Classificação de espécies por Visão Computacional

Um tutorial autodidata com explicação linha a linha do notebook

Repositório: https://github.com/fulvioamf/crab_vision_classifier

Autores: Fulvio Aurélio de Morais Freire & Vitor Carvalho Silva

(LABEEC/UFRN)

Versão: v16 | Data: 01/01/2026

Este material foi gerado a partir do notebook 'crab_vision_gray.ipynb' e contém: introdução teórica, guia de reprodução, interpretação de métricas e a documentação completa dos outputs do notebook.

Sumário

1. Introdução: conceitos essenciais

Este capítulo apresenta os conceitos necessários para compreender o pipeline do repositório e interpretar as métricas e figuras. O foco é permitir que um leitor de Biologia (ou áreas afins) acompanhe o raciocínio sem depender de um curso prévio em Ciência de Dados.

1.1 Inteligência Artificial, Aprendizado de Máquina e Redes Neurais

Inteligência Artificial (IA) é o campo mais amplo. Aprendizado de Máquina (Machine Learning) é um subconjunto em que modelos aprendem padrões a partir de dados, em vez de regras manuais. Redes neurais são modelos que aprendem transformações em camadas, ajustando parâmetros para minimizar um erro (função de perda).

1.2 Redes neurais profundas e CNNs

Deep Learning refere-se a redes com muitas camadas. Para imagens, a arquitetura mais usada é a Rede Neural Convolucional (CNN). Convoluções capturam padrões locais (bordas, texturas) e, em camadas profundas, combinam esses padrões em estruturas mais ricas (partes do organismo e formas). O classificador final transforma essas representações em probabilidades por classe.

1.3 Visão Computacional e classificação de espécies

Visão Computacional aplica IA para extrair informação de imagens (detecção, segmentação, classificação e outras tarefas).

Classificação de espécies é frequentemente um problema de grão fino: espécies próximas podem ter diferenças sutis. Por isso, o modelo pode confundir classes com morfologia semelhante, ou aprender atalhos indesejados (por exemplo, fundo/iluminação).

Implicações práticas para o dataset: padronização de aquisição (fundo, escala, orientação), curadoria de imagens, balanceamento de classes, splits bem definidos (treino/validação/teste) e controle de vazamento de dados (mesmo indivíduo/mesma sessão em splits diferentes).

Por que interpretabilidade importa: técnicas como Grad-CAM ajudam a verificar onde o modelo está “olhando”. Idealmente, regiões anatômicas relevantes devem ser destacadas; foco no fundo sugere viés e exige ajustes de dados/treino.

1.4 Transfer Learning, fine-tuning e arquiteturas (ResNet, AlexNet e MobileNet)

Transfer learning reaproveita um modelo pré-treinado (tipicamente em ImageNet) e o adapta ao seu problema. Em geral, substitui-se a camada final para o número de classes e treina-se esse “cabeçalho”; em seguida, pode-se fazer fine-tuning (descongelar parte do backbone) para ajustar a rede ao domínio biológico.

Principais diferenças entre arquiteturas:

Arquitetura	Ideia central	Custo computacional	Quando faz mais sentido
AlexNet	CNN clássica (mais antiga), com poucas camadas convolucionais e camadas totalmente conectadas no final.	Baixo a médio.	Didática e rápida para protótipos; baseline histórico. Em grão fino, tende a performar pior que redes modernas.

Arquitetura	Ideia central	Custo computacional	Quando faz mais sentido
ResNet (ex.: ResNet18/50)	Conexões residuais (skip connections) permitem redes mais profundas com treino estável.	Médio a alto (depende da profundidade).	Ótima escolha geral para transfer learning em imagens biológicas: desempenho e robustez, bom fine-tuning.
MobileNet (V2/V3)	Eficiência por convoluções separáveis em profundidade e otimizações para modelos leves.	Baixo.	Quando há restrição de GPU/tempo ou necessidade de inferência rápida; bom custo-benefício.

No notebook, arquiteturas como ResNet/AlexNet/MobileNet (ou outras disponíveis em torchvision.models) podem ser usadas como backbone. A escolha depende do tamanho do dataset, do hardware disponível e do objetivo (máxima acurácia vs. eficiência). Para resultados de artigo, registre arquitetura, seed e hiperparâmetros.

2. Como se orientar no repositório

Este repositório disponibiliza um notebook principal, um arquivo de dependências e um conjunto reduzido de dados para reprodução do fluxo (o dataset completo não é publicado neste momento).

2.1 Mapa do repositório

Item	Função
crab_vision_gray.ipynb	Notebook principal (pipeline completo).
requirements.txt	Dependências para reprodução do ambiente.
siris_dataset_split/	Dataset reduzido no padrão ImageFolder (pastas por classe/split).
README.md	Descrição do projeto e orientação ao visitante.

2.2 Quickstart: reprodução em 5 passos

1) Clone o repositório. 2) Crie e ative um ambiente virtual. 3) Instale as dependências (requirements.txt). 4) Abra o notebook e execute as células na ordem. 5) Interprete as saídas finais (loss/acc, matriz de confusão, ROC e Grad-CAM).

2.3 Boas práticas para reprodutibilidade

Fixe seeds, registre versões (Python, torch, torchvision), documente hiperparâmetros (lr, batch size, épocas) e mantenha um histórico de experimentos. Para reportar resultados finais (artigo), substitua exemplos por números do experimento final e inclua figuras finais já no padrão de publicação.

3. Como interpretar resultados (métricas e figuras)

Este guia resume como ler as principais saídas do notebook. No tutorial linha a linha, cada figura/tabela é explicada no ponto em que aparece.

3.1 Loss e acurácia (curvas de aprendizagem)

Use para diagnosticar convergência e overfitting. Se treino melhora e teste estagna/piora, investigue regularização e dados.

3.2 Matriz de confusão

Diagonal = acertos; fora da diagonal = confusões. Ajuda a identificar espécies confundidas e orientar coleta/curadoria de imagens.

3.3 Precision, recall e F1-score

F1 é útil em desbalanceamento. Baixo recall indica que a classe está sendo “perdida”. Baixa precision indica muitos falsos positivos.

3.4 ROC/AUC em multi-classe

One-vs-rest por classe. AUC próxima de 1 sugere boa separabilidade; próxima de 0.5 sugere baixa discriminação.

3.5 Grad-CAM (interpretabilidade)

Verifica onde o modelo está “olhando”. Idealmente destaca regiões anatômicas; foco no fundo pode indicar viés.

4. Tutorial linha a linha do notebook

A partir daqui, cada célula do notebook é documentada em sequência. Para cada célula: (i) tabela com código e explicação linha a linha; (ii) resultados gerados e interpretação.

4.1. Imports e configurações iniciais

Código e explicação linha a linha

Linha	Código	Explicação
1	`import numpy as np`	Importa bibliotecas necessárias para o pipeline.
2	`import torch`	Importa bibliotecas necessárias para o pipeline.
3	`import torch.nn as nn`	Importa bibliotecas necessárias para o pipeline.
4	`import torch.optim as optim`	Importa bibliotecas necessárias para o pipeline.
5	`from torchvision import datasets, transforms, models`	Importa bibliotecas necessárias para o pipeline.
6	`from torch.utils.data import DataLoader`	Importa bibliotecas necessárias para o pipeline.
7	`import os`	Importa bibliotecas necessárias para o pipeline.
8	`import matplotlib.pyplot as plt`	Importa bibliotecas necessárias para o pipeline.
9	`from sklearn.metrics import confusion_matrix`	Importa bibliotecas necessárias para o pipeline.
10	`from sklearn.metrics import classification_report`	Importa bibliotecas necessárias para o pipeline.
11	`import seaborn as sns`	Importa bibliotecas necessárias para o pipeline.
12	`import random`	Importa bibliotecas necessárias para o pipeline.
13	`from sklearn.metrics import roc_curve, auc`	Importa bibliotecas necessárias para o pipeline.
14	`from sklearn.preprocessing import label_binarize`	Importa bibliotecas necessárias para o pipeline.
15	`from sklearn.metrics import accuracy_score, f1_score`	Importa bibliotecas necessárias para o pipeline.
16		Linha em branco; separa blocos lógicos.
17	`SEED = 42`	Atribuição: define/atualiza variável ou objeto.
18	`random.seed(SEED)`	Instrução do script no contexto do bloco.
19	`np.random.seed(SEED)`	Instrução do script no contexto do bloco.
20	`torch.manual_seed(SEED)`	Instrução do script no contexto do bloco.
21	`torch.cuda.manual_seed(SEED)`	Instrução do script no contexto do bloco.
22	`torch.backends.cudnn.deterministic = True`	Atribuição: define/atualiza variável ou objeto.
23	`torch.backends.cudnn.benchmark = False`	Atribuição: define/atualiza variável ou objeto.

4.2. import os

Código e explicação linha a linha

Linha	Código	Explicação
1	`# import os`	Comentário; documenta a intenção do bloco (não executa).
2	`# import shutil`	Comentário; documenta a intenção do bloco (não executa).
3		Linha em branco; separa blocos lógicos.
4	`# def mover_imagens_para_pasta_raiz(dataset_path):`	Comentário; documenta a intenção do bloco (não executa).
5	`# for classe in os.listdir(dataset_path):`	Comentário; documenta a intenção do bloco (não executa).
6	`# classe_path = os.path.join(dataset_path, classe)`	Comentário; documenta a intenção do bloco (não executa).
7		Linha em branco; separa blocos lógicos.
8	`# # Pular se não for pasta`	Comentário; documenta a intenção do bloco (não executa).
9	`# if not os.path.isdir(classe_path):`	Comentário; documenta a intenção do bloco (não executa).
10	`# continue`	Comentário; documenta a intenção do bloco (não executa).
11		Linha em branco; separa blocos lógicos.
12	`# for raiz, subdirs, arquivos in os.walk(classe_path):`	Comentário; documenta a intenção do bloco (não executa).
13	`# for arquivo in arquivos:`	Comentário; documenta a intenção do bloco (não executa).
14	`# if arquivo.lower().endswith(('.jpg', '.jpeg', '.png')):`	Comentário; documenta a intenção do bloco (não executa).
15	`# origem = os.path.join(raiz, arquivo)`	Comentário; documenta a intenção do bloco (não executa).
16	`# destino = os.path.join(classe_path, arquivo)`	Comentário; documenta a intenção do bloco (não executa).
17		Linha em branco; separa blocos lógicos.
18	`# # Evitar sobrescrever se nome repetir`	Comentário; documenta a intenção do bloco (não executa).
19	`# base, ext = os.path.splitext(arquivo)`	Comentário; documenta a intenção do bloco (não executa).
20	`# contador = 1`	Comentário; documenta a intenção do bloco (não executa).
21	`# while os.path.exists(destino):`	Comentário; documenta a intenção do bloco (não executa).
22	`# destino = os.path.join(classe_path, f"{base}_{contador}{ext}")`	Comentário; documenta a intenção do bloco (não executa).
23	`# contador += 1`	Comentário; documenta a intenção do bloco (não executa).
24		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
25	`# shutil.copy2(origem, destino)`	Comentário; documenta a intenção do bloco (não executa).
26		Linha em branco; separa blocos lógicos.
27	`# print("■ Imagens copiadas para o nível da classe com sucesso!")`	Comentário; documenta a intenção do bloco (não executa).
28		Linha em branco; separa blocos lógicos.
29	`# # Use o caminho onde estão as classes`	Comentário; documenta a intenção do bloco (não executa).
30	`# mover_imagens_para_pasta_raiz("siris_dataset")`	Comentário; documenta a intenção do bloco (não executa).

4.3. Caminhos

Código e explicação linha a linha

Linha	Código	Explicação
1	`import os`	Importa bibliotecas necessárias para o pipeline.
2	`import shutil`	Importa bibliotecas necessárias para o pipeline.
3	`import random`	Importa bibliotecas necessárias para o pipeline.
4	`import cv2 as cv`	Importa bibliotecas necessárias para o pipeline.
5	`# Caminhos`	Comentário; documenta a intenção do bloco (não executa).
6	`original_dataset = "siris_dataset"`	Atribuição: define/atualiza variável ou objeto.
7	`destino_base = "siris_dataset_split"`	Atribuição: define/atualiza variável ou objeto.
8	`train_path = os.path.join(destino_base, "train")`	Atribuição: define/atualiza variável ou objeto.
9	`test_path = os.path.join(destino_base, "test")`	Atribuição: define/atualiza variável ou objeto.
10	`external_path = os.path.join(destino_base, "external")`	Atribuição: define/atualiza variável ou objeto.
11	`split_train = 0.7`	Atribuição: define/atualiza variável ou objeto.
12	`split_test = 0.2`	Atribuição: define/atualiza variável ou objeto.
13	`split_external = 0.1 # 10% para avaliação final`	Atribuição: define/atualiza variável ou objeto.
14		Linha em branco; separa blocos lógicos.
15		Linha em branco; separa blocos lógicos.
16	`# Extensões permitidas`	Comentário; documenta a intenção do bloco (não executa).
17	`extensoes_validas = (".jpg", ".jpeg", ".png", ".bmp")`	Atribuição: define/atualiza variável ou objeto.
18		Linha em branco; separa blocos lógicos.
19	`# Cria diretórios de saída`	Comentário; documenta a intenção do bloco (não executa).
20	`os.makedirs(train_path, exist_ok=True)`	Atribuição: define/atualiza variável ou objeto.
21	`os.makedirs(test_path, exist_ok=True)`	Atribuição: define/atualiza variável ou objeto.
22	`os.makedirs(external_path, exist_ok=True)`	Atribuição: define/atualiza variável ou objeto.
23		Linha em branco; separa blocos lógicos.
24	`# Para cada classe no dataset`	Comentário; documenta a intenção do bloco (não executa).
25	`for classe in os.listdir(original_dataset):`	Inicia um laço de iteração.
26	`classe_path = os.path.join(original_dataset, classe)`	Atribuição: define/atualiza variável ou objeto.

4.3. Caminhos (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
27	`if not os.path.isdir(classe_path):`	Estrutura condicional.
28	`continue`	Instrução do script no contexto do bloco.
29		Linha em branco; separa blocos lógicos.
30	`imagens = [f for f in os.listdir(classe_path) if f.lower().endswith(extensoes_validas)]`	Atribuição: define/atualiza variável ou objeto.
31	`random.shuffle(imagens)`	Instrução do script no contexto do bloco.
32		Linha em branco; separa blocos lógicos.
33	`total = len(imagens)`	Atribuição: define/atualiza variável ou objeto.
34	`train_end = int(total * split_train)`	Atribuição: define/atualiza variável ou objeto.
35	`test_end = train_end + int(total * split_test)`	Atribuição: define/atualiza variável ou objeto.
36		Linha em branco; separa blocos lógicos.
37	`imagens_train = imagens[:train_end]`	Atribuição: define/atualiza variável ou objeto.
38	`imagens_test = imagens[train_end:test_end]`	Atribuição: define/atualiza variável ou objeto.
39	`imagens_external = imagens[test_end:]`	Atribuição: define/atualiza variável ou objeto.
40		Linha em branco; separa blocos lógicos.
41	`# Criar diretórios de saída por classe`	Comentário; documenta a intenção do bloco (não executa).
42	`os.makedirs(os.path.join(train_path, classe), exist_ok=True)`	Atribuição: define/atualiza variável ou objeto.
43	`os.makedirs(os.path.join(test_path, classe), exist_ok=True)`	Atribuição: define/atualiza variável ou objeto.
44	`os.makedirs(os.path.join(external_path, classe), exist_ok=True)`	Atribuição: define/atualiza variável ou objeto.
45		Linha em branco; separa blocos lógicos.
46	`# Copiar imagens de treino`	Comentário; documenta a intenção do bloco (não executa).
47	`for img in imagens_train:`	Inicia um laço de iteração.
48	`img_gray = cv.imread(os.path.join(classe_path, img))`	Atribuição: define/atualiza variável ou objeto.
49	`img_gray = cv.cvtColor(img_gray, cv.COLOR_BGR2GRAY)`	Atribuição: define/atualiza variável ou objeto.
50	`src = os.path.join(classe_path, img)`	Atribuição: define/atualiza variável ou objeto.
51	`dst = os.path.join(train_path, classe, img)`	Atribuição: define/atualiza variável ou objeto.
52	`try:`	Instrução do script no contexto do bloco.

4.3. Caminhos (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
53	`cv.imwrite(os.path.join(train_path, classe, img), img_gray)`	Instrução do script no contexto do bloco.
54	`except Exception as e:`	Instrução do script no contexto do bloco.
55	`print(f"Erro ao copiar {src}: {e}")`	Imprime informação de acompanhamento no console.
56		Linha em branco; separa blocos lógicos.
57	`# Copiar imagens de teste`	Comentário; documenta a intenção do bloco (não executa).
58	`for img in imagens_test:`	Inicia um laço de iteração.
59	`img_gray = cv.imread(os.path.join(classe_path, img))`	Atribuição: define/atualiza variável ou objeto.
60	`img_gray = cv.cvtColor(img_gray, cv.COLOR_BGR2GRAY)`	Atribuição: define/atualiza variável ou objeto.
61	`src = os.path.join(classe_path, img)`	Atribuição: define/atualiza variável ou objeto.
62	`try:`	Instrução do script no contexto do bloco.
63	`cv.imwrite(os.path.join(test_path, classe, img), img_gray)`	Instrução do script no contexto do bloco.
64	`except Exception as e:`	Instrução do script no contexto do bloco.
65	`print(f"Erro ao copiar {src}: {e}")`	Imprime informação de acompanhamento no console.
66		Linha em branco; separa blocos lógicos.
67	`# Copiar imagens externas`	Comentário; documenta a intenção do bloco (não executa).
68	`for img in imagens_external:`	Inicia um laço de iteração.
69	`img_gray = cv.imread(os.path.join(classe_path, img))`	Atribuição: define/atualiza variável ou objeto.
70	`img_gray = cv.cvtColor(img_gray, cv.COLOR_BGR2GRAY)`	Atribuição: define/atualiza variável ou objeto.
71	`src = os.path.join(classe_path, img)`	Atribuição: define/atualiza variável ou objeto.
72	`try:`	Instrução do script no contexto do bloco.
73	`cv.imwrite(os.path.join(external_path, classe, img), img_gray)`	Instrução do script no contexto do bloco.
74	`except Exception as e:`	Instrução do script no contexto do bloco.
75	`print(f"Erro ao copiar {src}: {e}")`	Imprime informação de acompanhamento no console.
76		Linha em branco; separa blocos lógicos.
77	`print("■ Imagens divididas em treino, teste e conjunto externo com sucesso!")`	Imprime informação de acompanhamento no console.

Resultados gerados nesta célula

Saída (texto)

■ Imagens divididas em treino, teste e conjunto externo com sucesso!

4.4. 1. CONFIGURAÇÕES

Código e explicação linha a linha

Linha	Código	Explicação
1	`train_losses = []`	Atribuição: define/atualiza variável ou objeto.
2	`test_accuracies = []`	Atribuição: define/atualiza variável ou objeto.
3		Linha em branco; separa blocos lógicos.
4	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
5	`# 1. CONFIGURAÇÕES`	Comentário; documenta a intenção do bloco (não executa).
6	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
7	`device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
8	`batch_size = 32`	Atribuição: define/atualiza variável ou objeto.
9	`num_epochs = 10`	Atribuição: define/atualiza variável ou objeto.
10	`data_dir = "siris_dataset_split" # Caminho para dataset com train/ e test/`	Atribuição: define/atualiza variável ou objeto.
11		Linha em branco; separa blocos lógicos.
12	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
13	`# 2. TRANSFORMAÇÕES`	Comentário; documenta a intenção do bloco (não executa).
14	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
15	`transform = transforms.Compose([`	Cria pipeline de pré-processamento (transforms).
16	`transforms.Resize((224, 224)), # Todas as redes esperam 224x224`	Instrução do script no contexto do bloco.
17	`transforms.ToTensor(),`	Instrução do script no contexto do bloco.
18	`transforms.Normalize(mean=[0.485, 0.456, 0.406], # Padrão ImageNet`	Atribuição: define/atualiza variável ou objeto.
19	`std=[0.229, 0.224, 0.225])`	Atribuição: define/atualiza variável ou objeto.
20	`])`	Instrução do script no contexto do bloco.
21		Linha em branco; separa blocos lógicos.
22	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
23	`# 3. DATALOADERS`	Comentário; documenta a intenção do bloco (não executa).
24	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
25	`generator = torch.Generator().manual_seed(SEED)`	Atribuição: define/atualiza variável ou objeto.
26	`train_dataset = datasets.ImageFolder(os.path.join(data_dir, "train"), transform=transform)`	Carrega dataset no padrão ImageFolder (pastas = classes).

4.4. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
27	`test_dataset = datasets.ImageFolder(os.path.join(data_dir, "test"), transform=transform)`	Carrega dataset no padrão ImageFolder (pastas = classes).
28		Linha em branco; separa blocos lógicos.
29	`train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True, generator=generator)`	Cria DataLoader (batches, shuffle, paralelismo).
30	`test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False, generator=generator)`	Cria DataLoader (batches, shuffle, paralelismo).
31		Linha em branco; separa blocos lógicos.
32	`num_classes = len(train_dataset.classes)`	Atribuição: define/atualiza variável ou objeto.
33	`class_names = train_dataset.classes`	Atribuição: define/atualiza variável ou objeto.
34		Linha em branco; separa blocos lógicos.
35	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
36	`# 4. MODELO (troque aqui)`	Comentário; documenta a intenção do bloco (não executa).
37	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
38	`model = models.resnet18(pretrained=True)`	Carrega uma arquitetura CNN (em geral pré-treinada) para transfer learning.
39	`model.fc = nn.Linear(model.fc.in_features, num_classes) # Para ResNet18`	Atribuição: define/atualiza variável ou objeto.
40		Linha em branco; separa blocos lógicos.
41	`model = model.to(device)`	Atribuição: define/atualiza variável ou objeto.
42		Linha em branco; separa blocos lógicos.
43	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
44	`# 5. LOSS E OTIMIZADOR`	Comentário; documenta a intenção do bloco (não executa).
45	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
46	`criterion = nn.CrossEntropyLoss()`	Define a perda CrossEntropy (classificação multi-classe).
47	`optimizer = optim.Adam(model.parameters(), lr=0.0001)`	Define o otimizador Adam (com taxa de aprendizado).
48		Linha em branco; separa blocos lógicos.
49	`# ======================`	Comentário; documenta a intenção do bloco (não executa).

Linha	Código	Explicação
50	`# 6. TREINAMENTO`	Comentário; documenta a intenção do bloco (não executa).
51	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
52	`for epoch in range(num_epochs):`	Inicia um laço de iteração.

4.4. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
53	`model.train()`	Ativa modo de treinamento.
54	`running_loss = 0.0`	Atribuição: define/atualiza variável ou objeto.
55		Linha em branco; separa blocos lógicos.
56	`for images, labels in train_loader:`	Inicia um laço de iteração.
57	`images, labels = images.to(device), labels.to(device)`	Atribuição: define/atualiza variável ou objeto.
58		Linha em branco; separa blocos lógicos.
59	`optimizer.zero_grad()`	Instrução do script no contexto do bloco.
60	`outputs = model(images)`	Atribuição: define/atualiza variável ou objeto.
61	`loss = criterion(outputs, labels)`	Atribuição: define/atualiza variável ou objeto.
62	`loss.backward()`	Backpropagation (cálculo de gradientes).
63	`optimizer.step()`	Atualiza pesos com base nos gradientes.
64		Linha em branco; separa blocos lógicos.
65	`running_loss += loss.item()`	Atribuição: define/atualiza variável ou objeto.
66		Linha em branco; separa blocos lógicos.
67	`print(f"Época {epoch+1}, Loss: {running_loss / len(train_loader)}")`	Imprime informação de acompanhamento no console.
68		Linha em branco; separa blocos lógicos.
69	`train_losses.append(running_loss / len(train_loader))`	Instrução do script no contexto do bloco.
70		Linha em branco; separa blocos lógicos.
71	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
72	`# 7. AVALIAÇÃO NO TESTE`	Comentário; documenta a intenção do bloco (não executa).
73	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
74	`model.eval()`	Ativa modo de avaliação/inferência.
75	`correct = 0`	Atribuição: define/atualiza variável ou objeto.
76	`total = 0`	Atribuição: define/atualiza variável ou objeto.
77		Linha em branco; separa blocos lógicos.
78	`all_preds = []`	Atribuição: define/atualiza variável ou objeto.

4.4. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
79	`all_labels = []`	Atribuição: define/atualiza variável ou objeto.
80	`all_probs = []`	Atribuição: define/atualiza variável ou objeto.
81		Linha em branco; separa blocos lógicos.
82	`with torch.no_grad():`	Desativa gradientes para inferência mais rápida.
83	`for images, labels in test_loader:`	Inicia um laço de iteração.
84	`images, labels = images.to(device), labels.to(device)`	Atribuição: define/atualiza variável ou objeto.
85	`outputs = model(images)`	Atribuição: define/atualiza variável ou objeto.
86	`_, predicted = torch.max(outputs, 1)`	Atribuição: define/atualiza variável ou objeto.
87		Linha em branco; separa blocos lógicos.
88	`total += labels.size(0)`	Atribuição: define/atualiza variável ou objeto.
89	`correct += (predicted == labels).sum().item()`	Atribuição: define/atualiza variável ou objeto.
90		Linha em branco; separa blocos lógicos.
91	`# Armazena previsões e rótulos verdadeiros`	Comentário; documenta a intenção do bloco (não executa).
92	`probs = torch.softmax(outputs, dim=1).cpu().numpy()`	Atribuição: define/atualiza variável ou objeto.
93	`all_probs.extend(probs)`	Instrução do script no contexto do bloco.
94	`all_preds.extend(predicted.cpu().numpy())`	Instrução do script no contexto do bloco.
95	`all_labels.extend(labels.cpu().numpy())`	Instrução do script no contexto do bloco.
96		Linha em branco; separa blocos lógicos.
97	`# Acurácia final`	Comentário; documenta a intenção do bloco (não executa).
98	`acc = 100 * correct / total`	Atribuição: define/atualiza variável ou objeto.
99	`test_accuracies.append(acc)`	Instrução do script no contexto do bloco.
100	`print(f"■ Acurácia no teste após época {epoch+1}: {acc:.2f}%")`	Imprime informação de acompanhamento no console.
101		Linha em branco; separa blocos lógicos.
102	`# === Relatório e Matriz de Confusão ===`	Comentário; documenta a intenção do bloco (não executa).
103	`print(f"\n■ Acurácia final: {accuracy_score(all_labels, all_preds)*100:.2f}%")`	Imprime informação de acompanhamento no console.
104	`print(f"■ F1-score macro: {f1_score(all_labels, all_preds, average='macro'):.2f}")`	Imprime informação de acompanhamento no console.

4.4. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
105	`print(f"■ F1-score weighted: {f1_score(all_labels, all_preds, average='weighted'):.2f}")`	Imprime informação de acompanhamento no console.
106		Linha em branco; separa blocos lógicos.
107	`print("\n=== Classification Report ===")`	Imprime informação de acompanhamento no console.
108	`print(classification_report(all_labels, all_preds, target_names=class_names))`	Imprime informação de acompanhamento no console.
109		Linha em branco; separa blocos lógicos.
110	`# One-hot encoding dos rótulos verdadeiros`	Comentário; documenta a intenção do bloco (não executa).
111	`y_true = label_binarize(all_labels, classes=list(range(num_classes)))`	Atribuição: define/atualiza variável ou objeto.
112	`y_score = np.array(all_probs)`	Atribuição: define/atualiza variável ou objeto.
113		Linha em branco; separa blocos lógicos.
114	`# Plotar curva ROC para cada classe`	Comentário; documenta a intenção do bloco (não executa).
115	`plt.figure(figsize=(10, 8))`	Comando de plotagem/ajuste de figura (Matplotlib).
116	`for i in range(num_classes):`	Inicia um laço de iteração.
117	`fpr, tpr, _ = roc_curve(y_true[:, i], y_score[:, i])`	Calcula ROC e AUC (avaliação de separabilidade).
118	`roc_auc = auc(fpr, tpr)`	Calcula ROC e AUC (avaliação de separabilidade).
119	`plt.plot(fpr, tpr, lw=2, label=f'{class_names[i]} (AUC = {roc_auc:.2f})')`	Comando de plotagem/ajuste de figura (Matplotlib).
120		Linha em branco; separa blocos lógicos.
121	`plt.plot([0, 1], [0, 1], 'k--', lw=2)`	Comando de plotagem/ajuste de figura (Matplotlib).
122	`plt.xlabel('Taxa de Falsos Positivos (FPR)')`	Comando de plotagem/ajuste de figura (Matplotlib).
123	`plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')`	Comando de plotagem/ajuste de figura (Matplotlib).
124	`plt.title('Curvas ROC por Classe')`	Comando de plotagem/ajuste de figura (Matplotlib).
125	`plt.legend(loc="lower right")`	Comando de plotagem/ajuste de figura (Matplotlib).
126	`plt.grid(True)`	Comando de plotagem/ajuste de figura (Matplotlib).
127	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
128	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the
future, please use 'weights' instead.
warnings.warn(
```

```
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since
0.13 and may be removed in the future. The current behavior is equivalent to passing
`weights=ResNet18_Weights.IMAGENET1K_V1`. You can also use `weights=ResNet18_Weights.DEFAULT`
to get the most up-to-date weights.
warnings.warn(msg)
```

Saída (texto)

```
Época 1, Loss: 0.8670396407445272
Época 2, Loss: 0.15911893546581268
Época 3, Loss: 0.05169183034449816
Época 4, Loss: 0.03894476521139344
Época 5, Loss: 0.06308733653277158
Época 6, Loss: 0.04869929300621152
Época 7, Loss: 0.031198086868971586
Época 8, Loss: 0.028878680476918817
Época 9, Loss: 0.019262246050251026
Época 10, Loss: 0.027844933878319957
■ Acurácia no teste após época 10: 94.34%

■ Acurácia final: 94.34%
■ F1-score macro: 0.94
■ F1-score weighted: 0.94

=== Classification Report ===
precision recall f1-score support

C. bocourti1 0.97 1.00 0.98 58
C. danae1 0.88 0.95 0.92 63
C. exasperatus10 0.95 0.87 0.91 23
C. larvatus01 1.00 1.00 1.00 34
C. ornatus1 0.94 0.87 0.91 55
C. sapidus0 0.97 0.94 0.95 32

accuracy 0.94 265
macro avg 0.95 0.94 0.94 265
weighted avg 0.94 0.94 0.94 265
```

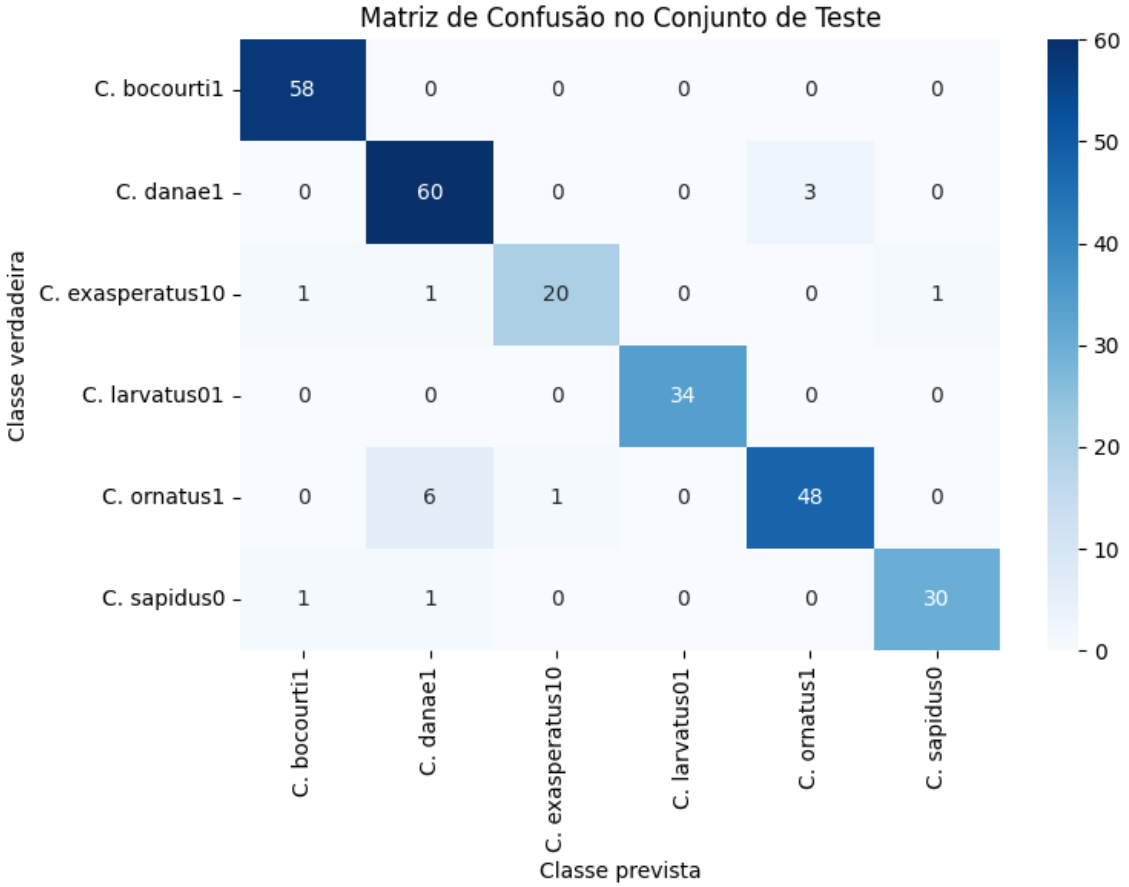
4.5. Matriz de confusão

Código e explicação linha a linha

Linha	Código	Explicação
1	`cm = confusion_matrix(all_labels, all_preds)`	Calcula matriz de confusão.
2	`plt.figure(figsize=(8, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
3	`sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)`	Atribuição: define/atualiza variável ou objeto.
4	`plt.xlabel("Classe prevista")`	Comando de plotagem/ajuste de figura (Matplotlib).
5	`plt.ylabel("Classe verdadeira")`	Comando de plotagem/ajuste de figura (Matplotlib).
6	`plt.title("Matriz de Confusão no Conjunto de Teste")`	Comando de plotagem/ajuste de figura (Matplotlib).
7	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
8	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Figura 1. Matriz de confusão



Diagonal = acertos; fora da diagonal = confusões.

4.6. Célula 6

Código e explicação linha a linha

Linha	Código	Explicação
1	`torch.save(model.state_dict(), './models_gray/modelo_final_res_net.pth')`	Instrução do script no contexto do bloco.

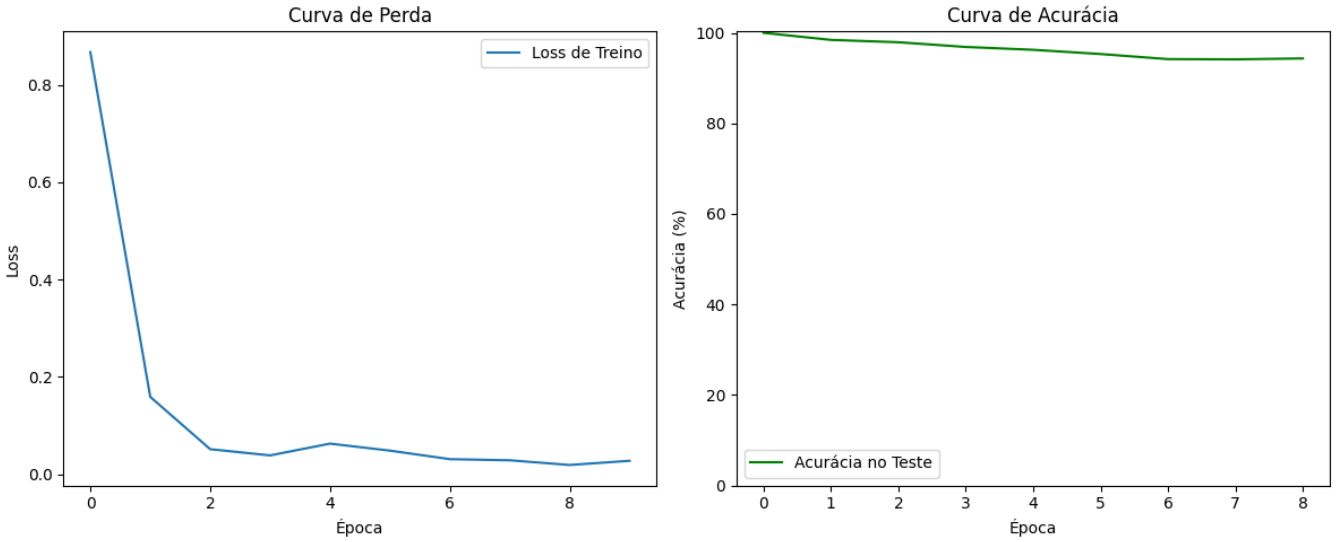
4.7. Plotar curva de perda

Código e explicação linha a linha

Linha	Código	Explicação
1	`# Plotar curva de perda`	Comentário; documenta a intenção do bloco (não executa).
2	`plt.figure(figsize=(12, 5))`	Comando de plotagem/ajuste de figura (Matplotlib).
3	`# plt.title("Evolução da Perda e Acurácia Durante o Treinamento")`	Comentário; documenta a intenção do bloco (não executa).
4		Linha em branco; separa blocos lógicos.
5	`plt.subplot(1, 2, 1)`	Comando de plotagem/ajuste de figura (Matplotlib).
6	`plt.plot(train_losses, label='Loss de Treino')`	Comando de plotagem/ajuste de figura (Matplotlib).
7	`plt.xlabel('Época')`	Comando de plotagem/ajuste de figura (Matplotlib).
8	`plt.ylabel('Loss')`	Comando de plotagem/ajuste de figura (Matplotlib).
9	`plt.title('Curva de Perda')`	Comando de plotagem/ajuste de figura (Matplotlib).
10	`plt.legend()`	Comando de plotagem/ajuste de figura (Matplotlib).
11		Linha em branco; separa blocos lógicos.
12	`# Plotar curva de acurácia`	Comentário; documenta a intenção do bloco (não executa).
13	`plt.subplot(1, 2, 2)`	Comando de plotagem/ajuste de figura (Matplotlib).
14	`plt.plot(test_accuracies, label='Acurácia no Teste', color='green')`	Comando de plotagem/ajuste de figura (Matplotlib).
15	`plt.xlabel('Época')`	Comando de plotagem/ajuste de figura (Matplotlib).
16	`plt.ylabel('Acurácia (%)')`	Comando de plotagem/ajuste de figura (Matplotlib).
17	`plt.title('Curva de Acurácia')`	Comando de plotagem/ajuste de figura (Matplotlib).
18	`plt.legend()`	Comando de plotagem/ajuste de figura (Matplotlib).
19	`plt.ylim(bottom=0)`	Comando de plotagem/ajuste de figura (Matplotlib).
20		Linha em branco; separa blocos lógicos.
21	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
22	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Figura 2. Curvas de aprendizagem



Diagnóstico de convergência/overfitting.

4.8. 1. CONFIGURAÇÕES

Código e explicação linha a linha

Linha	Código	Explicação
1		Linha em branco; separa blocos lógicos.
2	`train_losses = []`	Atribuição: define/atualiza variável ou objeto.
3	`test_accuracies = []`	Atribuição: define/atualiza variável ou objeto.
4		Linha em branco; separa blocos lógicos.
5	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
6	`# 1. CONFIGURAÇÕES`	Comentário; documenta a intenção do bloco (não executa).
7	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
8	`device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
9	`batch_size = 32`	Atribuição: define/atualiza variável ou objeto.
10	`num_epochs = 10`	Atribuição: define/atualiza variável ou objeto.
11	`data_dir = "siris_dataset_split" # Caminho para dataset com train/ e test/`	Atribuição: define/atualiza variável ou objeto.
12		Linha em branco; separa blocos lógicos.
13	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
14	`# 2. TRANSFORMAÇÕES`	Comentário; documenta a intenção do bloco (não executa).
15	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
16	`transform = transforms.Compose([`	Cria pipeline de pré-processamento (transforms).
17	`transforms.Resize((224, 224)), # Todas as redes esperam 224x224`	Instrução do script no contexto do bloco.
18	`transforms.ToTensor(),`	Instrução do script no contexto do bloco.
19	`transforms.Normalize(mean=[0.485, 0.456, 0.406], # Padrão ImageNet`	Atribuição: define/atualiza variável ou objeto.
20	`std=[0.229, 0.224, 0.225])`	Atribuição: define/atualiza variável ou objeto.
21	`])`	Instrução do script no contexto do bloco.
22		Linha em branco; separa blocos lógicos.
23	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
24	`# 3. DATALOADERS`	Comentário; documenta a intenção do bloco (não executa).
25	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
26	`generator = torch.Generator().manual_seed(SEED)`	Atribuição: define/atualiza variável ou objeto.

4.8. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
27	`train_dataset = datasets.ImageFolder(os.path.join(data_dir, "train"), transform=transform)`	Carrega dataset no padrão ImageFolder (pastas = classes).
28	`test_dataset = datasets.ImageFolder(os.path.join(data_dir, "test"), transform=transform)`	Carrega dataset no padrão ImageFolder (pastas = classes).
29		Linha em branco; separa blocos lógicos.
30	`train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True, generator=generator)`	Cria DataLoader (batches, shuffle, paralelismo).
31	`test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False, generator=generator)`	Cria DataLoader (batches, shuffle, paralelismo).
32		Linha em branco; separa blocos lógicos.
33	`num_classes = len(train_dataset.classes)`	Atribuição: define/atualiza variável ou objeto.
34	`class_names = train_dataset.classes`	Atribuição: define/atualiza variável ou objeto.
35		Linha em branco; separa blocos lógicos.
36	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
37	`# 4. MODELO`	Comentário; documenta a intenção do bloco (não executa).
38	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
39		Linha em branco; separa blocos lógicos.
40	`model = models.alexnet(pretrained=True)`	Carrega uma arquitetura CNN (em geral pré-treinada) para transfer learning.
41	`model.classifier[6] = nn.Linear(model.classifier[6].in_features, num_classes)`	Atribuição: define/atualiza variável ou objeto.
42		Linha em branco; separa blocos lógicos.
43	`model = model.to(device)`	Atribuição: define/atualiza variável ou objeto.
44		Linha em branco; separa blocos lógicos.
45	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
46	`# 5. LOSS E OTIMIZADOR`	Comentário; documenta a intenção do bloco (não executa).
47	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
48	`criterion = nn.CrossEntropyLoss()`	Define a perda CrossEntropy (classificação multi-classe).
49	`optimizer = optim.Adam(model.parameters(), lr=0.0001)`	Define o otimizador Adam (com taxa de aprendizado).
50		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
51	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
52	`# 6. TREINAMENTO`	Comentário; documenta a intenção do bloco (não executa).

4.8. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
53	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
54	`for epoch in range(num_epochs):`	Inicia um laço de iteração.
55	`model.train()`	Ativa modo de treinamento.
56	`running_loss = 0.0`	Atribuição: define/atualiza variável ou objeto.
57		Linha em branco; separa blocos lógicos.
58	`for images, labels in train_loader:`	Inicia um laço de iteração.
59	`images, labels = images.to(device), labels.to(device)`	Atribuição: define/atualiza variável ou objeto.
60		Linha em branco; separa blocos lógicos.
61	`optimizer.zero_grad()`	Instrução do script no contexto do bloco.
62	`outputs = model(images)`	Atribuição: define/atualiza variável ou objeto.
63	`loss = criterion(outputs, labels)`	Atribuição: define/atualiza variável ou objeto.
64	`loss.backward()`	Backpropagation (cálculo de gradientes).
65	`optimizer.step()`	Atualiza pesos com base nos gradientes.
66		Linha em branco; separa blocos lógicos.
67	`running_loss += loss.item()`	Atribuição: define/atualiza variável ou objeto.
68		Linha em branco; separa blocos lógicos.
69	`print(f"Época {epoch+1}, Loss: {running_loss / len(train_loader)}")`	Imprime informação de acompanhamento no console.
70		Linha em branco; separa blocos lógicos.
71	`train_losses.append(running_loss / len(train_loader))`	Instrução do script no contexto do bloco.
72		Linha em branco; separa blocos lógicos.
73	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
74	`# 7. AVALIAÇÃO NO TESTE`	Comentário; documenta a intenção do bloco (não executa).
75	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
76	`model.eval()`	Ativa modo de avaliação/inferência.
77	`correct = 0`	Atribuição: define/atualiza variável ou objeto.
78	`total = 0`	Atribuição: define/atualiza variável ou objeto.

4.8. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
79		Linha em branco; separa blocos lógicos.
80	`all_preds = []`	Atribuição: define/atualiza variável ou objeto.
81	`all_labels = []`	Atribuição: define/atualiza variável ou objeto.
82	`all_probs = []`	Atribuição: define/atualiza variável ou objeto.
83	`with torch.no_grad():`	Desativa gradientes para inferência mais rápida.
84	`for images, labels in test_loader:`	Inicia um laço de iteração.
85	`images, labels = images.to(device), labels.to(device)`	Atribuição: define/atualiza variável ou objeto.
86	`outputs = model(images)`	Atribuição: define/atualiza variável ou objeto.
87	`_, predicted = torch.max(outputs, 1)`	Atribuição: define/atualiza variável ou objeto.
88		Linha em branco; separa blocos lógicos.
89	`total += labels.size(0)`	Atribuição: define/atualiza variável ou objeto.
90	`correct += (predicted == labels).sum().item()`	Atribuição: define/atualiza variável ou objeto.
91		Linha em branco; separa blocos lógicos.
92	`# Armazena previsões e rótulos verdadeiros`	Comentário; documenta a intenção do bloco (não executa).
93	`probs = torch.softmax(outputs, dim=1).cpu().numpy()`	Atribuição: define/atualiza variável ou objeto.
94	`all_probs.extend(probs)`	Instrução do script no contexto do bloco.
95	`all_preds.extend(predicted.cpu().numpy())`	Instrução do script no contexto do bloco.
96	`all_labels.extend(labels.cpu().numpy())`	Instrução do script no contexto do bloco.
97		Linha em branco; separa blocos lógicos.
98	`# Acurácia final`	Comentário; documenta a intenção do bloco (não executa).
99	`acc = 100 * correct / total`	Atribuição: define/atualiza variável ou objeto.
100	`test_accuracies.append(acc)`	Instrução do script no contexto do bloco.
101	`print(f"■ Acurácia no teste após época {epoch+1}: {acc:.2f}%")`	Imprime informação de acompanhamento no console.
102		Linha em branco; separa blocos lógicos.
103	`# === Relatório e Matriz de Confusão ===`	Comentário; documenta a intenção do bloco (não executa).
104		Linha em branco; separa blocos lógicos.

4.8. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
105	`print(f"\n■ Acurácia final: {accuracy_score(all_labels, all_preds)*100:.2f}%")`	Imprime informação de acompanhamento no console.
106	`print(f"■ F1-score macro: {f1_score(all_labels, all_preds, average='macro'):.2f}")`	Imprime informação de acompanhamento no console.
107	`print(f"■ F1-score weighted: {f1_score(all_labels, all_preds, average='weighted'):.2f}")`	Imprime informação de acompanhamento no console.
108		Linha em branco; separa blocos lógicos.
109	`print("\n=== Classification Report ===")`	Imprime informação de acompanhamento no console.
110	`print(classification_report(all_labels, all_preds, target_names=class_names))`	Imprime informação de acompanhamento no console.
111		Linha em branco; separa blocos lógicos.
112	`cm = confusion_matrix(all_labels, all_preds)`	Calcula matriz de confusão.
113	`plt.figure(figsize=(8, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
114	`sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)`	Atribuição: define/atualiza variável ou objeto.
115	`plt.xlabel("Classe prevista")`	Comando de plotagem/ajuste de figura (Matplotlib).
116	`plt.ylabel("Classe verdadeira")`	Comando de plotagem/ajuste de figura (Matplotlib).
117	`plt.title("Matriz de Confusão no Conjunto de Teste")`	Comando de plotagem/ajuste de figura (Matplotlib).
118	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
119	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).
120		Linha em branco; separa blocos lógicos.
121		Linha em branco; separa blocos lógicos.
122	`# One-hot encoding dos rótulos verdadeiros`	Comentário; documenta a intenção do bloco (não executa).
123	`y_true = label_binarize(all_labels, classes=list(range(num_classes)))`	Atribuição: define/atualiza variável ou objeto.
124	`y_score = np.array(all_probs)`	Atribuição: define/atualiza variável ou objeto.
125		Linha em branco; separa blocos lógicos.
126	`# Plotar curva ROC para cada classe`	Comentário; documenta a intenção do bloco (não executa).
127	`plt.figure(figsize=(10, 8))`	Comando de plotagem/ajuste de figura (Matplotlib).
128	`for i in range(num_classes):`	Inicia um laço de iteração.
129	`fpr, tpr, _ = roc_curve(y_true[:, i], y_score[:, i])`	Calcula ROC e AUC (avaliação de separabilidade).
130	`roc_auc = auc(fpr, tpr)`	Calcula ROC e AUC (avaliação de separabilidade).

4.8. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
131	`plt.plot(fpr, tpr, lw=2, label=f'{class_names[i]} (AUC = {roc_auc:.2f})')`	Comando de plotagem/ajuste de figura (Matplotlib).
132		Linha em branco; separa blocos lógicos.
133	`plt.plot([0, 1], [0, 1], 'k--', lw=2)`	Comando de plotagem/ajuste de figura (Matplotlib).
134	`plt.xlabel('Taxa de Falsos Positivos (FPR)')`	Comando de plotagem/ajuste de figura (Matplotlib).
135	`plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')`	Comando de plotagem/ajuste de figura (Matplotlib).
136	`plt.title('Curvas ROC por Classe')`	Comando de plotagem/ajuste de figura (Matplotlib).
137	`plt.legend(loc="lower right")`	Comando de plotagem/ajuste de figura (Matplotlib).
138	`plt.grid(True)`	Comando de plotagem/ajuste de figura (Matplotlib).
139	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
140	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

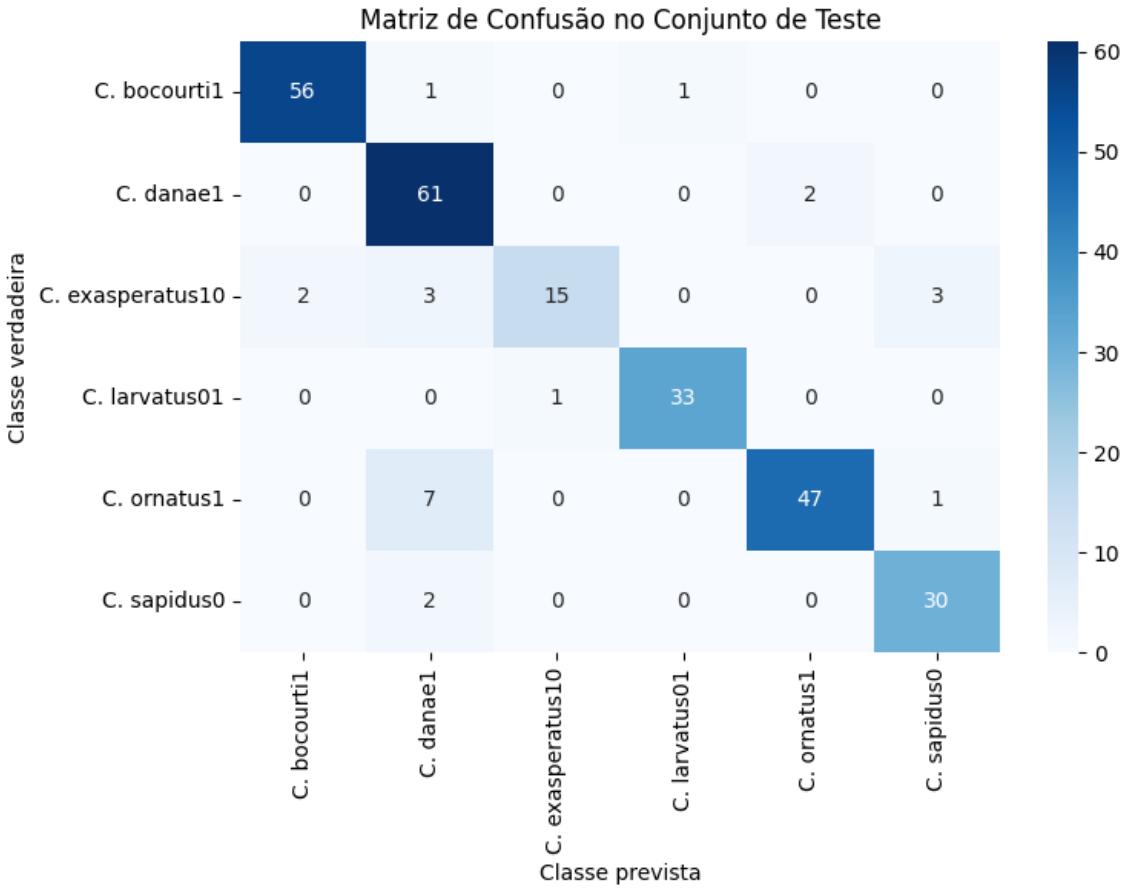
```
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the
future, please use 'weights' instead.
warnings.warn(
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since
0.13 and may be removed in the future. The current behavior is equivalent to passing
`weights=AlexNet_Weights.IMAGENET1K_V1`. You can also use `weights=AlexNet_Weights.DEFAULT` to
get the most up-to-date weights.
warnings.warn(msg)
```

Saída (texto)

```
Época 1, Loss: 1.0881989101568859
Época 2, Loss: 0.5112732554475466
Época 3, Loss: 0.24913768110175927
Época 4, Loss: 0.156528609773765
Época 5, Loss: 0.09725984937200943
Época 6, Loss: 0.08504721991096933
Época 7, Loss: 0.03392493337451015
Época 8, Loss: 0.030723032987831782
Época 9, Loss: 0.029184483063484853
Época 10, Loss: 0.04248757956083864
■ Acurácia no teste após época 10: 91.32%

■ Acurácia final: 91.32%
■ F1-score macro: 0.90
■ F1-score weighted: 0.91

=== Classification Report ===
precision recall f1-score support

C. bocourti1 0.97 0.97 0.97 58
C. danae1 0.82 0.97 0.89 63
C. exasperatus10 0.94 0.65 0.77 23
C. larvatus01 0.97 0.97 0.97 34
```

```
C. ornatus1 0.96 0.85 0.90 55
C. sapidus0 0.88 0.94 0.91 32

accuracy 0.91 265
macro avg 0.92 0.89 0.90 265
weighted avg 0.92 0.91 0.91 265
```

Figura 3. Matriz de confusão



Diagonal = acertos; fora da diagonal = confusões.

4.9. Célula 9

Código e explicação linha a linha

Linha	Código	Explicação
1	`torch.save(model.state_dict(), './models_gray/modelo_final_alex_net.pth')`	Instrução do script no contexto do bloco.

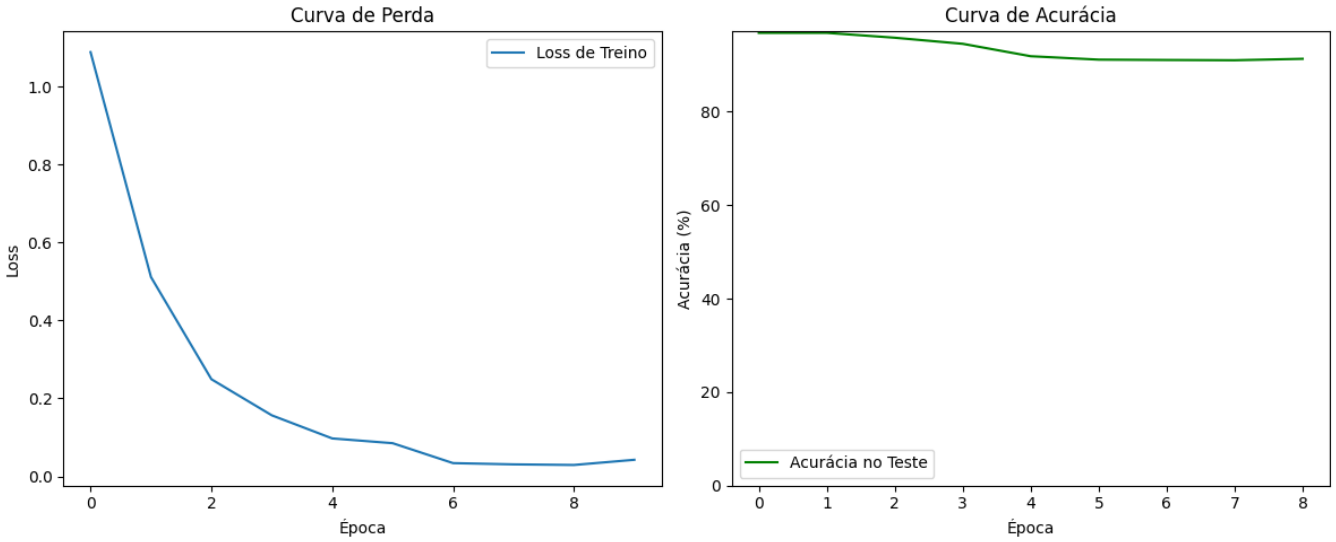
4.10. Plotar curva de perda

Código e explicação linha a linha

Linha	Código	Explicação
1	`# Plotar curva de perda`	Comentário; documenta a intenção do bloco (não executa).
2	`plt.figure(figsize=(12, 5))`	Comando de plotagem/ajuste de figura (Matplotlib).
3	`# plt.title("Evolução da Perda e Acurácia Durante o Treinamento")`	Comentário; documenta a intenção do bloco (não executa).
4		Linha em branco; separa blocos lógicos.
5	`plt.subplot(1, 2, 1)`	Comando de plotagem/ajuste de figura (Matplotlib).
6	`plt.plot(train_losses, label='Loss de Treino')`	Comando de plotagem/ajuste de figura (Matplotlib).
7	`plt.xlabel('Época')`	Comando de plotagem/ajuste de figura (Matplotlib).
8	`plt.ylabel('Loss')`	Comando de plotagem/ajuste de figura (Matplotlib).
9	`plt.title('Curva de Perda')`	Comando de plotagem/ajuste de figura (Matplotlib).
10	`plt.legend()`	Comando de plotagem/ajuste de figura (Matplotlib).
11		Linha em branco; separa blocos lógicos.
12	`# Plotar curva de acurácia`	Comentário; documenta a intenção do bloco (não executa).
13	`plt.subplot(1, 2, 2)`	Comando de plotagem/ajuste de figura (Matplotlib).
14	`plt.plot(test_accuracies, label='Acurácia no Teste', color='green')`	Comando de plotagem/ajuste de figura (Matplotlib).
15	`plt.xlabel('Época')`	Comando de plotagem/ajuste de figura (Matplotlib).
16	`plt.ylabel('Acurácia (%)')`	Comando de plotagem/ajuste de figura (Matplotlib).
17	`plt.title('Curva de Acurácia')`	Comando de plotagem/ajuste de figura (Matplotlib).
18	`plt.legend()`	Comando de plotagem/ajuste de figura (Matplotlib).
19	`plt.ylim(bottom=0)`	Comando de plotagem/ajuste de figura (Matplotlib).
20		Linha em branco; separa blocos lógicos.
21	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
22	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Figura 4. Curvas de aprendizagem



Diagnóstico de convergência/overfitting.

4.11. 1. CONFIGURAÇÕES

Código e explicação linha a linha

Linha	Código	Explicação
1		Linha em branco; separa blocos lógicos.
2	`train_losses = []`	Atribuição: define/atualiza variável ou objeto.
3	`test_accuracies = []`	Atribuição: define/atualiza variável ou objeto.
4		Linha em branco; separa blocos lógicos.
5	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
6	`# 1. CONFIGURAÇÕES`	Comentário; documenta a intenção do bloco (não executa).
7	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
8	`device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
9	`batch_size = 32`	Atribuição: define/atualiza variável ou objeto.
10	`num_epochs = 10`	Atribuição: define/atualiza variável ou objeto.
11	`data_dir = "siris_dataset_split" # Caminho para dataset com train/ e test/`	Atribuição: define/atualiza variável ou objeto.
12		Linha em branco; separa blocos lógicos.
13	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
14	`# 2. TRANSFORMAÇÕES`	Comentário; documenta a intenção do bloco (não executa).
15	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
16	`transform = transforms.Compose([`	Cria pipeline de pré-processamento (transforms).
17	`transforms.Resize((224, 224)), # Todas as redes esperam 224x224`	Instrução do script no contexto do bloco.
18	`transforms.ToTensor(),`	Instrução do script no contexto do bloco.
19	`transforms.Normalize(mean=[0.485, 0.456, 0.406], # Padrão ImageNet`	Atribuição: define/atualiza variável ou objeto.
20	`std=[0.229, 0.224, 0.225])`	Atribuição: define/atualiza variável ou objeto.
21	`])`	Instrução do script no contexto do bloco.
22		Linha em branco; separa blocos lógicos.
23	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
24	`# 3. DATALOADERS`	Comentário; documenta a intenção do bloco (não executa).
25	`# ======================`	Comentário; documenta a intenção do bloco (não executa).

4.11. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
26	`train_dataset = datasets.ImageFolder(os.path.join(data_dir, "train"), transform=transform)`	Carrega dataset no padrão ImageFolder (pastas = classes).
27	`test_dataset = datasets.ImageFolder(os.path.join(data_dir, "test"), transform=transform)`	Carrega dataset no padrão ImageFolder (pastas = classes).
28		Linha em branco; separa blocos lógicos.
29	`train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)`	Cria DataLoader (batches, shuffle, paralelismo).
30	`test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)`	Cria DataLoader (batches, shuffle, paralelismo).
31		Linha em branco; separa blocos lógicos.
32	`num_classes = len(train_dataset.classes)`	Atribuição: define/atualiza variável ou objeto.
33		Linha em branco; separa blocos lógicos.
34	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
35	`# 4. MODELO`	Comentário; documenta a intenção do bloco (não executa).
36	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
37		Linha em branco; separa blocos lógicos.
38	`model = models.mobilenet_v2(pretrained=True)`	Carrega uma arquitetura CNN (em geral pré-treinada) para transfer learning.
39	`model.classifier[1] = nn.Linear(model.classifier[1].in_features, num_classes)`	Atribuição: define/atualiza variável ou objeto.
40		Linha em branco; separa blocos lógicos.
41	`model = model.to(device)`	Atribuição: define/atualiza variável ou objeto.
42		Linha em branco; separa blocos lógicos.
43	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
44	`# 5. LOSS E OTIMIZADOR`	Comentário; documenta a intenção do bloco (não executa).
45	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
46	`criterion = nn.CrossEntropyLoss()`	Define a perda CrossEntropy (classificação multi-classe).
47	`optimizer = optim.Adam(model.parameters(), lr=0.0001)`	Define o otimizador Adam (com taxa de aprendizado).
48		Linha em branco; separa blocos lógicos.
49	`# ======================`	Comentário; documenta a intenção do bloco (não executa).

Linha	Código	Explicação
50	`# 6. TREINAMENTO`	Comentário; documenta a intenção do bloco (não executa).

4.11. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
51	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
52	`for epoch in range(num_epochs):`	Inicia um laço de iteração.
53	`model.train()`	Ativa modo de treinamento.
54	`running_loss = 0.0`	Atribuição: define/atualiza variável ou objeto.
55		Linha em branco; separa blocos lógicos.
56	`for images, labels in train_loader:`	Inicia um laço de iteração.
57	`images, labels = images.to(device), labels.to(device)`	Atribuição: define/atualiza variável ou objeto.
58		Linha em branco; separa blocos lógicos.
59	`optimizer.zero_grad()`	Instrução do script no contexto do bloco.
60	`outputs = model(images)`	Atribuição: define/atualiza variável ou objeto.
61	`loss = criterion(outputs, labels)`	Atribuição: define/atualiza variável ou objeto.
62	`loss.backward()`	Backpropagation (cálculo de gradientes).
63	`optimizer.step()`	Atualiza pesos com base nos gradientes.
64		Linha em branco; separa blocos lógicos.
65	`running_loss += loss.item()`	Atribuição: define/atualiza variável ou objeto.
66		Linha em branco; separa blocos lógicos.
67	`print(f"Época {epoch+1}, Loss: {running_loss / len(train_loader)}")`	Imprime informação de acompanhamento no console.
68		Linha em branco; separa blocos lógicos.
69	`train_losses.append(running_loss / len(train_loader))`	Instrução do script no contexto do bloco.
70		Linha em branco; separa blocos lógicos.
71	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
72	`# 7. AVALIAÇÃO NO TESTE`	Comentário; documenta a intenção do bloco (não executa).
73	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
74	`model.eval()`	Ativa modo de avaliação/inferência.
75	`correct = 0`	Atribuição: define/atualiza variável ou objeto.

4.11. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
76	`total = 0`	Atribuição: define/atualiza variável ou objeto.
77		Linha em branco; separa blocos lógicos.
78	`all_preds = []`	Atribuição: define/atualiza variável ou objeto.
79	`all_labels = []`	Atribuição: define/atualiza variável ou objeto.
80	`all_probs = []`	Atribuição: define/atualiza variável ou objeto.
81	`with torch.no_grad():`	Desativa gradientes para inferência mais rápida.
82	`for images, labels in test_loader:`	Inicia um laço de iteração.
83	`images, labels = images.to(device), labels.to(device)`	Atribuição: define/atualiza variável ou objeto.
84	`outputs = model(images)`	Atribuição: define/atualiza variável ou objeto.
85	`_, predicted = torch.max(outputs, 1)`	Atribuição: define/atualiza variável ou objeto.
86		Linha em branco; separa blocos lógicos.
87	`total += labels.size(0)`	Atribuição: define/atualiza variável ou objeto.
88	`correct += (predicted == labels).sum().item()`	Atribuição: define/atualiza variável ou objeto.
89		Linha em branco; separa blocos lógicos.
90	`# Armazena previsões e rótulos verdadeiros`	Comentário; documenta a intenção do bloco (não executa).
91	`probs = torch.softmax(outputs, dim=1).cpu().numpy()`	Atribuição: define/atualiza variável ou objeto.
92	`all_probs.extend(probs)`	Instrução do script no contexto do bloco.
93	`all_preds.extend(predicted.cpu().numpy())`	Instrução do script no contexto do bloco.
94	`all_labels.extend(labels.cpu().numpy())`	Instrução do script no contexto do bloco.
95		Linha em branco; separa blocos lógicos.
96	`# Acurácia final`	Comentário; documenta a intenção do bloco (não executa).
97	`acc = 100 * correct / total`	Atribuição: define/atualiza variável ou objeto.
98	`test_accuracies.append(acc)`	Instrução do script no contexto do bloco.
99	`print(f"■ Acurácia no teste após época {epoch+1}: {acc:.2f}%")`	Imprime informação de acompanhamento no console.
100		Linha em branco; separa blocos lógicos.

4.11. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
101	`# === Relatório e Matriz de Confusão ===`	Comentário; documenta a intenção do bloco (não executa).
102	`class_names = train_dataset.classes`	Atribuição: define/atualiza variável ou objeto.
103		Linha em branco; separa blocos lógicos.
104	`print(f"\n■ Acurácia final: {accuracy_score(all_labels, all_preds)*100:.2f}%")`	Imprime informação de acompanhamento no console.
105	`print(f"■ F1-score macro: {f1_score(all_labels, all_preds, average='macro'):.2f}")`	Imprime informação de acompanhamento no console.
106	`print(f"■ F1-score weighted: {f1_score(all_labels, all_preds, average='weighted'):.2f}")`	Imprime informação de acompanhamento no console.
107		Linha em branco; separa blocos lógicos.
108	`print("\n=== Classification Report ===")`	Imprime informação de acompanhamento no console.
109	`print(classification_report(all_labels, all_preds, target_names=class_names))`	Imprime informação de acompanhamento no console.
110		Linha em branco; separa blocos lógicos.
111	`cm = confusion_matrix(all_labels, all_preds)`	Calcula matriz de confusão.
112	`plt.figure(figsize=(8, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
113	`sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)`	Atribuição: define/atualiza variável ou objeto.
114	`plt.xlabel("Classe prevista")`	Comando de plotagem/ajuste de figura (Matplotlib).
115	`plt.ylabel("Classe verdadeira")`	Comando de plotagem/ajuste de figura (Matplotlib).
116	`plt.title("Matriz de Confusão no Conjunto de Teste")`	Comando de plotagem/ajuste de figura (Matplotlib).
117	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
118	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).
119		Linha em branco; separa blocos lógicos.
120	`# One-hot encoding dos rótulos verdadeiros`	Comentário; documenta a intenção do bloco (não executa).
121	`y_true = label_binarize(all_labels, classes=list(range(num_classes)))`	Atribuição: define/atualiza variável ou objeto.
122	`y_score = np.array(all_probs)`	Atribuição: define/atualiza variável ou objeto.
123		Linha em branco; separa blocos lógicos.
124	`# Plotar curva ROC para cada classe`	Comentário; documenta a intenção do bloco (não executa).
125	`plt.figure(figsize=(10, 8))`	Comando de plotagem/ajuste de figura (Matplotlib).

4.11. 1. CONFIGURAÇÕES (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
126	`for i in range(num_classes):`	Inicia um laço de iteração.
127	`fpr, tpr, _ = roc_curve(y_true[:, i], y_score[:, i])`	Calcula ROC e AUC (avaliação de separabilidade).
128	`roc_auc = auc(fpr, tpr)`	Calcula ROC e AUC (avaliação de separabilidade).
129	`plt.plot(fpr, tpr, lw=2, label=f'{class_names[i]} (AUC = {roc_auc:.2f})')`	Comando de plotagem/ajuste de figura (Matplotlib).
130		Linha em branco; separa blocos lógicos.
131	`plt.plot([0, 1], [0, 1], 'k--', lw=2)`	Comando de plotagem/ajuste de figura (Matplotlib).
132	`plt.xlabel('Taxa de Falsos Positivos (FPR)')`	Comando de plotagem/ajuste de figura (Matplotlib).
133	`plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')`	Comando de plotagem/ajuste de figura (Matplotlib).
134	`plt.title('Curvas ROC por Classe')`	Comando de plotagem/ajuste de figura (Matplotlib).
135	`plt.legend(loc="lower right")`	Comando de plotagem/ajuste de figura (Matplotlib).
136	`plt.grid(True)`	Comando de plotagem/ajuste de figura (Matplotlib).
137	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
138	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

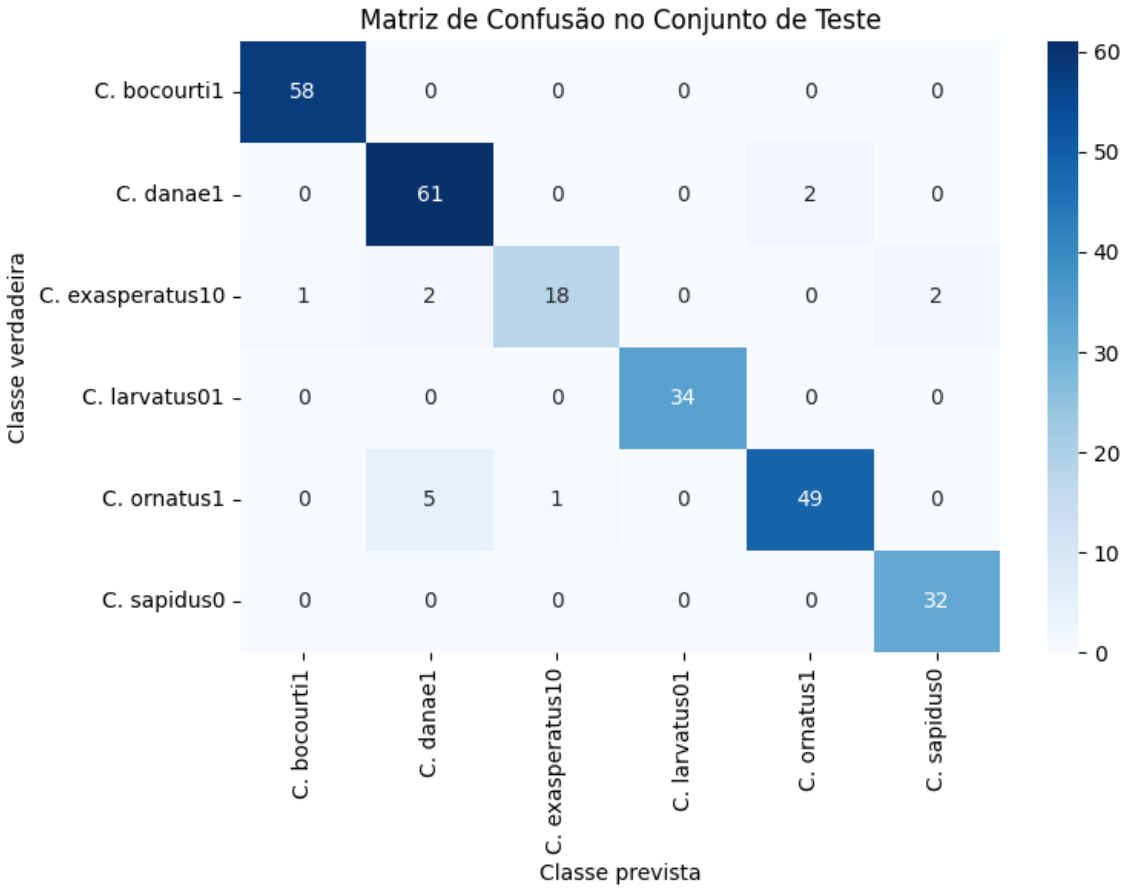
```
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the
future, please use 'weights' instead.
warnings.warn(
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since
0.13 and may be removed in the future. The current behavior is equivalent to passing
`weights=MobileNet_V2_Weights.IMAGENET1K_V1`. You can also use
`weights=MobileNet_V2_Weights.DEFAULT` to get the most up-to-date weights.
warnings.warn(msg)
```

Saída (texto)

```
Época 1, Loss: 1.0447706540425619
Época 2, Loss: 0.3419387757778168
Época 3, Loss: 0.16723648185531298
Época 4, Loss: 0.06556786006937425
Época 5, Loss: 0.06545100891962648
Época 6, Loss: 0.03728357786312699
Época 7, Loss: 0.02186536641481022
Época 8, Loss: 0.013251725723966956
Época 9, Loss: 0.020682721096090973
Época 10, Loss: 0.027087755578880508
■ Acurácia no teste após época 10: 95.09%

■ Acurácia final: 95.09%
■ F1-score macro: 0.95
■ F1-score weighted: 0.95

=== Classification Report ===
```

```
precision recall f1-score support

C. bocourti1 0.98 1.00 0.99 58
C. danae1 0.90 0.97 0.93 63
C. exasperatus10 0.95 0.78 0.86 23
C. larvatus01 1.00 1.00 1.00 34
C. ornatus1 0.96 0.89 0.92 55
C. sapidus0 0.94 1.00 0.97 32

accuracy 0.95 265
macro avg 0.95 0.94 0.95 265
weighted avg 0.95 0.95 0.95 265
```

Figura 5. Matriz de confusão



Diagonal = acertos; fora da diagonal = confusões.

4.12. Célula 12

Código e explicação linha a linha

Linha	Código	Explicação
1	`torch.save(model.state_dict(), './models_gray/modelo_final_mobile_net.pth')`	Instrução do script no contexto do bloco.

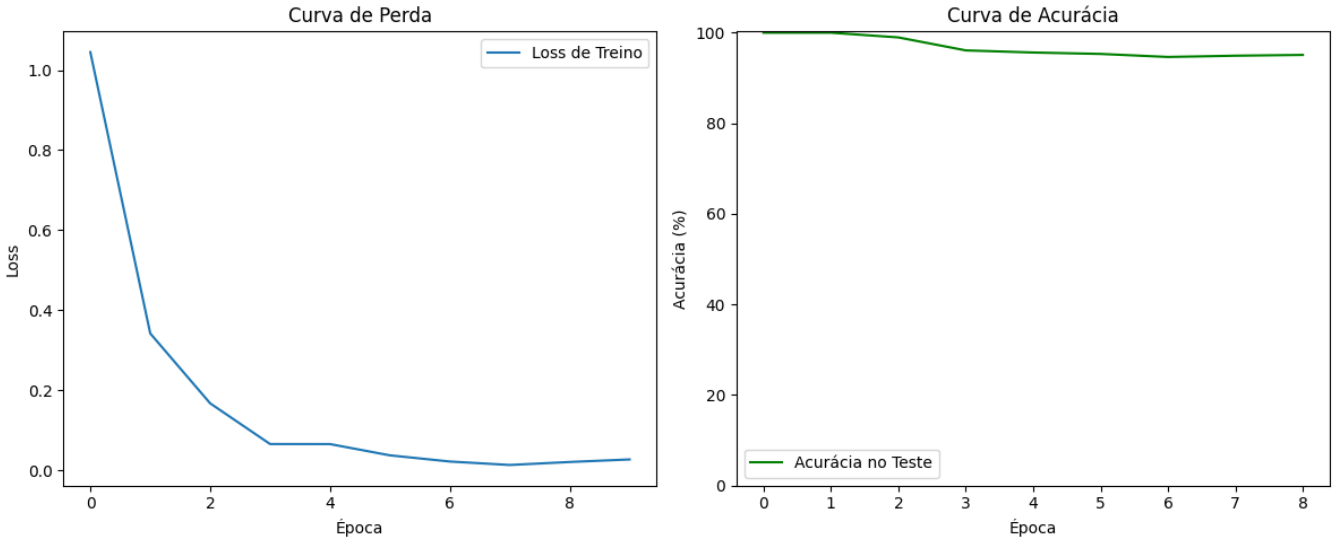
4.13. Plotar curva de perda

Código e explicação linha a linha

Linha	Código	Explicação
1	`# Plotar curva de perda`	Comentário; documenta a intenção do bloco (não executa).
2	`plt.figure(figsize=(12, 5))`	Comando de plotagem/ajuste de figura (Matplotlib).
3	`# plt.title("Evolução da Perda e Acurácia Durante o Treinamento")`	Comentário; documenta a intenção do bloco (não executa).
4	`plt.subplot(1, 2, 1)`	Comando de plotagem/ajuste de figura (Matplotlib).
5	`plt.plot(train_losses, label='Loss de Treino')`	Comando de plotagem/ajuste de figura (Matplotlib).
6	`plt.xlabel('Época')`	Comando de plotagem/ajuste de figura (Matplotlib).
7	`plt.ylabel('Loss')`	Comando de plotagem/ajuste de figura (Matplotlib).
8	`plt.title('Curva de Perda')`	Comando de plotagem/ajuste de figura (Matplotlib).
9	`plt.legend()`	Comando de plotagem/ajuste de figura (Matplotlib).
10		Linha em branco; separa blocos lógicos.
11	`# Plotar curva de acurácia`	Comentário; documenta a intenção do bloco (não executa).
12	`plt.subplot(1, 2, 2)`	Comando de plotagem/ajuste de figura (Matplotlib).
13	`plt.plot(test_accuracies, label='Acurácia no Teste', color='green')`	Comando de plotagem/ajuste de figura (Matplotlib).
14	`plt.xlabel('Época')`	Comando de plotagem/ajuste de figura (Matplotlib).
15	`plt.ylabel('Acurácia (%)')`	Comando de plotagem/ajuste de figura (Matplotlib).
16	`plt.title('Curva de Acurácia')`	Comando de plotagem/ajuste de figura (Matplotlib).
17	`plt.legend()`	Comando de plotagem/ajuste de figura (Matplotlib).
18	`plt.ylim(bottom=0)`	Comando de plotagem/ajuste de figura (Matplotlib).
19		Linha em branco; separa blocos lógicos.
20	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
21	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Figura 6. Curvas de aprendizagem



Diagnóstico de convergência/overfitting.

4.14. === CONFIGURAÇÕES ===

Código e explicação linha a linha

Linha	Código	Explicação
1	`import torch`	Importa bibliotecas necessárias para o pipeline.
2	`import os`	Importa bibliotecas necessárias para o pipeline.
3	`from PIL import Image`	Importa bibliotecas necessárias para o pipeline.
4	`from torchvision import transforms, models`	Importa bibliotecas necessárias para o pipeline.
5	`from sklearn.metrics import classification_report, confusion_matrix`	Importa bibliotecas necessárias para o pipeline.
6	`import numpy as np`	Importa bibliotecas necessárias para o pipeline.
7		Linha em branco; separa blocos lógicos.
8	`# === CONFIGURAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
9	`caminho_dataset = 'siris_dataset_split/external'`	Atribuição: define/atualiza variável ou objeto.
10	`classes = sorted(os.listdir(caminho_dataset)) # os nomes das pastas são os nomes das classes`	Atribuição: define/atualiza variável ou objeto.
11	`class_to_idx = {classe: idx for idx, classe in enumerate(classes)}`	Atribuição: define/atualiza variável ou objeto.
12	`caminho_modelo = './models_gray/modelo_final_res_net.pth' # Caminho para o modelo treinado`	Atribuição: define/atualiza variável ou objeto.
13		Linha em branco; separa blocos lógicos.
14	`# === TRANSFORMAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
15	`transform = transforms.Compose([`	Cria pipeline de pré-processamento (transforms).
16	`transforms.Resize((224, 224)),`	Instrução do script no contexto do bloco.
17	`transforms.ToTensor(),`	Instrução do script no contexto do bloco.
18	`transforms.Normalize(mean=[0.485, 0.456, 0.406],`	Atribuição: define/atualiza variável ou objeto.
19	`std=[0.229, 0.224, 0.225])`	Atribuição: define/atualiza variável ou objeto.
20	`])`	Instrução do script no contexto do bloco.
21		Linha em branco; separa blocos lógicos.
22	`# === CARREGAR MODELO ===`	Comentário; documenta a intenção do bloco (não executa).
23	`model = models.resnet18(pretrained=False)`	Carrega uma arquitetura CNN (em geral pré-treinada) para transfer learning.
24	`model.fc = torch.nn.Linear(model.fc.in_features, len(classes))`	Atribuição: define/atualiza variável ou objeto.
25	`model.load_state_dict(torch.load(caminho_modelo, map_location='cpu'))`	Atribuição: define/atualiza variável ou objeto.

4.14. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
26	`model.eval()`	Ativa modo de avaliação/inferência.
27		Linha em branco; separa blocos lógicos.
28	`# === AVALIAÇÃO ===`	Comentário; documenta a intenção do bloco (não executa).
29	`y_true = []`	Atribuição: define/atualiza variável ou objeto.
30	`y_pred = []`	Atribuição: define/atualiza variável ou objeto.
31		Linha em branco; separa blocos lógicos.
32	`for classe in classes:`	Inicia um laço de iteração.
33	`pasta_classe = os.path.join(caminho_dataset, classe)`	Atribuição: define/atualiza variável ou objeto.
34	`for img_nome in os.listdir(pasta_classe):`	Inicia um laço de iteração.
35	`if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):`	Estrutura condicional.
36	`continue`	Instrução do script no contexto do bloco.
37	`img_path = os.path.join(pasta_classe, img_nome)`	Atribuição: define/atualiza variável ou objeto.
38	`img = Image.open(img_path).convert('RGB')`	Atribuição: define/atualiza variável ou objeto.
39	`img_tensor = transform(img).unsqueeze(0)`	Atribuição: define/atualiza variável ou objeto.
40		Linha em branco; separa blocos lógicos.
41	`with torch.no_grad():`	Desativa gradientes para inferência mais rápida.
42	`output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.
43	`_, pred = torch.max(output, 1)`	Atribuição: define/atualiza variável ou objeto.
44		Linha em branco; separa blocos lógicos.
45	`y_true.append(class_to_idx[classe])`	Instrução do script no contexto do bloco.
46	`y_pred.append(pred.item())`	Instrução do script no contexto do bloco.
47		Linha em branco; separa blocos lógicos.
48	`# === RELATÓRIO ===`	Comentário; documenta a intenção do bloco (não executa).
49	`print("=== Classification Report ===")`	Imprime informação de acompanhamento no console.
50	`print(classification_report(y_true, y_pred, target_names=classes))`	Imprime informação de acompanhamento no console.

4.14. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
51		Linha em branco; separa blocos lógicos.
52	`print("=== Matriz de Confusão ===")`	Imprime informação de acompanhamento no console.
53	`cm = confusion_matrix(y_true, y_pred)`	Calcula matriz de confusão.
54	`plt.figure(figsize=(8, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
55	`sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=classes, yticklabels=classes)`	Atribuição: define/atualiza variável ou objeto.
56	`plt.xlabel("Classe prevista")`	Comando de plotagem/ajuste de figura (Matplotlib).
57	`plt.ylabel("Classe verdadeira")`	Comando de plotagem/ajuste de figura (Matplotlib).
58	`plt.title("■ Matriz de Confusão")`	Comando de plotagem/ajuste de figura (Matplotlib).
59	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
60	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

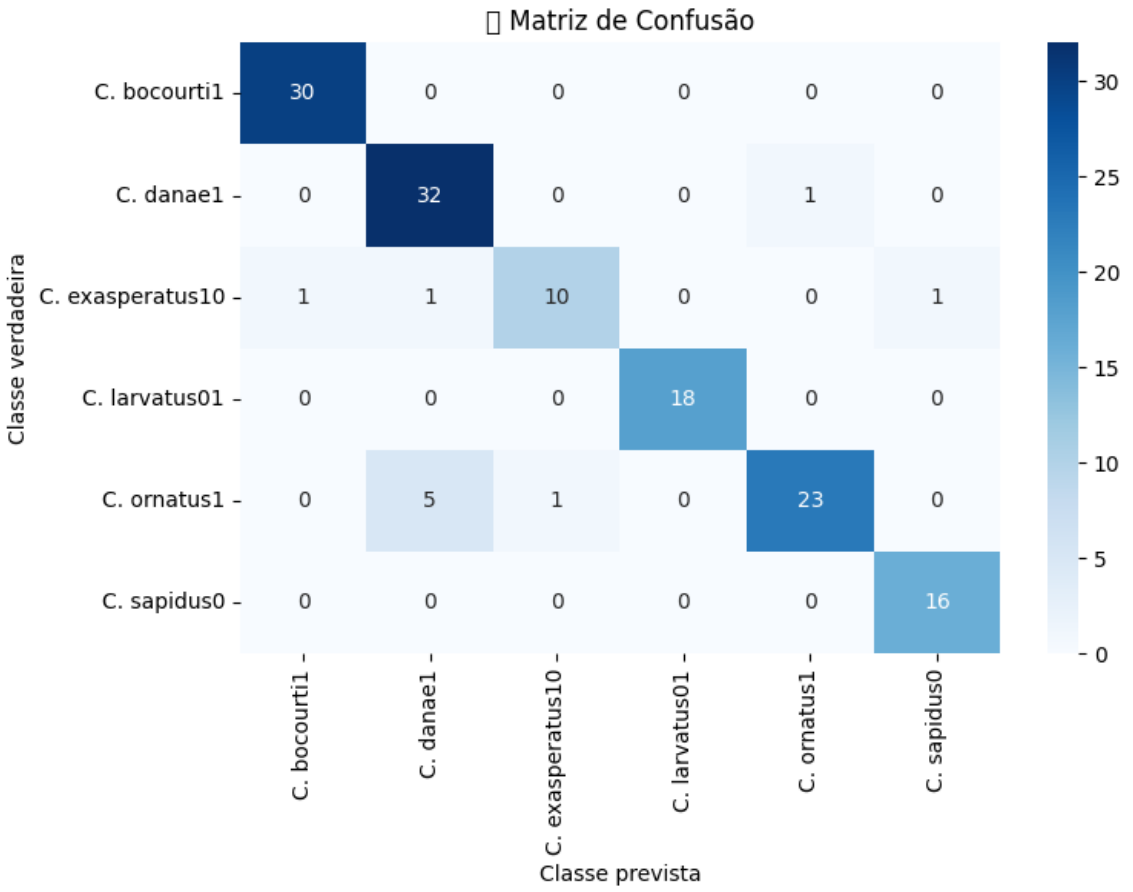
Saída (texto)

```
=== Classification Report ===
precision recall f1-score support

C. bocourti1 0.97 1.00 0.98 30
C. danae1 0.84 0.97 0.90 33
C. exasperatus10 0.91 0.77 0.83 13
C. larvatus01 1.00 1.00 1.00 18
C. ornatus1 0.96 0.79 0.87 29
C. sapidus0 0.94 1.00 0.97 16

accuracy 0.93 139
macro avg 0.94 0.92 0.93 139
weighted avg 0.93 0.93 0.93 139

=== Matriz de Confusão ===
```

Saída (texto)

```
C:\Users\vitin\AppData\Local\Temp\ipykernel_16104\1412345200.py:59: UserWarning: Glyph 128269
(\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) DejaVu Sans.
plt.tight_layout()
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\IPython\core\pylabtools.py:170:
UserWarning: Glyph 128269 (\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) DejaVu
Sans.
fig.canvas.print_figure(bytes_io, **kw)
```

Figura 7. Matriz de confusão



Diagonal = acertos; fora da diagonal = confusões.

4.15. === CONFIGURAÇÕES ===

Código e explicação linha a linha

Linha	Código	Explicação
1	`import torch`	Importa bibliotecas necessárias para o pipeline.
2	`import os`	Importa bibliotecas necessárias para o pipeline.
3	`from PIL import Image`	Importa bibliotecas necessárias para o pipeline.
4	`from torchvision import transforms, models`	Importa bibliotecas necessárias para o pipeline.
5	`from sklearn.metrics import classification_report, confusion_matrix`	Importa bibliotecas necessárias para o pipeline.
6	`import numpy as np`	Importa bibliotecas necessárias para o pipeline.
7		Linha em branco; separa blocos lógicos.
8	`# === CONFIGURAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
9	`caminho_dataset = 'siris_dataset_split/external'`	Atribuição: define/atualiza variável ou objeto.
10	`classes = sorted(os.listdir(caminho_dataset)) # os nomes das pastas são os nomes das classes`	Atribuição: define/atualiza variável ou objeto.
11	`class_to_idx = {classe: idx for idx, classe in enumerate(classes)}`	Atribuição: define/atualiza variável ou objeto.
12	`caminho_modelo = './models_gray/modelo_final_alex_net.pth' # Caminho para o modelo treinado`	Atribuição: define/atualiza variável ou objeto.
13		Linha em branco; separa blocos lógicos.
14	`# === TRANSFORMAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
15	`transform = transforms.Compose([`	Cria pipeline de pré-processamento (transforms).
16	`transforms.Resize((224, 224)),`	Instrução do script no contexto do bloco.
17	`transforms.ToTensor(),`	Instrução do script no contexto do bloco.
18	`transforms.Normalize(mean=[0.485, 0.456, 0.406],`	Atribuição: define/atualiza variável ou objeto.
19	`std=[0.229, 0.224, 0.225])`	Atribuição: define/atualiza variável ou objeto.
20	`])`	Instrução do script no contexto do bloco.
21		Linha em branco; separa blocos lógicos.
22	`# === CARREGAR MODELO ===`	Comentário; documenta a intenção do bloco (não executa).
23	`model = models.alexnet(pretrained=False)`	Carrega uma arquitetura CNN (em geral pré-treinada) para transfer learning.
24	`model.classifier[6] = torch.nn.Linear( model.classifier[6].in_features, len(classes))`	Atribuição: define/atualiza variável ou objeto.
25	`model.load_state_dict(torch.load(caminho_modelo, map_location='cpu'))`	Atribuição: define/atualiza variável ou objeto.

4.15. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
26	`model.eval()`	Ativa modo de avaliação/inferência.
27		Linha em branco; separa blocos lógicos.
28	`# === AVALIAÇÃO ===`	Comentário; documenta a intenção do bloco (não executa).
29	`y_true = []`	Atribuição: define/atualiza variável ou objeto.
30	`y_pred = []`	Atribuição: define/atualiza variável ou objeto.
31		Linha em branco; separa blocos lógicos.
32	`for classe in classes:`	Inicia um laço de iteração.
33	`pasta_classe = os.path.join(caminho_dataset, classe)`	Atribuição: define/atualiza variável ou objeto.
34	`for img_nome in os.listdir(pasta_classe):`	Inicia um laço de iteração.
35	`if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):`	Estrutura condicional.
36	`continue`	Instrução do script no contexto do bloco.
37	`img_path = os.path.join(pasta_classe, img_nome)`	Atribuição: define/atualiza variável ou objeto.
38	`img = Image.open(img_path).convert('RGB')`	Atribuição: define/atualiza variável ou objeto.
39	`img_tensor = transform(img).unsqueeze(0)`	Atribuição: define/atualiza variável ou objeto.
40		Linha em branco; separa blocos lógicos.
41	`with torch.no_grad():`	Desativa gradientes para inferência mais rápida.
42	`output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.
43	`_, pred = torch.max(output, 1)`	Atribuição: define/atualiza variável ou objeto.
44		Linha em branco; separa blocos lógicos.
45	`y_true.append(class_to_idx[classe])`	Instrução do script no contexto do bloco.
46	`y_pred.append(pred.item())`	Instrução do script no contexto do bloco.
47		Linha em branco; separa blocos lógicos.
48	`# === RELATÓRIO ===`	Comentário; documenta a intenção do bloco (não executa).
49	`print("=== Classification Report ===")`	Imprime informação de acompanhamento no console.
50	`print(classification_report(y_true, y_pred, target_names=classes))`	Imprime informação de acompanhamento no console.

4.15. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
51		Linha em branco; separa blocos lógicos.
52	`print("=== Matriz de Confusão ===")`	Imprime informação de acompanhamento no console.
53	`cm = confusion_matrix(y_true, y_pred)`	Calcula matriz de confusão.
54	`plt.figure(figsize=(8, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
55	`sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=classes, yticklabels=classes)`	Atribuição: define/atualiza variável ou objeto.
56	`plt.xlabel("Classe prevista")`	Comando de plotagem/ajuste de figura (Matplotlib).
57	`plt.ylabel("Classe verdadeira")`	Comando de plotagem/ajuste de figura (Matplotlib).
58	`plt.title("■ Matriz de Confusão")`	Comando de plotagem/ajuste de figura (Matplotlib).
59	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
60	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the
future, please use 'weights' instead.
warnings.warn(
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since
0.13 and may be removed in the future. The current behavior is equivalent to passing
`weights=None`.
warnings.warn(msg)
```

Saída (texto)

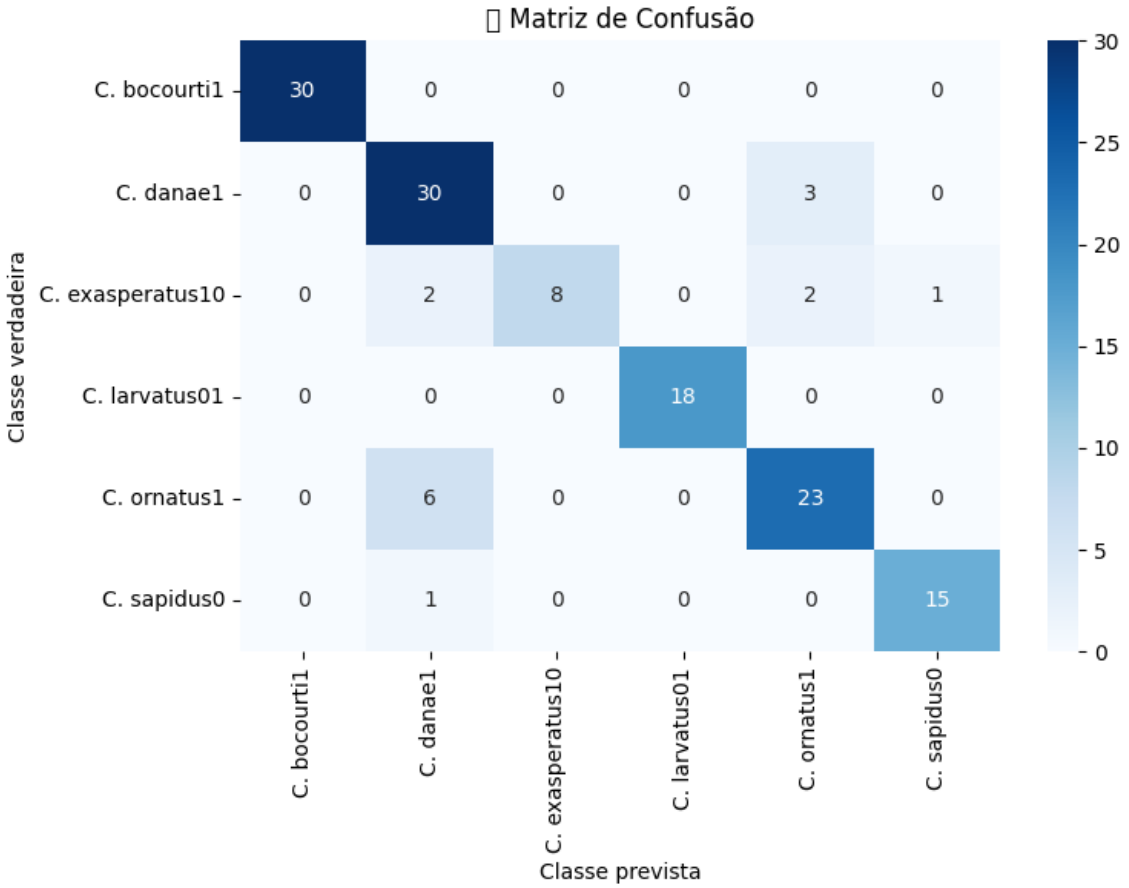
```
=== Classification Report ===
precision recall f1-score support

C. bocourti1 1.00 1.00 1.00 30
C. danae1 0.77 0.91 0.83 33
C. exasperatus10 1.00 0.62 0.76 13
C. larvatus01 1.00 1.00 1.00 18
C. ornatus1 0.82 0.79 0.81 29
C. sapidus0 0.94 0.94 0.94 16

accuracy 0.89 139
macro avg 0.92 0.88 0.89 139
weighted avg 0.90 0.89 0.89 139

=== Matriz de Confusão ===
```

Saída (texto)

```
C:\Users\vitin\AppData\Local\Temp\ipykernel_16104\2567391791.py:59: UserWarning: Glyph 128269
(\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) DejaVu Sans.
plt.tight_layout()
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\IPython\core\pylabtools.py:170:
UserWarning: Glyph 128269 (\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) DejaVu
Sans.
```

```
fig.canvas.print_figure(bytes_io, **kw)
```

Figura 8. Matriz de confusão



Diagonal = acertos; fora da diagonal = confusões.

4.16. === CONFIGURAÇÕES ===

Código e explicação linha a linha

Linha	Código	Explicação
1	`import torch`	Importa bibliotecas necessárias para o pipeline.
2	`import os`	Importa bibliotecas necessárias para o pipeline.
3	`from PIL import Image`	Importa bibliotecas necessárias para o pipeline.
4	`from torchvision import transforms, models`	Importa bibliotecas necessárias para o pipeline.
5	`from sklearn.metrics import classification_report, confusion_matrix`	Importa bibliotecas necessárias para o pipeline.
6	`import numpy as np`	Importa bibliotecas necessárias para o pipeline.
7		Linha em branco; separa blocos lógicos.
8	`# === CONFIGURAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
9	`caminho_dataset = 'siris_dataset_split/external'`	Atribuição: define/atualiza variável ou objeto.
10	`classes = sorted(os.listdir(caminho_dataset)) # os nomes das pastas são os nomes das classes`	Atribuição: define/atualiza variável ou objeto.
11	`class_to_idx = {classe: idx for idx, classe in enumerate(classes)}`	Atribuição: define/atualiza variável ou objeto.
12	`caminho_modelo = './models_gray/modelo_final_mobile_net.pth' # Caminho para o modelo treinado`	Atribuição: define/atualiza variável ou objeto.
13		Linha em branco; separa blocos lógicos.
14	`# === TRANSFORMAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
15	`transform = transforms.Compose([`	Cria pipeline de pré-processamento (transforms).
16	`transforms.Resize((224, 224)),`	Instrução do script no contexto do bloco.
17	`transforms.ToTensor(),`	Instrução do script no contexto do bloco.
18	`transforms.Normalize(mean=[0.485, 0.456, 0.406],`	Atribuição: define/atualiza variável ou objeto.
19	`std=[0.229, 0.224, 0.225])`	Atribuição: define/atualiza variável ou objeto.
20	`])`	Instrução do script no contexto do bloco.
21		Linha em branco; separa blocos lógicos.
22	`# === CARREGAR MODELO ===`	Comentário; documenta a intenção do bloco (não executa).
23	`model = models.mobilenet_v2(pretrained=False)`	Carrega uma arquitetura CNN (em geral pré-treinada) para transfer learning.
24	`model.classifier[1] = torch.nn.Linear(model.last_channel, len(classes))`	Atribuição: define/atualiza variável ou objeto.
25	`model.load_state_dict(torch.load(caminho_modelo, map_location='cpu'))`	Atribuição: define/atualiza variável ou objeto.

4.16. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
26	`model.eval()`	Ativa modo de avaliação/inferência.
27		Linha em branco; separa blocos lógicos.
28		Linha em branco; separa blocos lógicos.
29	`# === AVALIAÇÃO ===`	Comentário; documenta a intenção do bloco (não executa).
30	`y_true = []`	Atribuição: define/atualiza variável ou objeto.
31	`y_pred = []`	Atribuição: define/atualiza variável ou objeto.
32		Linha em branco; separa blocos lógicos.
33	`for classe in classes:`	Inicia um laço de iteração.
34	`pasta_classe = os.path.join(caminho_dataset, classe)`	Atribuição: define/atualiza variável ou objeto.
35	`for img_nome in os.listdir(pasta_classe):`	Inicia um laço de iteração.
36	`if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):`	Estrutura condicional.
37	`continue`	Instrução do script no contexto do bloco.
38	`img_path = os.path.join(pasta_classe, img_nome)`	Atribuição: define/atualiza variável ou objeto.
39	`img = Image.open(img_path).convert('RGB')`	Atribuição: define/atualiza variável ou objeto.
40	`img_tensor = transform(img).unsqueeze(0)`	Atribuição: define/atualiza variável ou objeto.
41		Linha em branco; separa blocos lógicos.
42	`with torch.no_grad():`	Desativa gradientes para inferência mais rápida.
43	`output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.
44	`_, pred = torch.max(output, 1)`	Atribuição: define/atualiza variável ou objeto.
45		Linha em branco; separa blocos lógicos.
46	`y_true.append(class_to_idx[classe])`	Instrução do script no contexto do bloco.
47	`y_pred.append(pred.item())`	Instrução do script no contexto do bloco.
48		Linha em branco; separa blocos lógicos.
49	`# === RELATÓRIO ===`	Comentário; documenta a intenção do bloco (não executa).
50	`print("=== Classification Report ===")`	Imprime informação de acompanhamento no console.

4.16. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
51	`print(classification_report(y_true, y_pred, target_names=classes))`	Imprime informação de acompanhamento no console.
52		Linha em branco; separa blocos lógicos.
53	`print("=== Matriz de Confusão ===")`	Imprime informação de acompanhamento no console.
54	`cm = confusion_matrix(y_true, y_pred)`	Calcula matriz de confusão.
55	`plt.figure(figsize=(8, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
56	`sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=classes, yticklabels=classes)`	Atribuição: define/atualiza variável ou objeto.
57	`plt.xlabel("Classe prevista")`	Comando de plotagem/ajuste de figura (Matplotlib).
58	`plt.ylabel("Classe verdadeira")`	Comando de plotagem/ajuste de figura (Matplotlib).
59	`plt.title("■ Matriz de Confusão")`	Comando de plotagem/ajuste de figura (Matplotlib).
60	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
61	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the
future, please use 'weights' instead.
warnings.warn(
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since
0.13 and may be removed in the future. The current behavior is equivalent to passing
`weights=None`.
warnings.warn(msg)
```

Saída (texto)

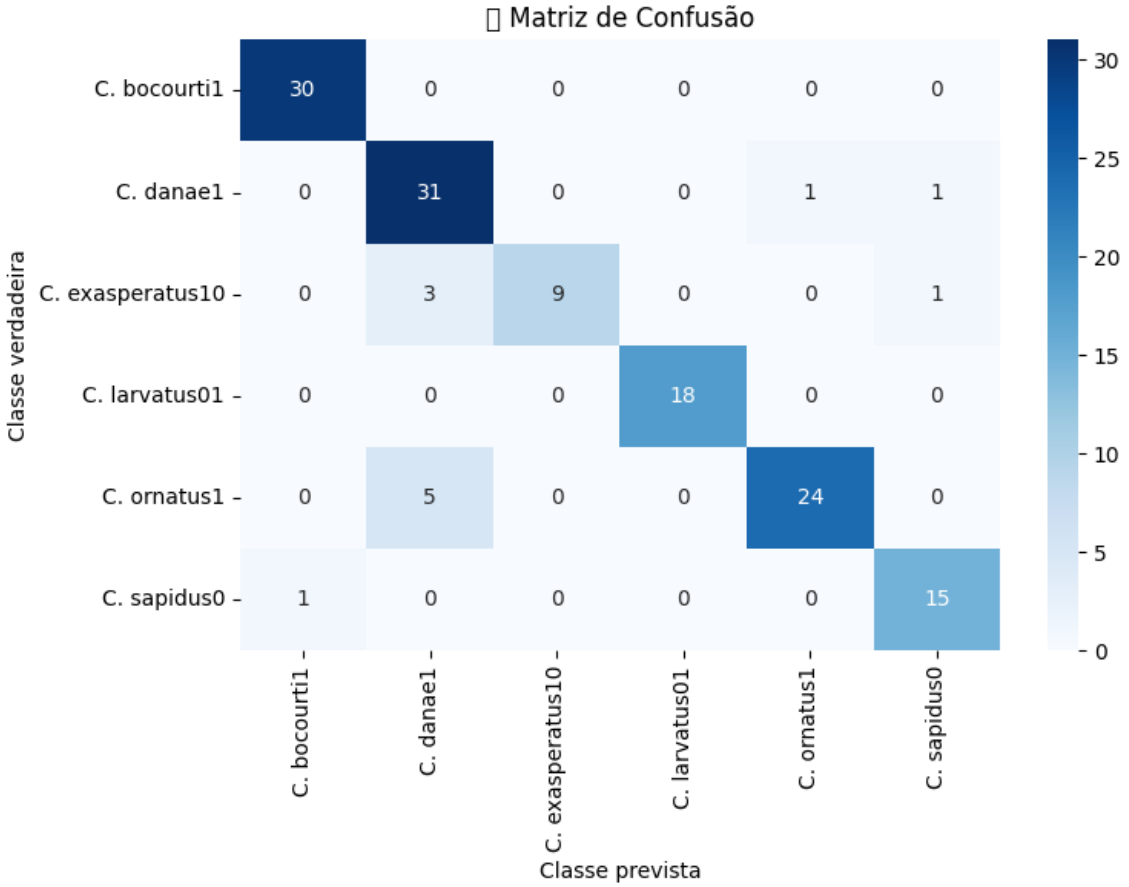
```
=== Classification Report ===
precision recall f1-score support

C. bocourti1 0.97 1.00 0.98 30
C. danae1 0.79 0.94 0.86 33
C. exasperatus10 1.00 0.69 0.82 13
C. larvatus01 1.00 1.00 1.00 18
C. ornatus1 0.96 0.83 0.89 29
C. sapidus0 0.88 0.94 0.91 16

accuracy 0.91 139
macro avg 0.93 0.90 0.91 139
weighted avg 0.92 0.91 0.91 139

=== Matriz de Confusão ===
```

Saída (texto)

```
C:\Users\vitin\AppData\Local\Temp\ipykernel_16104\1842501701.py:60: UserWarning: Glyph 128269
(\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) DejaVu Sans.
plt.tight_layout()
```

```
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\IPython\core\pylabtools.py:170:
UserWarning: Glyph 128269 (\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) DejaVu
Sans.
fig.canvas.print_figure(bytes_io, **kw)
```

Figura 9. Matriz de confusão



Diagonal = acertos; fora da diagonal = confusões.

4.17. === CONFIGURAÇÕES ===

Código e explicação linha a linha

Linha	Código	Explicação
1	`import os`	Importa bibliotecas necessárias para o pipeline.
2	`import torch`	Importa bibliotecas necessárias para o pipeline.
3	`from PIL import Image`	Importa bibliotecas necessárias para o pipeline.
4	`import matplotlib.pyplot as plt`	Importa bibliotecas necessárias para o pipeline.
5	`from torchvision import models, transforms`	Importa bibliotecas necessárias para o pipeline.
6	`from torchcam.methods import GradCAM`	Importa bibliotecas necessárias para o pipeline.
7	`from torchcam.utils import overlay_mask`	Importa bibliotecas necessárias para o pipeline.
8	`from torchvision.transforms.functional import to_pil_image`	Importa bibliotecas necessárias para o pipeline.
9		Linha em branco; separa blocos lógicos.
10	`# === CONFIGURAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
11	`device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
12	`modelo_path = "./models_gray/modelo_final_res_net.pth"`	Atribuição: define/atualiza variável ou objeto.
13	`caminho_dataset = 'siris_dataset_split/external'`	Atribuição: define/atualiza variável ou objeto.
14	`classes = sorted(os.listdir(caminho_dataset))`	Atribuição: define/atualiza variável ou objeto.
15		Linha em branco; separa blocos lógicos.
16	`# === TRANSFORMAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
17	`transform = transforms.Compose([`	Cria pipeline de pré-processamento (transforms).
18	`transforms.Resize((224, 224)),`	Instrução do script no contexto do bloco.
19	`transforms.ToTensor(),`	Instrução do script no contexto do bloco.
20	`transforms.Normalize(mean=[0.485, 0.456, 0.406],`	Atribuição: define/atualiza variável ou objeto.
21	`std=[0.229, 0.224, 0.225])`	Atribuição: define/atualiza variável ou objeto.
22	`])`	Instrução do script no contexto do bloco.
23		Linha em branco; separa blocos lógicos.
24	`# === CARREGAR MODELO ===`	Comentário; documenta a intenção do bloco (não executa).
25	`model = models.resnet18(pretrained=False)`	Carrega uma arquitetura CNN (em geral pré-treinada) para transfer learning.
26	`model.fc = torch.nn.Linear(model.fc.in_features, len(classes))`	Atribuição: define/atualiza variável ou objeto.

4.17. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
27	`model.load_state_dict(torch.load(modelo_path, map_location=device))`	Atribuição: define/atualiza variável ou objeto.
28	`model.to(device)`	Instrução do script no contexto do bloco.
29	`model.eval()`	Ativa modo de avaliação/inferência.
30		Linha em branco; separa blocos lógicos.
31	`plt.figure(figsize=(15, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
32	`cont = 0`	Atribuição: define/atualiza variável ou objeto.
33	`for idx, classe in enumerate(classes):`	Inicia um laço de iteração.
34	`pasta_classe = os.path.join(caminho_dataset, classe)`	Atribuição: define/atualiza variável ou objeto.
35	`img_nome = os.listdir(pasta_classe)[0]`	Atribuição: define/atualiza variável ou objeto.
36		Linha em branco; separa blocos lógicos.
37	`if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):`	Estrutura condicional.
38	`continue`	Instrução do script no contexto do bloco.
39		Linha em branco; separa blocos lógicos.
40	`# === CARREGAR IMAGEM ===`	Comentário; documenta a intenção do bloco (não executa).
41	`img_path = os.path.join(pasta_classe, img_nome)`	Atribuição: define/atualiza variável ou objeto.
42	`img = Image.open(img_path).convert('RGB')`	Atribuição: define/atualiza variável ou objeto.
43		Linha em branco; separa blocos lógicos.
44	`img_tensor = transform(img).unsqueeze(0)`	Atribuição: define/atualiza variável ou objeto.
45		Linha em branco; separa blocos lógicos.
46	`output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.
47	`_, pred = torch.max(output, 1)`	Atribuição: define/atualiza variável ou objeto.
48		Linha em branco; separa blocos lógicos.
49	`# === APLICAR Grad-CAM ===`	Comentário; documenta a intenção do bloco (não executa).
50	`cam_extractor = GradCAM(model, target_layer='layer4')`	Configura Grad-CAM para interpretabilidade.
51		Linha em branco; separa blocos lógicos.
52	`output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.

4.17. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
53	`pred_class = output.argmax().item()`	Atribuição: define/atualiza variável ou objeto.
54		Linha em branco; separa blocos lógicos.
55	`# Extrair ativação e sobrepor com imagem original`	Comentário; documenta a intenção do bloco (não executa).
56	`activation_map = cam_extractor(pred_class, output)[0].cpu()`	Atribuição: define/atualiza variável ou objeto.
57		Linha em branco; separa blocos lógicos.
58	`# Converter imagem e ativação para PIL`	Comentário; documenta a intenção do bloco (não executa).
59	`activation_pil = to_pil_image(activation_map, mode='F')`	Atribuição: define/atualiza variável ou objeto.
60		Linha em branco; separa blocos lógicos.
61	`# Gerar imagem com sobreposição (mapa de calor)`	Comentário; documenta a intenção do bloco (não executa).
62	`result = overlay_mask(img, activation_pil, alpha=0.5)`	Atribuição: define/atualiza variável ou objeto.
63		Linha em branco; separa blocos lógicos.
64	`# Mostrar com matplotlib`	Comentário; documenta a intenção do bloco (não executa).
65	`img_nome = img_nome.split('_')[0]`	Atribuição: define/atualiza variável ou objeto.
66	`cont+=1`	Atribuição: define/atualiza variável ou objeto.
67	`plt.subplot(3,6, idx + cont)`	Comando de plotagem/ajuste de figura (Matplotlib).
68	`plt.title(f"Imagem: {img_nome}")`	Comando de plotagem/ajuste de figura (Matplotlib).
69	`plt.imshow(img)`	Comando de plotagem/ajuste de figura (Matplotlib).
70	`cont+=1`	Atribuição: define/atualiza variável ou objeto.
71	`plt.subplot(3,6, idx + cont)`	Comando de plotagem/ajuste de figura (Matplotlib).
72	`plt.axis("off")`	Comando de plotagem/ajuste de figura (Matplotlib).
73	`plt.imshow(result)`	Comando de plotagem/ajuste de figura (Matplotlib).
74	`plt.title(f"Classe prevista: {classes[pred_class]}")`	Comando de plotagem/ajuste de figura (Matplotlib).
75	`plt.axis("off")`	Comando de plotagem/ajuste de figura (Matplotlib).
76	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

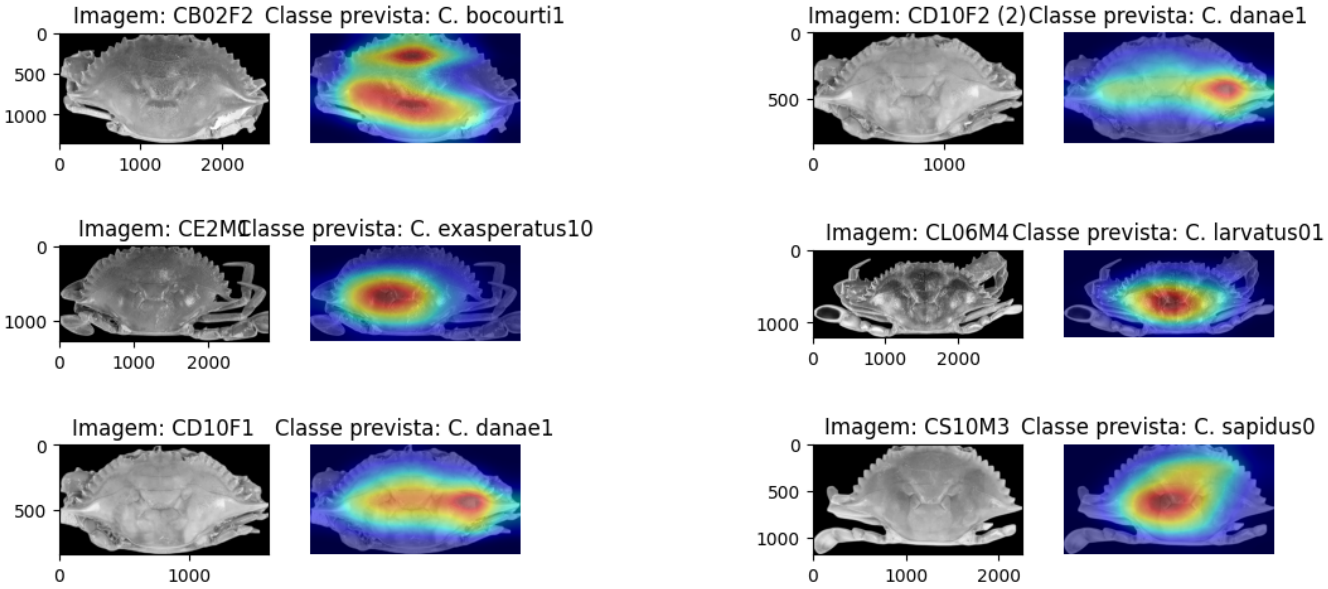
Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the
future, please use 'weights' instead.
warnings.warn(
c:\Users\vitin\Documents\LABEEC\siri\Lib\site-packages\torchvision\models\_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since
0.13 and may be removed in the future. The current behavior is equivalent to passing
```

```
`weights=None`.
warnings.warn(msg)
```

Figura 10. Grad-CAM



Mostra regiões mais influentes na decisão.

4.18. === CONFIGURAÇÕES ===

Código e explicação linha a linha

Linha	Código	Explicação
1	`import os`	Importa bibliotecas necessárias para o pipeline.
2	`import torch`	Importa bibliotecas necessárias para o pipeline.
3	`from PIL import Image`	Importa bibliotecas necessárias para o pipeline.
4	`import matplotlib.pyplot as plt`	Importa bibliotecas necessárias para o pipeline.
5	`from torchvision import models, transforms`	Importa bibliotecas necessárias para o pipeline.
6	`from torchcam.methods import GradCAM`	Importa bibliotecas necessárias para o pipeline.
7	`from torchcam.utils import overlay_mask`	Importa bibliotecas necessárias para o pipeline.
8	`from torchvision.transforms.functional import to_pil_image`	Importa bibliotecas necessárias para o pipeline.
9		Linha em branco; separa blocos lógicos.
10	`# === CONFIGURAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
11	`device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
12	`modelo_path = "./models_gray/modelo_final_alex_net.pth"`	Atribuição: define/atualiza variável ou objeto.
13	`caminho_dataset = 'siris_dataset_split/external'`	Atribuição: define/atualiza variável ou objeto.
14	`classes = sorted(os.listdir(caminho_dataset))`	Atribuição: define/atualiza variável ou objeto.
15		Linha em branco; separa blocos lógicos.
16	`# === TRANSFORMAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
17	`transform = transforms.Compose([`	Cria pipeline de pré-processamento (transforms).
18	`transforms.Resize((224, 224)),`	Instrução do script no contexto do bloco.
19	`transforms.ToTensor(),`	Instrução do script no contexto do bloco.
20	`transforms.Normalize(mean=[0.485, 0.456, 0.406],`	Atribuição: define/atualiza variável ou objeto.
21	`std=[0.229, 0.224, 0.225])`	Atribuição: define/atualiza variável ou objeto.
22	`])`	Instrução do script no contexto do bloco.
23		Linha em branco; separa blocos lógicos.
24	`# === CARREGAR MODELO ===`	Comentário; documenta a intenção do bloco (não executa).
25	`model = models.alexnet(pretrained=False)`	Carrega uma arquitetura CNN (em geral pré-treinada) para transfer learning.
26	`model.classifier[6] = torch.nn.Linear( model.classifier[6].in_features, len(classes))`	Atribuição: define/atualiza variável ou objeto.

4.18. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
27	`model.load_state_dict(torch.load(modelo_path, map_location='cpu'))`	Atribuição: define/atualiza variável ou objeto.
28	`model.eval()`	Ativa modo de avaliação/inferência.
29		Linha em branco; separa blocos lógicos.
30	`plt.figure(figsize=(15, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
31	`cont = 0`	Atribuição: define/atualiza variável ou objeto.
32	`for idx, classe in enumerate(classes):`	Inicia um laço de iteração.
33	`pasta_classe = os.path.join(caminho_dataset, classe)`	Atribuição: define/atualiza variável ou objeto.
34	`img_nome = os.listdir(pasta_classe)[0]`	Atribuição: define/atualiza variável ou objeto.
35		Linha em branco; separa blocos lógicos.
36	`if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):`	Estrutura condicional.
37	`continue`	Instrução do script no contexto do bloco.
38		Linha em branco; separa blocos lógicos.
39	`# === CARREGAR IMAGEM ===`	Comentário; documenta a intenção do bloco (não executa).
40	`img_path = os.path.join(pasta_classe, img_nome)`	Atribuição: define/atualiza variável ou objeto.
41	`img = Image.open(img_path).convert('RGB')`	Atribuição: define/atualiza variável ou objeto.
42		Linha em branco; separa blocos lógicos.
43	`img_tensor = transform(img).unsqueeze(0)`	Atribuição: define/atualiza variável ou objeto.
44		Linha em branco; separa blocos lógicos.
45	`output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.
46	`_, pred = torch.max(output, 1)`	Atribuição: define/atualiza variável ou objeto.
47		Linha em branco; separa blocos lógicos.
48	`# === APLICAR Grad-CAM ===`	Comentário; documenta a intenção do bloco (não executa).
49	`cam_extractor = GradCAM(model, target_layer='features.11')`	Configura Grad-CAM para interpretabilidade.
50		Linha em branco; separa blocos lógicos.
51	`output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.
52	`pred_class = output.argmax().item()`	Atribuição: define/atualiza variável ou objeto.

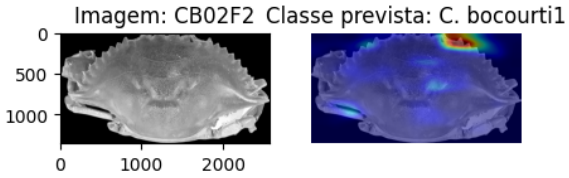
4.18. === CONFIGURAÇÕES === (continuação)

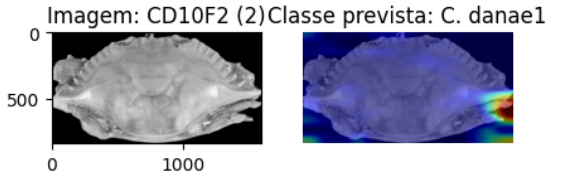
Código e explicação linha a linha

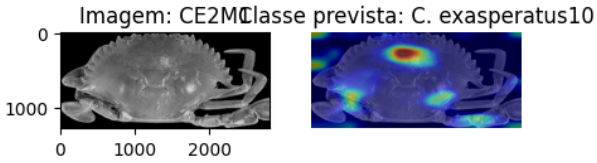
Linha	Código	Explicação
53		Linha em branco; separa blocos lógicos.
54	`# Extrair ativação e sobrepor com imagem original`	Comentário; documenta a intenção do bloco (não executa).
55	`activation_map = cam_extractor(pred_class, output)[0].cpu()`	Atribuição: define/atualiza variável ou objeto.
56		Linha em branco; separa blocos lógicos.
57	`# Converter imagem e ativação para PIL`	Comentário; documenta a intenção do bloco (não executa).
58	`activation_pil = to_pil_image(activation_map, mode='F')`	Atribuição: define/atualiza variável ou objeto.
59		Linha em branco; separa blocos lógicos.
60	`# Gerar imagem com sobreposição (mapa de calor)`	Comentário; documenta a intenção do bloco (não executa).
61	`result = overlay_mask(img, activation_pil, alpha=0.5)`	Atribuição: define/atualiza variável ou objeto.
62		Linha em branco; separa blocos lógicos.
63	`# Mostrar com matplotlib`	Comentário; documenta a intenção do bloco (não executa).
64	`img_nome = img_nome.split('_')[0]`	Atribuição: define/atualiza variável ou objeto.
65	`cont+=1`	Atribuição: define/atualiza variável ou objeto.
66	`plt.subplot(3,6, idx + cont)`	Comando de plotagem/ajuste de figura (Matplotlib).
67	`plt.title(f"Imagem: {img_nome}")`	Comando de plotagem/ajuste de figura (Matplotlib).
68	`plt.imshow(img)`	Comando de plotagem/ajuste de figura (Matplotlib).
69	`cont+=1`	Atribuição: define/atualiza variável ou objeto.
70	`plt.subplot(3,6, idx + cont)`	Comando de plotagem/ajuste de figura (Matplotlib).
71	`plt.axis("off")`	Comando de plotagem/ajuste de figura (Matplotlib).
72	`plt.imshow(result)`	Comando de plotagem/ajuste de figura (Matplotlib).
73	`plt.title(f"Classe prevista: {classes[pred_class]}")`	Comando de plotagem/ajuste de figura (Matplotlib).
74	`plt.axis("off")`	Comando de plotagem/ajuste de figura (Matplotlib).
75	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

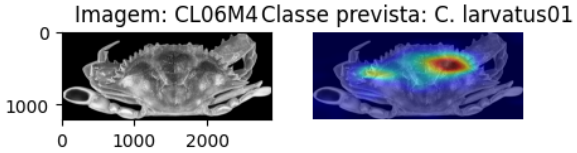
Resultados gerados nesta célula

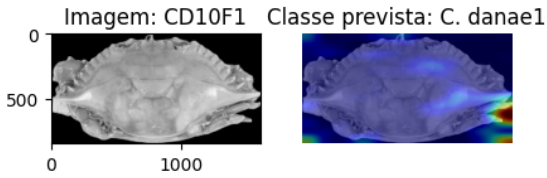
Figura 11. Grad-CAM

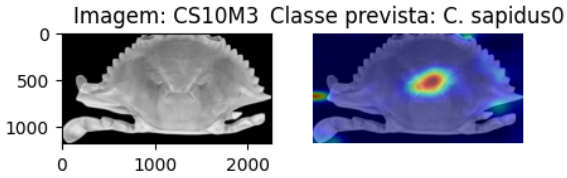












Mostra regiões mais influentes na decisão.

4.19. === CONFIGURAÇÕES ===

Código e explicação linha a linha

Linha	Código	Explicação
1	`import os`	Importa bibliotecas necessárias para o pipeline.
2	`import torch`	Importa bibliotecas necessárias para o pipeline.
3	`from PIL import Image`	Importa bibliotecas necessárias para o pipeline.
4	`import matplotlib.pyplot as plt`	Importa bibliotecas necessárias para o pipeline.
5	`from torchvision import models, transforms`	Importa bibliotecas necessárias para o pipeline.
6	`from torchcam.methods import GradCAM`	Importa bibliotecas necessárias para o pipeline.
7	`from torchcam.utils import overlay_mask`	Importa bibliotecas necessárias para o pipeline.
8	`from torchvision.transforms.functional import to_pil_image`	Importa bibliotecas necessárias para o pipeline.
9		Linha em branco; separa blocos lógicos.
10	`# === CONFIGURAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
11	`device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
12	`modelo_path = "./models_gray/modelo_final_mobile_net.pth"`	Atribuição: define/atualiza variável ou objeto.
13	`caminho_dataset = 'siris_dataset_split/external'`	Atribuição: define/atualiza variável ou objeto.
14	`classes = sorted(os.listdir(caminho_dataset))`	Atribuição: define/atualiza variável ou objeto.
15		Linha em branco; separa blocos lógicos.
16	`# === TRANSFORMAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
17	`transform = transforms.Compose([`	Cria pipeline de pré-processamento (transforms).
18	`transforms.Resize((224, 224)),`	Instrução do script no contexto do bloco.
19	`transforms.ToTensor(),`	Instrução do script no contexto do bloco.
20	`transforms.Normalize(mean=[0.485, 0.456, 0.406],`	Atribuição: define/atualiza variável ou objeto.
21	`std=[0.229, 0.224, 0.225])`	Atribuição: define/atualiza variável ou objeto.
22	`])`	Instrução do script no contexto do bloco.
23		Linha em branco; separa blocos lógicos.
24	`# === CARREGAR MODELO ===`	Comentário; documenta a intenção do bloco (não executa).
25	`model = models.mobilenet_v2(pretrained=False)`	Carrega uma arquitetura CNN (em geral pré-treinada) para transfer learning.
26	`model.classifier[1] = torch.nn.Linear(model.last_channel, len(classes))`	Atribuição: define/atualiza variável ou objeto.

4.19. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
27	`model.load_state_dict(torch.load(modelo_path, map_location='cpu'))`	Atribuição: define/atualiza variável ou objeto.
28	`model.eval()`	Ativa modo de avaliação/inferência.
29		Linha em branco; separa blocos lógicos.
30	`plt.figure(figsize=(15, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
31	`cont = 0`	Atribuição: define/atualiza variável ou objeto.
32	`for idx, classe in enumerate(classes):`	Inicia um laço de iteração.
33	`pasta_classe = os.path.join(caminho_dataset, classe)`	Atribuição: define/atualiza variável ou objeto.
34	`img_nome = os.listdir(pasta_classe)[0]`	Atribuição: define/atualiza variável ou objeto.
35		Linha em branco; separa blocos lógicos.
36	`if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):`	Estrutura condicional.
37	`continue`	Instrução do script no contexto do bloco.
38		Linha em branco; separa blocos lógicos.
39	`# === CARREGAR IMAGEM ===`	Comentário; documenta a intenção do bloco (não executa).
40	`img_path = os.path.join(pasta_classe, img_nome)`	Atribuição: define/atualiza variável ou objeto.
41	`img = Image.open(img_path).convert('RGB')`	Atribuição: define/atualiza variável ou objeto.
42		Linha em branco; separa blocos lógicos.
43	`img_tensor = transform(img).unsqueeze(0)`	Atribuição: define/atualiza variável ou objeto.
44		Linha em branco; separa blocos lógicos.
45	`output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.
46	`_, pred = torch.max(output, 1)`	Atribuição: define/atualiza variável ou objeto.
47		Linha em branco; separa blocos lógicos.
48	`# === APLICAR Grad-CAM ===`	Comentário; documenta a intenção do bloco (não executa).
49	`cam_extractor = GradCAM(model, target_layer='features.18')`	Configura Grad-CAM para interpretabilidade.
50		Linha em branco; separa blocos lógicos.
51	`output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.
52	`pred_class = output.argmax().item()`	Atribuição: define/atualiza variável ou objeto.

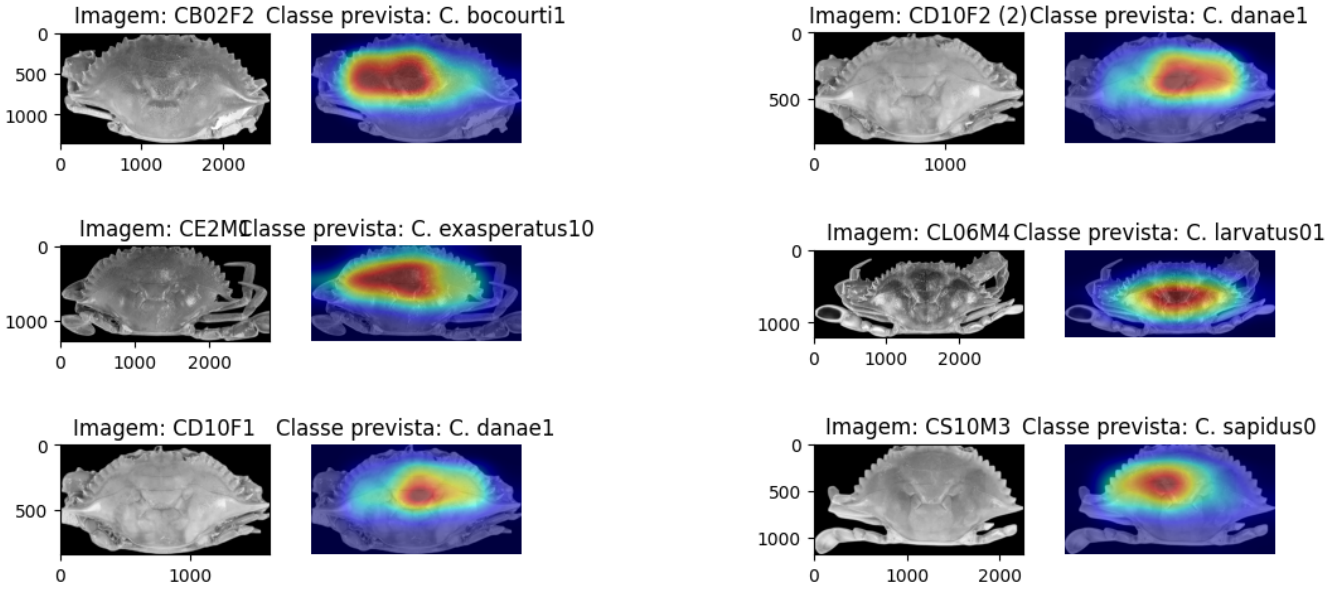
4.19. === CONFIGURAÇÕES === (continuação)

Código e explicação linha a linha

Linha	Código	Explicação
53		Linha em branco; separa blocos lógicos.
54	`# Extrair ativação e sobrepor com imagem original`	Comentário; documenta a intenção do bloco (não executa).
55	`activation_map = cam_extractor(pred_class, output)[0].cpu()`	Atribuição: define/atualiza variável ou objeto.
56		Linha em branco; separa blocos lógicos.
57	`# Converter imagem e ativação para PIL`	Comentário; documenta a intenção do bloco (não executa).
58	`activation_pil = to_pil_image(activation_map, mode='F')`	Atribuição: define/atualiza variável ou objeto.
59		Linha em branco; separa blocos lógicos.
60	`# Gerar imagem com sobreposição (mapa de calor)`	Comentário; documenta a intenção do bloco (não executa).
61	`result = overlay_mask(img, activation_pil, alpha=0.5)`	Atribuição: define/atualiza variável ou objeto.
62		Linha em branco; separa blocos lógicos.
63	`# Mostrar com matplotlib`	Comentário; documenta a intenção do bloco (não executa).
64	`img_nome = img_nome.split('_')[0]`	Atribuição: define/atualiza variável ou objeto.
65	`cont+=1`	Atribuição: define/atualiza variável ou objeto.
66	`plt.subplot(3,6, idx + cont)`	Comando de plotagem/ajuste de figura (Matplotlib).
67	`plt.title(f"Imagem: {img_nome}")`	Comando de plotagem/ajuste de figura (Matplotlib).
68	`plt.imshow(img)`	Comando de plotagem/ajuste de figura (Matplotlib).
69	`cont+=1`	Atribuição: define/atualiza variável ou objeto.
70	`plt.subplot(3,6, idx + cont)`	Comando de plotagem/ajuste de figura (Matplotlib).
71	`plt.axis("off")`	Comando de plotagem/ajuste de figura (Matplotlib).
72	`plt.imshow(result)`	Comando de plotagem/ajuste de figura (Matplotlib).
73	`plt.title(f"Classe prevista: {classes[pred_class]}")`	Comando de plotagem/ajuste de figura (Matplotlib).
74	`plt.axis("off")`	Comando de plotagem/ajuste de figura (Matplotlib).
75	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Figura 12. Grad-CAM



Mostra regiões mais influentes na decisão.