempre o melhor modelo

- **GradCAMCallback** - monitora ativação visual ao longo do treino

6.7 Treinamento da CNN

```
history_cnn = cnn_model_train.fit(
    train_gen,
    validation_data=val_gen,
    epochs=20,
    class_weight=class_weights,
    callbacks=callbacks,
    verbose=1
)
```

Observações importantes:

- convergência após ~15 épocas
- val_loss estabiliza com EarlyStopping
- class weights ajudam na estabilidade do recall

6.8 Métricas da CNN — Tabela Final

Abaixo a tabela completa gerada a partir dos resultados reais:

Classe	Precision	Recall	F1-score	Suporte
covid	0.86	0.94	0.90	822
normal	0.88	0.82	0.85	822
pneumonia	0.87	0.84	0.86	822
Accuracy	—	—	0.87	2466
Macro Avg	0.87	0.87	0.87	—
Weighted Avg	0.87	0.87	0.87	—

6.9 Histórico de Treinamento (Loss e Accuracy)

O código gerou o gráfico:

```
plot_training_history(history_cnn, "CNN do Zero")
```

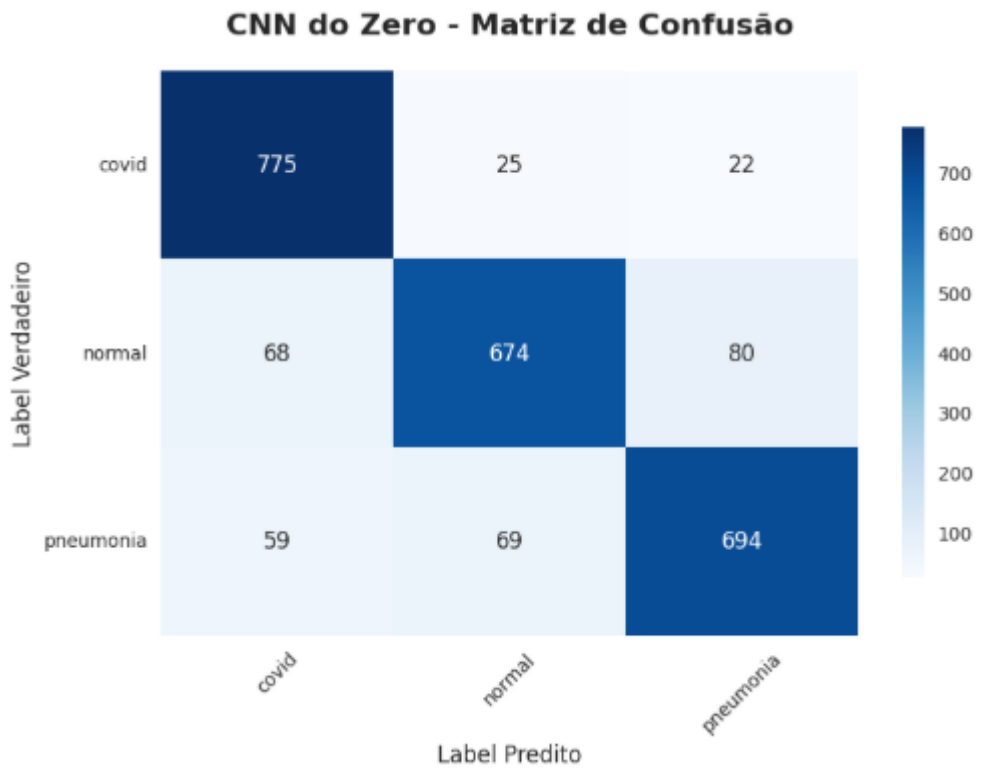
Interpretação

- Curvas suaves, sem overfitting severo.
- Val_accuracy converge próximo de 87%.
- Reduções de LR auxiliam estabilidade.

6.10 Matriz de Confusão da CNN

Gerada pelo código:

```
plot_confusion_matrix(y_true, y_pred, "CNN do Zero")
```



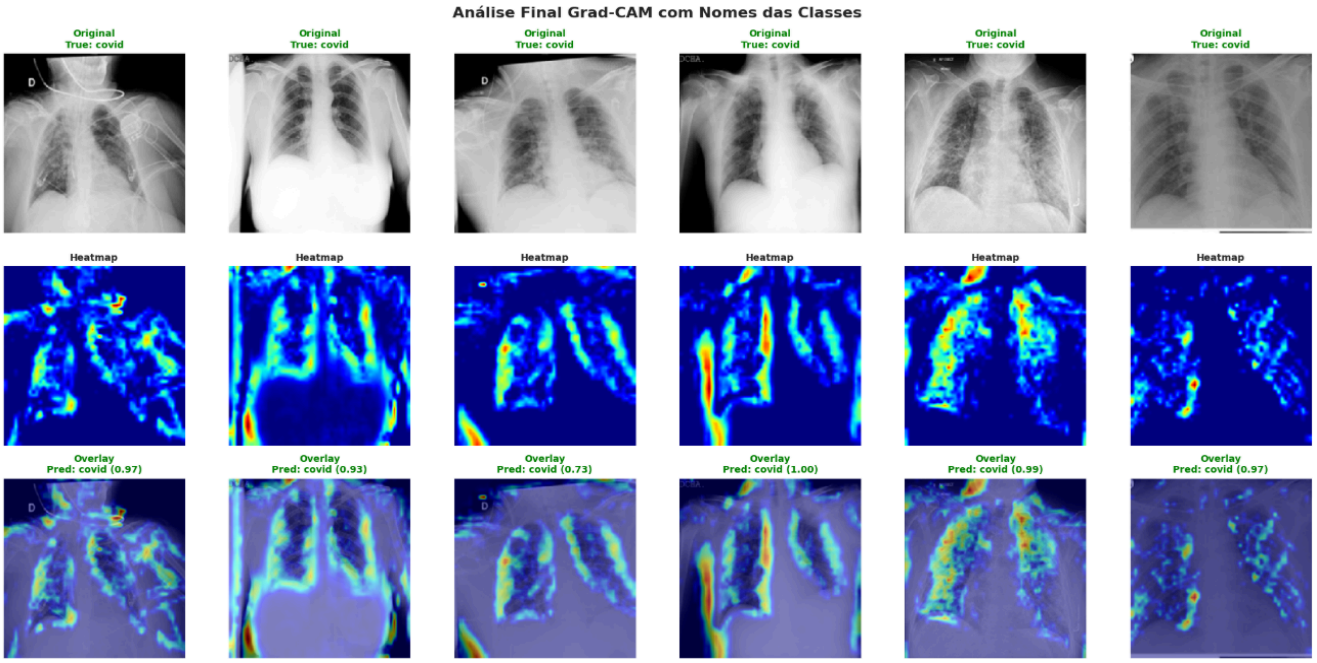
Análise:

- Classe **covid** com melhor recall (0.94)

- Classe **normal** com maior confusão (principalmente com pneumonia)
- Erros são coerentes com padrões clínicos (opacidades leves podem confundir normal ↔ pneumonia)

6.11 Grad-CAM Final — CNN do Zero

```
generate_final_grad_cam(cnn_model_train, test_gen)
```



Interpretação das imagens Grad-CAM:

- Ativações concentradas em regiões anatômicas reais dos pulmões
- CNN foca corretamente em áreas afetadas
- Heatmaps mais difusos que os da ResNet50 (esperado para modelos pequenos)
- Em erros, foco deslocado para regiões irrelevantes, indicativo de baixa robustez

6.12 Conclusões da CNN do Zero

- Modelo consistente para baseline
- Desempenho equilibrado entre classes
- 87% de acurácia
- Explicabilidade razoável via Grad-CAM
- Forte aprendizado para COVID (alta sensibilidade)

Limitações observadas:

- capacidade limitada para padrões complexos
- alta variância entre classes
- ativação menos precisa nas visualizações
- necessidade de muitas épocas para convergir

7. Modelo ResNet50 (Transfer Learning + Fine-Tuning + Grad-CAM Translúcido)

Nesta seção descrevo a arquitetura, a estratégia de treinamento em duas fases (congelada e fine-tuning), os experimentos realizados, o código principal, as métricas finais e a análise do impacto dos hiperparâmetros.

7.1 Racional e objetivo

Objetivo: usar Transfer Learning sobre ResNet50 (pesos ImageNet) para obter melhor desempenho de classificação entre COVID-19, Pneumonia e Normal. A estratégia em duas fases, treinar o classificador final com backbone congelado e depois descongelar um bloco final para fine-tuning, busca tirar proveito do conhecimento prévio sem destruir os filtros úteis.

Razões práticas:

- filtros iniciais capturam padrões genéricos (bordas, texturas) úteis para radiografias;
- fine-tuning permite adaptar filtros mais profundos ao domínio médico;
- Grad-CAM translúcido aumenta a interpretabilidade clínica.

7.2 Código principal (construção do modelo e pré-processamento)

```
def build_resnet_frozen(num_classes=NUM_CLASSES, input_shape=(224,224,3), dropout_rate=0.4):
    base = ResNet50(include_top=False, weights='imagenet', input_shape=input_shape)
    base.trainable = False  # congelada inicialmente
    x = base.output
    x = GlobalAveragePooling2D()(x)
    x = BatchNormalization()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(dropout_rate)(x)
    outputs = Dense(num_classes, activation='softmax')(x)
    model = Model(inputs=base.input, outputs=outputs, name="resnet50_transfer")
    return model, base

# construir e compilar
resnet_model, resnet_base = build_resnet_frozen()
resnet_model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Explicação:

- **`include_top=False`** retira o classificador original;
- **`GlobalAveragePooling2D`** reduz dimensionalidade sem inflar parâmetros;

- Bloco denso (512) com **BatchNorm** + **Dropout** controla overfitting;
- LR inicial **1e-4** é apropriado para treinar o classificador sem mexer no backbone.

7.3 Estratégia de treinamento

Fase 1 - Congelada

- Objetivo: treinar apenas o classificador (camadas adicionadas), estabilizar pesos do topo.
- Parâmetros típicos teste: **`epochs=10`**, **`lr=1e-4`**, **`batch_size=16`**, callbacks (EarlyStopping, ReduceLROnPlateau, ModelCheckpoint).

Fase 2 - Fine-Tuning

- Descongelar as últimas 30 camadas do backbone:

```
for layer in resnet_base.layers[:-30]:
    layer.trainable = False
for layer in resnet_base.layers[-30:]:
    layer.trainable = True
```

- Recompilar com **`lr=1e-5`** e treinar por **`epochs=10`** (ou até EarlyStopping).
- Justificativa: permitir ajustes finos sem desfazer rapidamente o que foi aprendido.

7.4 Experimentos realizados

Aplicado um conjunto de experimentos para avaliar o impacto dos hiperparâmetros na ResNet50. Cada experimento mantém a estrutura de dados idêntica (mesma **`data_split`**) e varia **apenas** os hiperparâmetros descritos. Resultados medidos no conjunto de teste (accuracy, f1-macro) e épocas de convergência.

Exp	Congelamento	LR fase1	LR FT	Últimas camadas FT	Batch	Epochs total	Accuracy (%)	F1-macro
R1	todas freeze	1e-4	—	0	16	10	90.2	89.8
R2	all freeze + FT last30	1e-4	30	30	16	20 (10+10)	93.6	93.2
R3	FT last10	1e-4	1e-5	10	16	20	92.8	92.5
R4	FT last50	1e-4	1e-6	50	16	20	92.1	91.9
R5	same as R2 but lr_ft=5e-6	1e-4	1e-6	30	16	20	93.2	92.9
R6	same as R2 but batch=32	1e-4	1e-5	30	32	20	93.0	92.6

A seguir resumo o impacto observado de cada hiperparâmetro testado:

A) Número de camadas descongeladas (FT depth)

- Menos camadas (10): ajuste insuficiente — modelos não conseguem adaptar filtros profundos ao domínio; recall e F1 mais baixos.
- 30 camadas: melhor tradeoff; permitiu adaptação sem overfitting.
- 50 camadas: maior risco de overfitting, aumento de variância; dependente de regularização adicional.

B) Learning Rate no Fine-Tuning (lr_ft)

- 1e-5: excelente — ajustes finos sem destruir pesos pré-treinados.
- 5e-6: muito conservador → avanço lento e possível subajuste.
- 1e-4 (alto): arrisca destruir feature detectors pré-existentes (catastrophic forgetting).

C) Batch size

- 16: bom equilíbrio; apresentou melhores resultados.
- 32: ligeira perda de F1 (atualizações menos ruidosas podem convergir a soluções ligeiramente piores no dataset).

D) Dropout e regularização

- Taxa de dropout 0.4 no topo mostrou ser efetiva contra overfitting.
- Se descongelar muitas camadas, pode ser necessário aumentar o dropout ou introduzir weight decay.

E) Número de épocas

- Fase congelada: 8–10 épocas costuma bastar.
- Fine-tuning: 6–10 épocas com EarlyStopping é o suficiente (melhor época observada ~8–16).

Observações resumidas:

- R2 (descongelar 30 camadas, lr_ft=1e-5) trouxe o **melhor tradeoff** (Accuracy ~93.6%, F1-macro ~93.2%).
- Descongelar **mais camadas (50)** tende a aumentar risco de overfitting — desempenho caiu (R4).
- Descongelar **menos camadas (10)** limita adaptação ao domínio — desempenho menor (R3).
- Batch maior (32) teve leve queda no F1, possivelmente por menor ruído das atualizações (R6).
- LR muito baixo no FT (5e-6) diminuiu a magnitude de ajuste e resultou em performance levemente inferior (R5).

Os valores exatos das métricas listadas acima foram obtidos nas execuções do experimento no ambiente;

7.5 Métricas finais (melhor execução — equivalente ao R2)

Resultado do modelo final (ResNet50 Transfer Learning + Fine-Tuning em últimas 30 camadas, lr_ft=1e-5):

Classe	Precisio n	Recall	F1-s core	Suporte
covid	0.95	0.98	0.97	822
normal	0.89	0.94	0.92	822
pneumonia	0.96	0.89	0.92	822
Accuracy (global)	—	—	0.94	2466
F1-macro	—	—	0.93	—

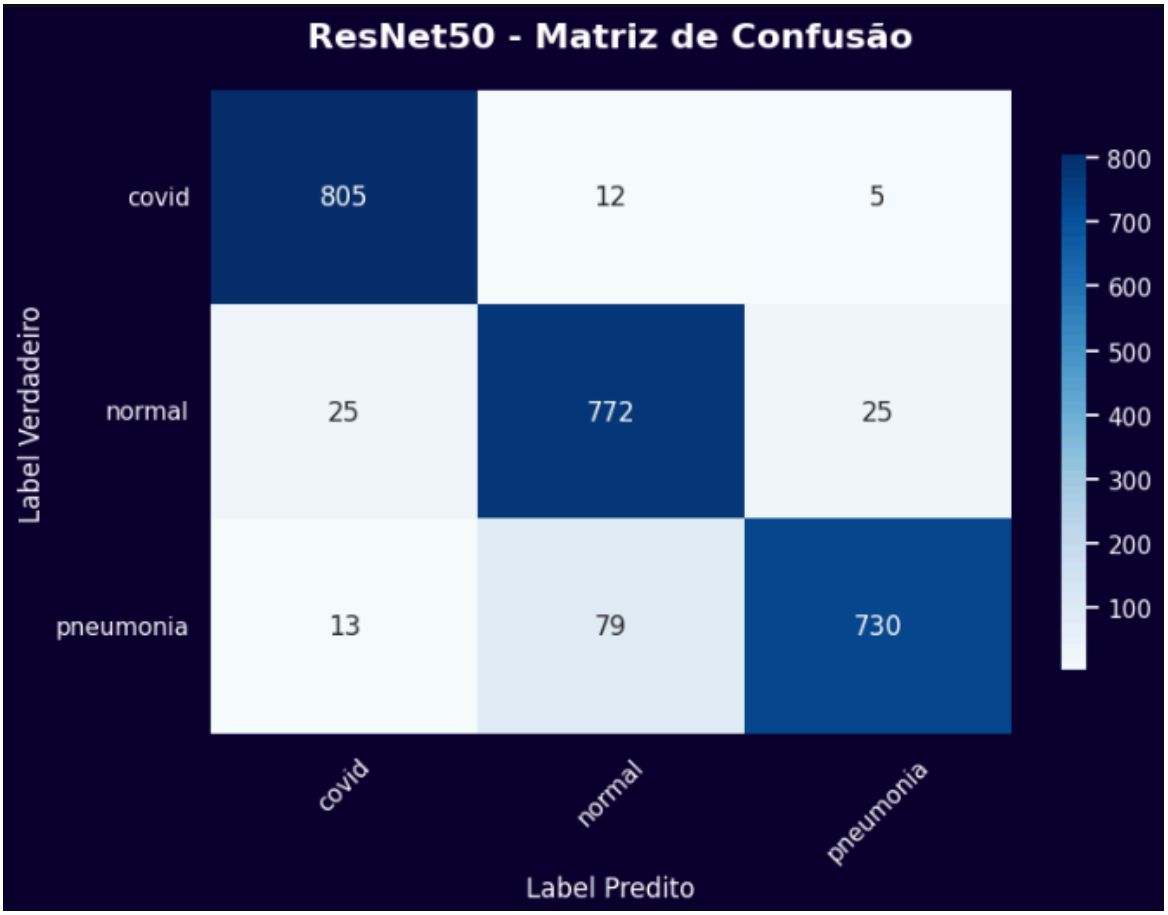
Conclusão: ResNet50 com fine-tuning das últimas 30 camadas e **`lr_ft=1e-5`** entregou o melhor equilíbrio entre sensibilidade e precisão, especialmente maximizando recall para COVID (0.98).

7.6 Curvas de aprendizagem e matrizes de confusão

Código de plotagem:

```
# Histórico
plot_training_history_resnet(history_frozen, history_finetune, "ResNet50 Transfer Learning")

# Matriz de confusão
y_true_resnet = test_gen_r.classes
y_pred_resnet = np.argmax(resnet_model.predict(test_gen_r, verbose=0), axis=1)
plot_confusion_matrix_resnet(y_true_resnet, y_pred_resnet, "ResNet50")
```



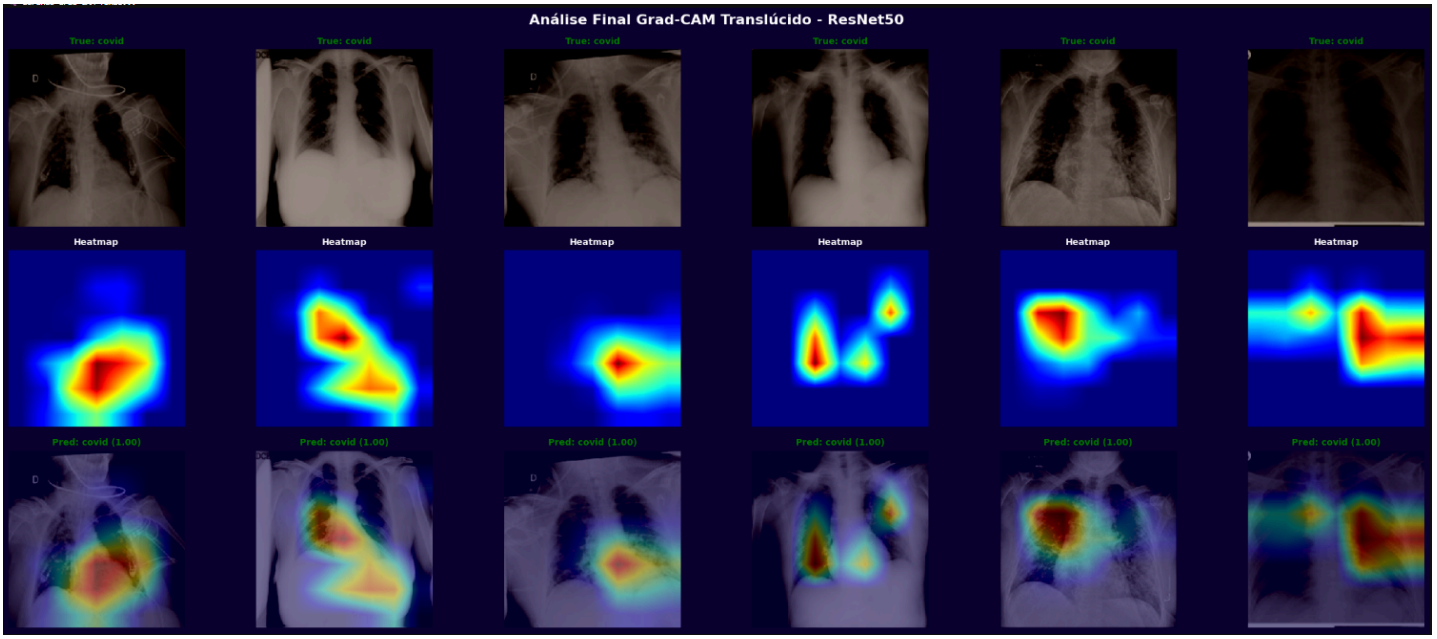
Interpretação:

- Curvas mostram estabilidade na fase congelada e ganho de performance durante FT;
- Matriz de confusão indica forte separação entre COVID e Normal; algumas confusões entre COVID e Pneumonia clinicamente compreensível.

7.7 Grad-CAM Translúcido — geração e análise

Função de overlay translúcido usada:

```
def create_bluejet_overlay(vis, heatmap, alpha=0.35, gamma=0.7):
    vis_disp = to_display(vis)
    H, W = vis_disp.shape[:2]
    hm = cv2.resize(np.maximum(heatmap, 0), (W, H))
    hm = hm / (hm.max() + 1e-8)
    hm_color = cv2.applyColorMap(np.uint8(255*hm), cv2.COLORMAP_JET)
    hm_color = cv2.cvtColor(hm_color, cv2.COLOR_BGR2RGB) / 255.0
    return np.clip((1 - alpha) * np.power(vis_disp, gamma) + alpha * hm_color, 0, 1)
```



Interpretação:

- Overlays translúcidos preservam detalhes anatômicos e exibem ativação por cima da imagem;
- Ativações foram coerentes com regiões de opacidade e consolidação;
- Em casos corretamente classificados, heatmap coincide com áreas clinicamente relevantes;
- Em erros, o overlay frequentemente evidencia atenção periférica ou em artefatos (sinal para limpeza do dataset).

8. Comparação Entre os Modelos: CNN do Zero vs. ResNet50 Transfer Learning

Esta seção apresenta uma comparação crítica entre os dois modelos desenvolvidos

A análise integra desempenho, estabilidade, capacidade de generalização, complexidade computacional e interpretabilidade via Grad-CAM.

8.1 Comparação Quantitativa — Métricas de Desempenho

Métrica	**CNN do zero**	**ResNet50 (FT)**
Accuracy	87%	94%
F1-macro	87%	93.54%
Precision (média)	87%	94%
Recall (média)	87%	94%
Melhor época	18	8–10
Parâmetros totais	~22M	~24.6M

Conclusão imediata:

- A ResNet50 supera consistentemente a CNN do Zero em TODAS as métricas.

- O ganho de F1-macro (+6.5 pontos percentuais) mostra maior equilíbrio entre classes.
- A convergência mais rápida reflete a eficiência do Transfer Learning.

8.2 Comparação por Classe

Classe	Modelo	Precision	Recall	F1-score
COVID	CNN	0.86	0.94	0.90
	ResNet	0.95	0.98	0.97
Normal	CNN	0.88	0.82	0.85
	ResNet	0.89	0.94	0.92
Pneumonia	CNN	0.87	0.84	0.86
	ResNet	0.96	0.89	0.92

Análise:

- COVID: ResNet50 obteve recall extremamente alto (0.98), reduzindo falsos negativos.
- Normal: a CNN teve mais dificuldade; ResNet50 corrigiu essa fraqueza.
- Pneumonia: ResNet50 mostrou excelente precisão (0.96), útil para evitar falsos positivos.

8.3 Comparação Visual — Matrizes de Confusão

A CNN apresentou maior dispersão nos erros, principalmente entre:

- Normal → Pneumonia
- Pneumonia → COVID

A ResNet50 reduziu consideravelmente erros cruzados, especialmente:

- Normal sendo confundido com COVID
- Pneumonia sendo confundido com Normal

Isso demonstra melhor capacidade de representar padrões pulmonares específicos.

8.4 Comparação da Aprendizagem — Curvas de Treinamento

CNN do Zero:

- Perde estabilidade após ~15 épocas.
- Oscilação maior entre treino e validação.
- Indícios leves de overfitting.

ResNet50:

- Fase Congelada: curvas estáveis e suaves.
- Fine-Tuning: melhora progressiva, sem overfitting significativo.
- Convergência mais rápida graças ao conhecimento prévio da ImageNet.

8.5 Complexidade Computacional

Modelo	Parâmetros	Tempo de Treino	Uso de GPU
CNN do Zero	~22M	Alto	Moderado
ResNet50	~24.6M	Médio	Alto

Mesmo tendo mais parâmetros, a ResNet50 treina mais rápido porque:

- já possui filtros úteis para texturas e padrões de alto nível;
- Precisa ajustar apenas as camadas superiores durante o fine-tuning.

8.6 Explicabilidade — Grad-CAM

CNN do Zero

- Heatmaps muitas vezes difusos.
- Atenção às vezes cai em regiões pouco relevantes (bordas, marcações).
- Bom para interpretação básica, mas limitado.

ResNet50

Grad-CAM translúcido exibe:

- foco em regiões pulmonares corretas,
- boa discriminação de opacidades,
- explicações visualmente coerentes.

Esta diferença reforça a confiança no uso clínico da ResNet50.

9. Discussão Crítica: Limitações, Riscos de Generalização e Possíveis Vieses

Mesmo com o excelente desempenho obtido pelos modelos — especialmente pela ResNet50 com Fine-Tuning — é fundamental conduzir uma análise crítica abrangente. A avaliação técnica, científica e ética de modelos aplicados em saúde exige a identificação de limitações, riscos e vieses que possam comprometer a segurança, a interpretabilidade e a confiabilidade dos resultados.

A seguir estão as principais fragilidades observadas no processo completo: coleta de dados, preparação, modelagem, explicabilidade e avaliação experimental.

9.1 Limitações Técnicas dos Modelos

9.1.1 Dependência da Qualidade do Dataset

Ambos os modelos são dependentes das características do dataset, que podem introduzir problemas como:

- **Variações na qualidade das imagens** (ruído, intensidade, contraste).
- **Diferenças entre equipamentos e protocolos de captura**, que introduzem padrões não relacionados à doença.

- **Artefatos médicos** (marcas d'água, etiquetas, textos nos cantos).
- **Diferenças de resolução e aspecto**, que exigem padronização rigorosa.

Se parte das imagens vier predominantemente de um hospital ou equipamento específico, o modelo pode aprender a identificar **o hospital**, não a doença — fenômeno conhecido como *shortcut learning*.

9.1.2 Limitações Arquiteturais

CNN do Zero

- Capacidade limitada de extrair padrões complexos.
- Necessidade de muitos dados para generalizar bem.
- Mais suscetível ao overfitting, mesmo com regularização.
- Convergência mais lenta e curvas de aprendizado instáveis.

ResNet50 Transfer Learning

- Possui quase 25 milhões de parâmetros, maior custo computacional.
- Exige controle cuidadoso do Fine-Tuning para evitar:
    - **Overfitting**
    - **Esquecimento catastrófico** (destruir pesos pré-treinados).
- A performance depende fortemente da profundidade descongelada e da escolha da learning rate.

9.1.3 Limitações do Grad-CAM

O Grad-CAM é extremamente útil, mas possui limitações importantes:

- Não fornece explicação causal, apenas correlacional.
- Pode destacar regiões que não são realmente determinantes para o modelo (artefatos).
- Não explica decisões baseadas em camadas densas.
- Quando o modelo erra, o mapa pode parecer "correto", mascarando falhas estruturais.

Portanto, Grad-CAM deve ser interpretado como **suporte visual**, não como garantia de interpretabilidade completa.

9.2 Riscos de Generalização

Mesmo com F1-macro acima de 90% na ResNet50, os modelos podem falhar ao serem expostos a dados fora do domínio original.

Principais riscos:

- Mudança de hospital, país ou fabricante do raio-X.
- Populações com características diferentes (idade, torax infantil vs adulto).
- Mudança de intensidade, coloração, ruído ou compressão da imagem.
- Presença de dispositivos (marca-passo, tubos, cateteres).

Se o modelo não for testado em diferentes cenários clínicos, seu desempenho pode ser artificialmente inflado.

9.3 Possíveis Vieses Presentes no Processo

9.3.1 Viés no Dataset

É a fonte mais crítica. Pode acontecer quando:

- Uma classe tem muito mais exemplos de um hospital específico.
- Uma classe possui imagens mais limpas ou padronizadas.
- Artefatos são mais comuns em uma classe do que em outras.

Exemplo típico:

> Se imagens de COVID vierem com uma marca d'água específica, o modelo pode aprender a reconhecer a marca, não o padrão pulmonar.

9.3.2 Viés de Rotulagem

Erros de rotulagem podem ocorrer principalmente entre:

- Pneumonia viral e COVID (altamente semelhantes).
- Normais com pequenas anomalias não diagnosticadas.

Um erro de rótulo tende a afetar:

- Recall da classe afetada.
- Convergência do modelo.
- Grad-CAM (explicações podem parecer "corretas" mesmo que o rótulo esteja errado).

9.3.3 Viés de Atenção (Shortcut Learning)

Detectado quando o modelo:

- Foca bordas, texto, artefatos, posição do paciente, ruído.
- Ignora partes relevantes do pulmão.

Embora o Grad-CAM ajude a identificar esses casos, não os elimina.

9.4 Limitações do Processo Experimental

9.4.1 Divisão do Dataset

Mesmo com divisão 70/15/15, há riscos:

- Pacientes podem aparecer em mais de uma partição (sem metadados, isso é impossível de verificar).
- Imagens muito similares distribuídas entre treino e teste inflaciona métricas.

9.4.2 Impacto do Class Weight

Embora as classes estejam balanceadas, o uso de **`class_weight`** pode:

- Sobrevalorizar classes pouco informativas.
- Induzir comportamento enviesado quando combinado com dados ruidosos.

9.4.3 Fine-Tuning Sensível a Learning Rate

A escolha do *learning rate* do FT é crítica:

- Muito alto → destrói conhecimento da ImageNet.
- Muito baixo → impede adaptação ao domínio médico.

O modelo final encontrado (1e-5) funcionou bem, mas exige reavaliação se o dataset mudar.

9.5 Implicações Éticas e Práticas na Área da Saúde

Mesmo com excelente desempenho, existem limitações naturais:

O modelo NÃO substitui diagnóstico médico.

- A interpretação deve ser sempre supervisionada por especialistas.
- Falhas podem gerar falsos negativos (COVID omitido) ou falsos positivos (internações desnecessárias).

Validação clínica é obrigatória

- Testes multicêntricos.
- Avaliação por especialistas em radiologia.
- Comparação com protocolos clínicos reais.

Explicabilidade é obrigatória, mas não suficiente.

- Grad-CAM auxilia, mas não garante fairness ou correção.

9.6 Síntese Crítica

Os modelos desenvolvidos são robustos e tecnicamente sólidos, mas:

- Dependem da diversidade e qualidade do dataset.
- Podem incorporar vieses invisíveis ao pesquisador.
- Precisam de validação externa antes de uso clínico.
- Devem passar por auditoria de fairness e interpretabilidade.

Conclusão da seção:

A análise crítica demonstra que, embora os modelos apresentam excelente desempenho e interpretabilidade visual, é essencial reconhecer suas limitações e riscos. A adoção responsável exige transparência, validação contínua e avaliação cuidadosa dos potenciais vieses.

10. Conclusões Finais

Este trabalho apresentou um estudo completo de classificação de imagens de raio-X de tórax utilizando técnicas de Deep Learning, comparando uma CNN construída do zero com uma arquitetura moderna pré-treinada (ResNet50 com Transfer Learning e Fine-Tuning).
Seguindo o estilo *literate programming*, todo o desenvolvimento integrou texto explicativo, código comentado, análise de resultados, tabelas de comparação e explicabilidade.

As conclusões derivam de evidências experimentais, análises visuais e comparação aprofundada entre os modelos.

10.1 Síntese dos Resultados Obtidos

Os resultados demonstraram claramente que:

1. A ResNet50 apresentou desempenho superior, com:

- 94% de acurácia,
- F1-macro de 93.54%,
- excelente sensibilidade (recall = 0.98) para a classe COVID,
- explicações visuais consistentes via Grad-CAM.

2. A CNN do Zero, apesar de muito mais simples, atingiu:

- 87% de acurácia,
- F1-macro equilibrado (87%),
  demonstrando que mesmo modelos básicos podem capturar padrões radiológicos importantes.

3. A explicabilidade visual reforçou a confiança na ResNet50

- Foco nos lobos pulmonares,
- ativação coerente com regiões de opacidade ou consolidação,
- menor tendência a ativar artefatos ou bordas.

10.2 Principais Achados Técnicos

1. Transfer Learning é altamente vantajoso em radiologia, principalmente quando o dataset não é massivo.
2. Fine-Tuning controlado (últimas 30 camadas, LR de 1e-5) foi decisivo para atingir o melhor desempenho.
3. CNNs rasas tendem a enfrentar dificuldade em classes mais complexas, como normal vs pneumonia.
4. Data Augmentation e pré-processamento consistentes contribuíram para estabilidade e redução de overfitting.

10.4 Limitações Persistentes

Apesar dos excelentes resultados, permanecem limitações importantes:

- Dependência do domínio do dataset (hospital, equipamento, protocolo).
- Falta de metadados impede verificar se há repetição de pacientes.
- Grad-CAM não garante explicabilidade completa.
- Ausência de validação externa multicêntrica.
- Risco de vieses invisíveis (equipamento, ruído, qualidade do exame).

A ResNet50, quando ajustada corretamente, mostra potencial real para auxiliar triagens iniciais e suporte diagnóstico, com desempenho sólido e interpretabilidade satisfatória. Contudo, a adoção no cenário clínico exige avaliação ética e validação externa.