

Shrimp Vision Classifier

Classificação de espécies por Visão Computacional (Livro + Notebook)

https://github.com/fulvioamf/shrimp_vision_classifier

Fulvio Freire (UFRN — LABEEC)

Versão: v1.0.0 — Atualizado: 31/12/2025

Este PDF é um material autodidata que acompanha o repositório GitHub. Ele explica o pipeline de visão computacional para classificação de espécies, interpreta as saídas (tabelas e figuras) e detalha, linha a linha, as funções utilizadas no notebook.

Sumário

1. Introdução	3
2. Como se orientar no repositório	4
3. Como interpretar os resultados	5
4. Tutorial linha a linha do notebook	6

1. Introdução

Este livro (Shrimp Vision Classifier) foca na execução e interpretação do notebook de camarões. Para a fundamentação conceitual completa (IA, aprendizado de máquina, redes neurais, deep learning, visão computacional e transfer learning), utilize o ****livro Crab Vision Classifier**** como referência-base.

Como acessar o livro-base (Crab): repositório https://github.com/fulvioamf/crab_vision_classifier e PDF em ``docs/livro/tutorial_crab_vision_livro_v15.pdf``. Para referência bibliográfica, você pode usar o DOI de versão do livro Crab: 10.5281/zenodo.18111319.

A partir daqui, este livro cobre: (i) como se orientar no repositório do Shrimp, (ii) como ler as métricas e figuras geradas e (iii) o tutorial linha a linha do notebook.

2. Como se orientar no repositório

Repositório (Shrimp): https://github.com/fulvioamf/shrimp_vision_classifier. A estrutura pode variar ligeiramente conforme a evolução do projeto, mas a lógica geral é a mesma: um notebook principal, um PDF (livro) e arquivos de suporte (dependências e metadados).

Arquivo/pasta	O que é / para que serve
README.md	Página inicial do projeto (visão geral, instruções, links).
docs/livro/	Contém o livro em PDF. Neste repositório: docs/livro/tutorial_shrimp_vision_livro.pdf.
shrimp_vision_gray.ipynb	Notebook principal (pipeline completo e resultados).
requirements.txt	Dependências Python para reproduzir o ambiente.
LICENSE	Licença do repositório (uso e redistribuição).
CITATION.cff	Metadados de citação (GitHub/Zenodo).

Execução rápida (reprodução mínima)

- 1) Baixe/clone o repositório do GitHub.
- 2) Crie um ambiente virtual (venv/conda) e instale as dependências: ``pip install -r requirements.txt``.
- 3) Abra e execute o notebook (Jupyter/VS Code).
- 4) Confirme a estrutura do dataset (pastas por classe) antes de treinar.
- 5) Rode o treinamento e, ao final, gere as figuras finais (loss/acc, matriz de confusão, ROC e Grad-CAM).

Observação sobre dados: por razões de publicação científica, o repositório pode conter apenas um subconjunto do dataset (ou um zip reduzido) para demonstração. O pipeline, no entanto, é o mesmo.

3. Como interpretar os resultados

O notebook treina e compara arquiteturas diferentes (ResNet18, AlexNet e MobileNetV2) usando transfer learning e/ou fine-tuning. Ao final, ele gera métricas e figuras para avaliar desempenho e interpretabilidade. Nesta seção, a interpretação é apresentada no padrão 'artigo': o que cada figura mede e como ler.

3.1 Curvas de loss e acurácia

As curvas de **loss** (perda) e **acurácia** ao longo das épocas ajudam a verificar se o treinamento converge. Em geral, espera-se queda de loss e aumento de acurácia, com estabilidade no conjunto de validação/teste.

3.2 Matriz de confusão

A matriz de confusão mostra, classe a classe, onde o modelo acerta e erra. Uma boa matriz tem valores concentrados na diagonal principal. Erros sistemáticos entre duas classes apontam similaridade morfológica, ruído ou necessidade de mais dados.

3.3 Curvas ROC e AUC (multi-classe)

Em classificação multi-classe, a ROC geralmente é calculada como **one-vs-rest** (cada classe contra todas as outras). A **AUC** resume o poder discriminativo: quanto mais próxima de 1, melhor a separação.

3.4 Grad-CAM (interpretabilidade)

Grad-CAM destaca regiões da imagem que mais contribuíram para a predição. É útil para detectar vieses (por exemplo, foco no fundo) e para validar se o modelo está usando estruturas anatômicas relevantes.

Na Seção 4, as figuras são reapresentadas no contexto exato da célula que as gera, com comentários objetivos sobre o que observar.

4. Tutorial linha a linha do notebook

A seguir, o notebook é decomposto em blocos (células) e, dentro de cada bloco, cada linha de código é explicada. Quando uma célula gera saídas (texto/tabelas/figuras), elas são mostradas e interpretadas.

4.1 Imports e configurações iniciais

Linha	Código	Explicação
1	<code>import numpy as np</code>	Importa bibliotecas/módulos necessários.
2	<code>import torch</code>	Importa bibliotecas/módulos necessários.
3	<code>import torch.nn as nn</code>	Importa bibliotecas/módulos necessários.
4	<code>import torch.optim as optim</code>	Importa bibliotecas/módulos necessários.
5	<code>from torchvision import datasets, transforms, models</code>	Importa bibliotecas/módulos necessários.
6	<code>from torch.utils.data import DataLoader</code>	Importa bibliotecas/módulos necessários.
7	<code>import os</code>	Importa bibliotecas/módulos necessários.
8	<code>import matplotlib.pyplot as plt</code>	Importa bibliotecas/módulos necessários.
9	<code>from sklearn.metrics import confusion_matrix</code>	Importa bibliotecas/módulos necessários.
10	<code>from sklearn.metrics import classification_report</code>	Importa bibliotecas/módulos necessários.
11	<code>import seaborn as sns</code>	Importa bibliotecas/módulos necessários.
12	<code>import random</code>	Importa bibliotecas/módulos necessários.
13	<code>from sklearn.metrics import roc_curve, auc</code>	Importa bibliotecas/módulos necessários.
14	<code>from sklearn.preprocessing import label_binarize</code>	Importa bibliotecas/módulos necessários.
15	<code>from sklearn.metrics import accuracy_score, f1_score</code>	Importa bibliotecas/módulos necessários.
16		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
17	<code>SEED = 42</code>	Atribuição: define/atualiza variável ou objeto.
18	<code>random.seed(SEED)</code>	Instrução do script no contexto do bloco.
19	<code>np.random.seed(SEED)</code>	Instrução do script no contexto do bloco.
20	<code>torch.manual_seed(SEED)</code>	Instrução do script no contexto do bloco.
21	<code>torch.cuda.manual_seed(SEED)</code>	Instrução do script no contexto do bloco.
22	<code>torch.backends.cudnn.deterministic = True</code>	Atribuição: define/atualiza variável ou objeto.
23	<code>torch.backends.cudnn.benchmark = False</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
1	<code># import os</code>	Comentário; documenta a intenção do bloco (não executa).
2	<code># import shutil</code>	Comentário; documenta a intenção do bloco (não executa).
3		Linha em branco; separa blocos lógicos.
4	<code># def mover_imagens_para_pasta_raiz(dataset_path):</code>	Comentário; documenta a intenção do bloco (não executa).
5	<code># for classe in os.listdir(dataset_path):</code>	Comentário; documenta a intenção do bloco (não executa).
6	<code># classe_path = os.path.join(dataset_path, classe)</code>	Comentário; documenta a intenção do bloco (não executa).
7		Linha em branco; separa blocos lógicos.
8	<code># # Pular se não for pasta</code>	Comentário; documenta a intenção do bloco (não executa).
9	<code># if not os.path.isdir(classe_path):</code>	Comentário; documenta a intenção do bloco (não executa).
10	<code># continue</code>	Comentário; documenta a intenção do bloco (não executa).
11		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
12	<code># for raiz, subdirs, arquivos in os.walk(classe_path):</code>	Comentário; documenta a intenção do bloco (não executa).
13	<code># for arquivo in arquivos:</code>	Comentário; documenta a intenção do bloco (não executa).
14	<code># if arquivo.lower().endswith(('.jpg', '.jpeg', '.png')):</code>	Comentário; documenta a intenção do bloco (não executa).
15	<code># origem = os.path.join(raiz, arquivo)</code>	Comentário; documenta a intenção do bloco (não executa).
16	<code># destino = os.path.join(classe_path, arquivo)</code>	Comentário; documenta a intenção do bloco (não executa).
17		Linha em branco; separa blocos lógicos.
18	<code># # Evitar sobrescrever se nome repetir</code>	Comentário; documenta a intenção do bloco (não executa).
19	<code># base, ext = os.path.splitext(arquivo)</code>	Comentário; documenta a intenção do bloco (não executa).
20	<code># contador = 1</code>	Comentário; documenta a intenção do bloco (não executa).
21	<code># while os.path.exists(destino):</code>	Comentário; documenta a intenção do bloco (não executa).
22	<code># destino = os.path.join(classe_path, f"{base}_{contador}{ext}")</code>	Comentário; documenta a intenção do bloco (não executa).
23	<code># contador += 1</code>	Comentário; documenta a intenção do bloco (não executa).
24		Linha em branco; separa blocos lógicos.
25	<code># shutil.copy2(origem, destino)</code>	Comentário; documenta a intenção do bloco (não executa).
26		Linha em branco; separa blocos lógicos.
27	<code># print("✅ Imagens copiadas para o nível da classe com sucesso!")</code>	Comentário; documenta a intenção do bloco (não executa).
28		Linha em branco; separa blocos lógicos.
29	<code># # Use o caminho onde estão as classes</code>	Comentário; documenta a intenção do bloco (não executa).
30	<code># mover_imagens_para_pasta_raiz("camaroes_dataset")</code>	Comentário; documenta a intenção do bloco (não executa).

4.2 Separando dataset

Linha	Código	Explicação
1	<code>import os</code>	Importa bibliotecas/módulos necessários.
2	<code>import shutil</code>	Importa bibliotecas/módulos necessários.
3	<code>import random</code>	Importa bibliotecas/módulos necessários.
4	<code>import cv2 as cv</code>	Importa bibliotecas/módulos necessários.
5	<code># Caminhos</code>	Comentário; documenta a intenção do bloco (não executa).
6	<code>original_dataset = "camaroos_dataset/images"</code>	Atribuição: define/atualiza variável ou objeto.
7	<code>destino_base = "camaroos_dataset_split"</code>	Atribuição: define/atualiza variável ou objeto.
8	<code>train_path = os.path.join(destino_base, "train")</code>	Atribuição: define/atualiza variável ou objeto.
9	<code>test_path = os.path.join(destino_base, "test")</code>	Atribuição: define/atualiza variável ou objeto.
10	<code>external_path = os.path.join(destino_base, "external")</code>	Atribuição: define/atualiza variável ou objeto.
11	<code>split_train = 0.7</code>	Atribuição: define/atualiza variável ou objeto.
12	<code>split_test = 0.2</code>	Atribuição: define/atualiza variável ou objeto.
13	<code>split_external = 0.1 # 10% para avaliação final</code>	Atribuição: define/atualiza variável ou objeto.
14		Linha em branco; separa blocos lógicos.
15		Linha em branco; separa blocos lógicos.
16	<code># Extensões permitidas</code>	Comentário; documenta a intenção do bloco (não executa).
17	<code>extensoes_validas = (".jpg", ".jpeg", ".png", ".bmp")</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
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.
27	print(classe_path)	Imprime informação/diagnóstico no console.
28	if not os.path.isdir(classe_path):	Estrutura condicional (controle de fluxo).
29	continue	Instrução do script no contexto do bloco.
30		Linha em branco; separa blocos lógicos.
31	imagens = [f for f in os.listdir(classe_path) if f.lower().endswith(extensoes_validas)]	Atribuição: define/atualiza variável ou objeto.
32	random.shuffle(imagens)	Instrução do script no contexto do bloco.
33		Linha em branco; separa blocos lógicos.
34	total = len(imagens)	Atribuição: define/atualiza variável ou objeto.
35	train_end = int(total * split_train)	Atribuição: define/atualiza variável ou objeto.
36	test_end = train_end + int(total * split_test)	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
37		Linha em branco; separa blocos lógicos.
38	<code>imagens_train = imagens[:train_end]</code>	Atribuição: define/atualiza variável ou objeto.
39	<code>imagens_test = imagens[train_end:test_end]</code>	Atribuição: define/atualiza variável ou objeto.
40	<code>imagens_external = imagens[test_end:]</code>	Atribuição: define/atualiza variável ou objeto.
41		Linha em branco; separa blocos lógicos.
42	<code># Criar diretórios de saída por classe</code>	Comentário; documenta a intenção do bloco (não executa).
43	<code>os.makedirs(os.path.join(train_path, classe), exist_ok=True)</code>	Atribuição: define/atualiza variável ou objeto.
44	<code>os.makedirs(os.path.join(test_path, classe), exist_ok=True)</code>	Atribuição: define/atualiza variável ou objeto.
45	<code>os.makedirs(os.path.join(external_path, classe), exist_ok=True)</code>	Atribuição: define/atualiza variável ou objeto.
46		Linha em branco; separa blocos lógicos.
47	<code># Copiar imagens de treino</code>	Comentário; documenta a intenção do bloco (não executa).
48	<code>for img in imagens_train:</code>	Inicia um laço de iteração.
49	<code>img_gray = cv.imread(os.path.join(classe_path, img))</code>	Atribuição: define/atualiza variável ou objeto.
50	<code>img_gray = cv.cvtColor(img_gray, cv.COLOR_BGR2GRAY)</code>	Atribuição: define/atualiza variável ou objeto.
51	<code>src = os.path.join(classe_path, img)</code>	Atribuição: define/atualiza variável ou objeto.
52	<code>dst = os.path.join(train_path, classe, img)</code>	Atribuição: define/atualiza variável ou objeto.
53	<code>try:</code>	Instrução do script no contexto do bloco.
54	<code>cv.imwrite(os.path.join(train_path, classe, img), img_gray)</code>	Instrução do script no contexto do bloco.
55	<code>except Exception as e:</code>	Instrução do script no contexto do bloco.

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

Linha	Código	Explicação
75	<code>except Exception as e:</code>	Instrução do script no contexto do bloco.
76	<code>print(f"Erro ao copiar {src}:{e}")</code>	Imprime informação/diagnóstico no console.
77		Linha em branco; separa blocos lógicos.
78	<code>print("✅ Imagens divididas em treino, teste e conjunto externo com sucesso!")</code>	Imprime informação/diagnóstico no console.

Resultados gerados nesta célula

Saída (texto)

```
camaroos_dataset/images\XB
camaroos_dataset/images\XD
camaroos_dataset/images\XK
✅ Imagens divididas em treino, teste e conjunto externo com sucesso!
```

4.3 Treinamento do modelo ResNet18

Linha	Código	Explicação
1	<code>train_losses = []</code>	Atribuição: define/atualiza variável ou objeto.
2	<code>test accuracies = []</code>	Atribuição: define/atualiza variável ou objeto.
3		Linha em branco; separa blocos lógicos.
4	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
5	<code># 1. CONFIGURAÇÕES</code>	Comentário; documenta a intenção do bloco (não executa).
6	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
7	<code>device = torch.device("cuda" if torch.cuda.is_available() else "cpu")</code>	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
8	<code>batch_size = 32</code>	Atribuição: define/atualiza variável ou objeto.
9	<code>num_epochs = 10</code>	Atribuição: define/atualiza variável ou objeto.
10	<code>data_dir = "camaroos_dataset_split" # Caminho para dataset com train/ e test/</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
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([Atribuição: define/atualiza variável ou objeto.
16	transforms.Resize((224, 224)), # Todas as redes esperam 224x224	Define transformação/pré-processamento de imagens (torchvision.transforms).
17	transforms.ToTensor(),	Define transformação/pré-processamento de imagens (torchvision.transforms).
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)	Atribuição: define/atualiza variável ou objeto.
27	test_dataset = datasets.ImageFolder(os.path.join(data_dir, "test"), transform=transform)	Atribuição: define/atualiza variável ou objeto.
28		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
29	<code>train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True, generator=generator)</code>	Atribuição: define/atualiza variável ou objeto.
30	<code>test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False, generator=generator)</code>	Atribuição: define/atualiza variável ou objeto.
31		Linha em branco; separa blocos lógicos.
32	<code>num_classes = len(train_dataset.classes)</code>	Atribuição: define/atualiza variável ou objeto.
33	<code>class_names = train_dataset.classes</code>	Atribuição: define/atualiza variável ou objeto.
34		Linha em branco; separa blocos lógicos.
35	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
36	<code># 4. MODELO (troque aqui)</code>	Comentário; documenta a intenção do bloco (não executa).
37	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
38	<code>model = models.resnet18(pretrained=True)</code>	Atribuição: define/atualiza variável ou objeto.
39	<code>model.fc = nn.Linear(model.fc.in_features, num_classes) # Para ResNet18</code>	Atribuição: define/atualiza variável ou objeto.
40		Linha em branco; separa blocos lógicos.
41	<code>model = model.to(device)</code>	Atribuição: define/atualiza variável ou objeto.
42		Linha em branco; separa blocos lógicos.
43	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
44	<code># 5. LOSS E OTIMIZADOR</code>	Comentário; documenta a intenção do bloco (não executa).
45	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
46	<code>criterion = nn.CrossEntropyLoss()</code>	Atribuição: define/atualiza variável ou objeto.
47	<code>optimizer = optim.Adam(model.parameters(), lr=0.0001)</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
48		Linha em branco; separa blocos lógicos.
49	# =====	Comentário; documenta a intenção do bloco (não executa).
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.
53	model.train()	Coloca o modelo em modo de treino.
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()	Configura o otimizador (atualiza pesos do modelo).
60	outputs = model(images)	Atribuição: define/atualiza variável ou objeto.
61	loss = criterion(outputs, labels)	Atribuição: define/atualiza variável ou objeto.
62	loss.backward()	Define função de perda (critério de otimização).
63	optimizer.step()	Configura o otimizador (atualiza pesos do modelo).
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/diagnóstico no console.

Linha	Código	Explicação
68		Linha em branco; separa blocos lógicos.
69	<code>train_losses.append(running_loss / len(train_loader))</code>	Define função de perda (critério de otimização).
70		Linha em branco; separa blocos lógicos.
71	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
72	<code># 7. AVALIAÇÃO NO TESTE</code>	Comentário; documenta a intenção do bloco (não executa).
73	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
74	<code>model.eval()</code>	Coloca o modelo em modo de avaliação (desativa dropout etc.).
75	<code>correct = 0</code>	Atribuição: define/atualiza variável ou objeto.
76	<code>total = 0</code>	Atribuição: define/atualiza variável ou objeto.
77		Linha em branco; separa blocos lógicos.
78	<code>all_preds = []</code>	Atribuição: define/atualiza variável ou objeto.
79	<code>all_labels = []</code>	Atribuição: define/atualiza variável ou objeto.
80	<code>all_probs = []</code>	Atribuição: define/atualiza variável ou objeto.
81		Linha em branco; separa blocos lógicos.
82	<code>with torch.no_grad():</code>	Instrução do script no contexto do bloco.
83	<code>for images, labels in test_loader:</code>	Inicia um laço de iteração.
84	<code>images, labels = images.to(device), labels.to(device)</code>	Atribuição: define/atualiza variável ou objeto.
85	<code>outputs = model(images)</code>	Atribuição: define/atualiza variável ou objeto.
86	<code>_, predicted = torch.max(outputs, 1)</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
87		Linha em branco; separa blocos lógicos.
88	<code>total += labels.size(0)</code>	Atribuição: define/atualiza variável ou objeto.
89	<code>correct += (predicted == labels).sum().item()</code>	Instrução do script no contexto do bloco.
90		Linha em branco; separa blocos lógicos.
91	<code># Armazena previsões e rótulos verdadeiros</code>	Comentário; documenta a intenção do bloco (não executa).
92	<code>probs = torch.softmax(outputs, dim=1).cpu().numpy()</code>	Atribuição: define/atualiza variável ou objeto.
93	<code>all_probs.extend(probs)</code>	Instrução do script no contexto do bloco.
94	<code>all_preds.extend(predicted.cpu().numpy())</code>	Instrução do script no contexto do bloco.
95	<code>all_labels.extend(labels.cpu().numpy())</code>	Instrução do script no contexto do bloco.
96		Linha em branco; separa blocos lógicos.
97	<code># Acurácia final</code>	Comentário; documenta a intenção do bloco (não executa).
98	<code>acc = 100 * correct / total</code>	Atribuição: define/atualiza variável ou objeto.
99	<code>test accuracies.append(acc)</code>	Instrução do script no contexto do bloco.
100	<code>print(f" Acurácia no teste após época {epoch+1}: {acc:.2f}%")</code>	Imprime informação/diagnóstico no console.
101		Linha em branco; separa blocos lógicos.
102	<code># === Relatório e Matriz de Confusão ===</code>	Comentário; documenta a intenção do bloco (não executa).
103	<code>print(f"\n Acurácia final: {accuracy_score(all_labels, all_preds)*100:.2f}%")</code>	Imprime informação/diagnóstico no console.
104	<code>print(f" F1-score macro: {f1_score(all_labels, all_preds, average='macro'):.2f}")</code>	Imprime informação/diagnóstico no console.
105	<code>print(f" F1-score weighted: {f1_score(all_labels, all_preds, average='weighted'):.2f}")</code>	Imprime informação/diagnóstico no console.

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

Linha	Código	Explicação
125	<code>plt.legend(loc="lower right")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
126	<code>plt.grid(True)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
127	<code>plt.tight_layout()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
128	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\camarao\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\camarao\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)

Época 1, Loss: 0.8695076167583465
Época 2, Loss: 0.27753268331289294
Época 3, Loss: 0.08753506131470204
Época 4, Loss: 0.06704652644693851
Época 5, Loss: 0.02275913832709193
Época 6, Loss: 0.01189259933307767
Época 7, Loss: 0.009142525889910758
Época 8, Loss: 0.0041374156950041655
Época 9, Loss: 0.006109298043884337
Época 10, Loss: 0.006217700033448637
□ Acurácia no teste após época 10: 85.71%

□ Acurácia final: 85.71%
□ F1-score macro: 0.83
□ F1-score weighted: 0.86

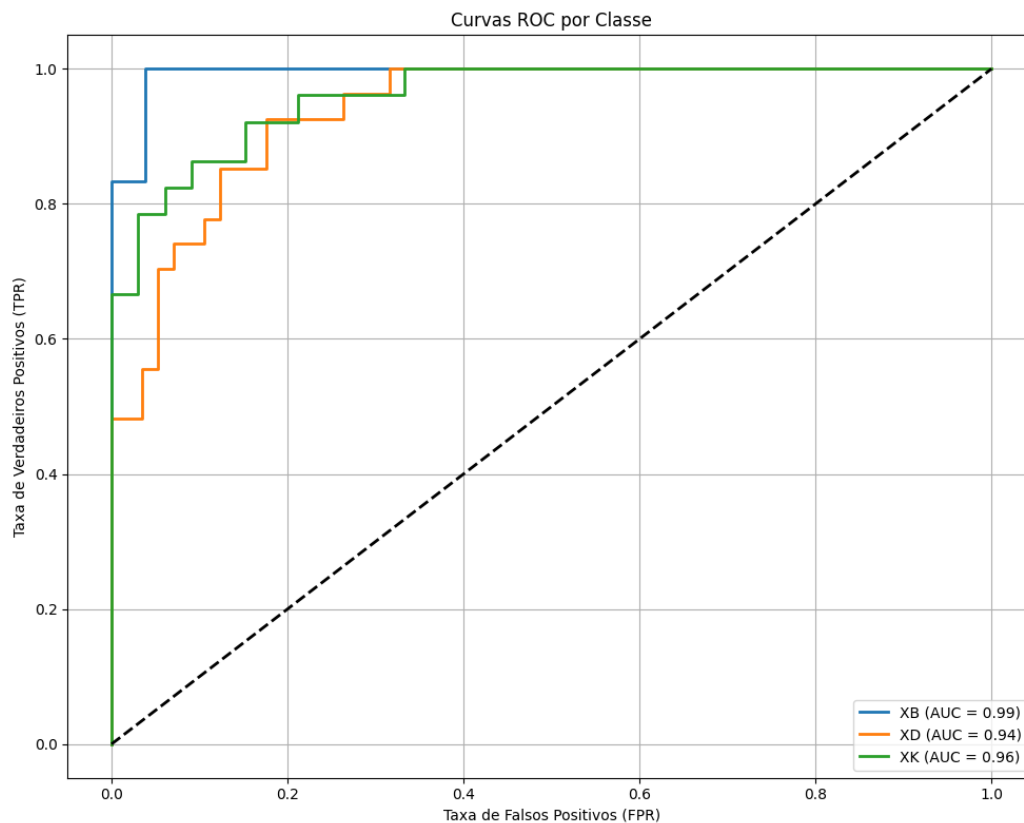
=== Classification Report ===
precision recall f1-score support

XB 1.00 0.67 0.80 6
XD 0.78 0.78 0.78 27
XK 0.89 0.92 0.90 51

accuracy 0.86 84
macro avg 0.89 0.79 0.83 84
weighted avg 0.86 0.86 0.86 84
```

Interpretação rápida: acurácia reportada ~85.71.

Figura 1. Matriz de confusão.

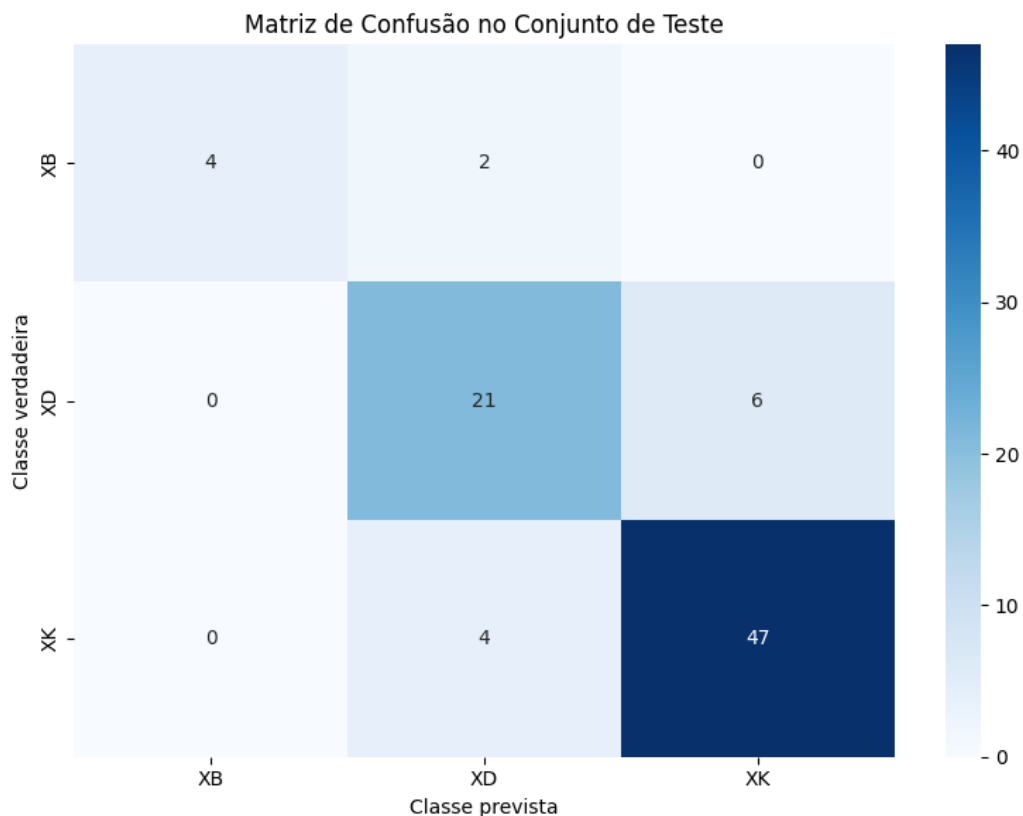


Diagonal = acertos; fora da diagonal = confusões. As maiores confusões indicam classes visualmente similares ou necessidade de mais dados/curadoria.

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

Resultados gerados nesta célula

Figura 2. Matriz de confusão.



Diagonal = acertos; fora da diagonal = confusões. As maiores confusões indicam classes visualmente similares ou necessidade de mais dados/curadoria.

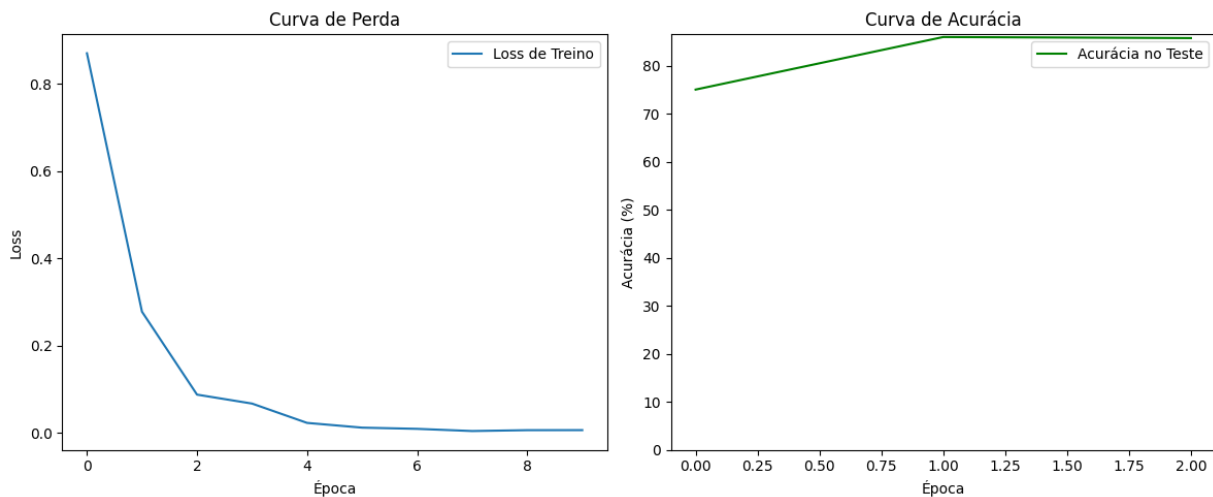
Linha	Código	Explicação
1	<code>torch.save(model.state_dict(), './models_gray/modelo_final_res_net.pth')</code>	Instrução do script no contexto do bloco.

Linha	Código	Explicação
1	<code># Plotar curva de perda</code>	Comentário; documenta a intenção do bloco (não executa).
2	<code>plt.figure(figsize=(12, 5))</code>	Comando de plotagem/ajuste de figura (Matplotlib).
3	<code># plt.title("Evolução da Perda e Acurácia Durante o Treinamento")</code>	Comentário; documenta a intenção do bloco (não executa).
4		Linha em branco; separa blocos lógicos.
5	<code>plt.subplot(1, 2, 1)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
6	<code>plt.plot(train_losses, label='Loss de Treino')</code>	Define função de perda (critério de otimização).
7	<code>plt.xlabel('Época')</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Linha	Código	Explicação
8	<code>plt.ylabel('Loss')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
9	<code>plt.title('Curva de Perda')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
10	<code>plt.legend()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
11		Linha em branco; separa blocos lógicos.
12	<code># Plotar curva de acurácia</code>	Comentário; documenta a intenção do bloco (não executa).
13	<code>plt.subplot(1, 2, 2)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
14	<code>plt.plot(test_accuracies, label='Acurácia no Teste', color='green')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
15	<code>plt.xlabel('Época')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
16	<code>plt.ylabel('Acurácia (%)')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
17	<code>plt.title('Curva de Acurácia')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
18	<code>plt.legend()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
19	<code>plt.ylim(bottom=0)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
20		Linha em branco; separa blocos lógicos.
21	<code>plt.tight_layout()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
22	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Figura 3. Curvas de loss e acurácia.



Use para verificar convergência e overfitting: se a loss de treino cai, mas a de validação/teste estagna ou piora, ajuste regularização, dados e hiperparâmetros.

4.4 AlexNet Treinamento

Linha	Código	Explicação
1	<code>train_losses = []</code>	Atribuição: define/atualiza variável ou objeto.
2	<code>test accuracies = []</code>	Atribuição: define/atualiza variável ou objeto.
3		Linha em branco; separa blocos lógicos.
4	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
5	<code># 1. CONFIGURAÇÕES</code>	Comentário; documenta a intenção do bloco (não executa).
6	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
7	<code>device = torch.device("cuda" if torch.cuda.is_available() else "cpu")</code>	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
8	<code>batch_size = 32</code>	Atribuição: define/atualiza variável ou objeto.
9	<code>num_epochs = 10</code>	Atribuição: define/atualiza variável ou objeto.
10	<code>data_dir = "cameroes_dataset_split" # Caminho para dataset com train/ e test/</code>	Atribuição: define/atualiza variável ou objeto.
11		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
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([Atribuição: define/atualiza variável ou objeto.
16	transforms.Resize((224, 224)), # Todas as redes esperam 224x224	Define transformação/pré-processamento de imagens (torchvision.transforms).
17	transforms.ToTensor(),	Define transformação/pré-processamento de imagens (torchvision.transforms).
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)	Atribuição: define/atualiza variável ou objeto.
27	test_dataset = datasets.ImageFolder(os.path.join(data_dir, "test"), transform=transform)	Atribuição: define/atualiza variável ou objeto.
28		Linha em branco; separa blocos lógicos.
29	train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True, generator=generator)	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
30	<code>test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False, generator=generator)</code>	Atribuição: define/atualiza variável ou objeto.
31		Linha em branco; separa blocos lógicos.
32	<code>num_classes = len(train_dataset.classes)</code>	Atribuição: define/atualiza variável ou objeto.
33	<code>class_names = train_dataset.classes</code>	Atribuição: define/atualiza variável ou objeto.
34		Linha em branco; separa blocos lógicos.
35	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
36	<code># 4. MODELO</code>	Comentário; documenta a intenção do bloco (não executa).
37	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
38		Linha em branco; separa blocos lógicos.
39	<code>model = models.alexnet(pretrained=True)</code>	Atribuição: define/atualiza variável ou objeto.
40	<code>model.classifier[6] = nn.Linear(model.classifier[6].in_features, num_classes)</code>	Atribuição: define/atualiza variável ou objeto.
41		Linha em branco; separa blocos lógicos.
42	<code>model = model.to(device)</code>	Atribuição: define/atualiza variável ou objeto.
43		Linha em branco; separa blocos lógicos.
44	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
45	<code># 5. LOSS E OTIMIZADOR</code>	Comentário; documenta a intenção do bloco (não executa).
46	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
47	<code>criterion = nn.CrossEntropyLoss()</code>	Atribuição: define/atualiza variável ou objeto.
48	<code>optimizer = optim.Adam(model.parameters(), lr=0.0001)</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
49		Linha em branco; separa blocos lógicos.
50	# =====	Comentário; documenta a intenção do bloco (não executa).
51	# 6. TREINAMENTO	Comentário; documenta a intenção do bloco (não executa).
52	# =====	Comentário; documenta a intenção do bloco (não executa).
53	for epoch in range(num_epochs):	Inicia um laço de iteração.
54	model.train()	Coloca o modelo em modo de treino.
55	running_loss = 0.0	Atribuição: define/atualiza variável ou objeto.
56		Linha em branco; separa blocos lógicos.
57	for images, labels in train_loader:	Inicia um laço de iteração.
58	images, labels = images.to(device), labels.to(device)	Atribuição: define/atualiza variável ou objeto.
59		Linha em branco; separa blocos lógicos.
60	optimizer.zero_grad()	Configura o otimizador (atualiza pesos do modelo).
61	outputs = model(images)	Atribuição: define/atualiza variável ou objeto.
62	loss = criterion(outputs, labels)	Atribuição: define/atualiza variável ou objeto.
63	loss.backward()	Define função de perda (critério de otimização).
64	optimizer.step()	Configura o otimizador (atualiza pesos do modelo).
65		Linha em branco; separa blocos lógicos.
66	running_loss += loss.item()	Atribuição: define/atualiza variável ou objeto.
67		Linha em branco; separa blocos lógicos.
68	print(f"Época {epoch+1}, Loss: {running_loss / len(train_loader)}")	Imprime informação/diagnóstico no console.

Linha	Código	Explicação
69		Linha em branco; separa blocos lógicos.
70	<code>train_losses.append(running_loss / len(train_loader))</code>	Define função de perda (critério de otimização).
71		Linha em branco; separa blocos lógicos.
72	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
73	<code># 7. AVALIAÇÃO NO TESTE</code>	Comentário; documenta a intenção do bloco (não executa).
74	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
75	<code>model.eval()</code>	Coloca o modelo em modo de avaliação (desativa dropout etc.).
76	<code>correct = 0</code>	Atribuição: define/atualiza variável ou objeto.
77	<code>total = 0</code>	Atribuição: define/atualiza variável ou objeto.
78		Linha em branco; separa blocos lógicos.
79	<code>all_preds = []</code>	Atribuição: define/atualiza variável ou objeto.
80	<code>all_labels = []</code>	Atribuição: define/atualiza variável ou objeto.
81	<code>all_probs = []</code>	Atribuição: define/atualiza variável ou objeto.
82	<code>with torch.no_grad():</code>	Instrução do script no contexto do bloco.
83	<code>for images, labels in test_loader:</code>	Inicia um laço de iteração.
84	<code>images, labels = images.to(device), labels.to(device)</code>	Atribuição: define/atualiza variável ou objeto.
85	<code>outputs = model(images)</code>	Atribuição: define/atualiza variável ou objeto.
86	<code>_, predicted = torch.max(outputs, 1)</code>	Atribuição: define/atualiza variável ou objeto.
87		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
88	<code>total += labels.size(0)</code>	Atribuição: define/atualiza variável ou objeto.
89	<code>correct += (predicted == labels).sum().item()</code>	Instrução do script no contexto do bloco.
90		Linha em branco; separa blocos lógicos.
91	<code># Armazena previsões e rótulos verdadeiros</code>	Comentário; documenta a intenção do bloco (não executa).
92	<code>probs = torch.softmax(outputs, dim=1).cpu().numpy()</code>	Atribuição: define/atualiza variável ou objeto.
93	<code>all_probs.extend(probs)</code>	Instrução do script no contexto do bloco.
94	<code>all_preds.extend(predicted.cpu().numpy())</code>	Instrução do script no contexto do bloco.
95	<code>all_labels.extend(labels.cpu().numpy())</code>	Instrução do script no contexto do bloco.
96		Linha em branco; separa blocos lógicos.
97	<code># Acurácia final</code>	Comentário; documenta a intenção do bloco (não executa).
98	<code>acc = 100 * correct / total</code>	Atribuição: define/atualiza variável ou objeto.
99	<code>test_accuracies.append(acc)</code>	Instrução do script no contexto do bloco.
100	<code>print(f" Acurácia no teste após época {epoch+1}: {acc:.2f}%")</code>	Imprime informação/diagnóstico no console.
101		Linha em branco; separa blocos lógicos.
102	<code># === Relatório e Matriz de Confusão ===</code>	Comentário; documenta a intenção do bloco (não executa).
103		Linha em branco; separa blocos lógicos.
104	<code>print(f"\n Acurácia final: {accuracy_score(all_labels, all_preds)*100:.2f}%")</code>	Imprime informação/diagnóstico no console.
105	<code>print(f" F1-score macro: {f1_score(all_labels, all_preds, average='macro'):.2f}%")</code>	Imprime informação/diagnóstico no console.
106	<code>print(f" F1-score weighted: {f1_score(all_labels, all_preds, average='weighted'):.2f}%")</code>	Imprime informação/diagnóstico no console.

Linha	Código	Explicação
107		Linha em branco; separa blocos lógicos.
108	<code>print("\n=== Classification Report ===")</code>	Imprime informação/diagnóstico no console.
109	<code>print(classification_report(all_labels, all_preds, target_names=class_names))</code>	Imprime informação/diagnóstico no console.
110		Linha em branco; separa blocos lógicos.
111	<code>cm = confusion_matrix(all_labels, all_preds)</code>	Atribuição: define/atualiza variável ou objeto.
112	<code>plt.figure(figsize=(8, 6))</code>	Comando de plotagem/ajuste de figura (Matplotlib).
113	<code>sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)</code>	Instrução do script no contexto do bloco.
114	<code>plt.xlabel("Classe prevista")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
115	<code>plt.ylabel("Classe verdadeira")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
116	<code>plt.title("Matriz de Confusão no Conjunto de Teste")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
117	<code>plt.tight_layout()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
118	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
119		Linha em branco; separa blocos lógicos.
120		Linha em branco; separa blocos lógicos.
121	<code># One-hot encoding dos rótulos verdadeiros</code>	Comentário; documenta a intenção do bloco (não executa).
122	<code>y_true = label_binarize(all_labels, classes=list(range(num_classes)))</code>	Atribuição: define/atualiza variável ou objeto.
123	<code>y_score = np.array(all_probs)</code>	Atribuição: define/atualiza variável ou objeto.
124		Linha em branco; separa blocos lógicos.
125	<code># Plotar curva ROC para cada classe</code>	Comentário; documenta a intenção do bloco (não executa).

Linha	Código	Explicação
126	<code>plt.figure(figsize=(10, 8))</code>	Comando de plotagem/ajuste de figura (Matplotlib).
127	<code>for i in range(num_classes):</code>	Inicia um laço de iteração.
128	<code> fpr, tpr, _ = roc_curve(y_true[:, i], y_score[:, i])</code>	Atribuição: define/atualiza variável ou objeto.
129	<code> roc_auc = auc(fpr, tpr)</code>	Atribuição: define/atualiza variável ou objeto.
130	<code> plt.plot(fpr, tpr, lw=2, label=f'{class_names[i]} (AUC = {roc_auc:.2f})')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
131		Linha em branco; separa blocos lógicos.
132	<code>plt.plot([0, 1], [0, 1], 'k--', lw=2)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
133	<code>plt.xlabel('Taxa de Falsos Positivos (FPR)')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
134	<code>plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
135	<code>plt.title('Curvas ROC por Classe')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
136	<code>plt.legend(loc="lower right")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
137	<code>plt.grid(True)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
138	<code>plt.tight_layout()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
139	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

```

c:\Users\vitin\Documents\LABEEC\camarao\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\camarao\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)

Época 1, Loss: 0.8730146527290344
Época 2, Loss: 0.6231968343257904
Época 3, Loss: 0.4675635129213333
Época 4, Loss: 0.31637365445494653
Época 5, Loss: 0.12769808182492853
Época 6, Loss: 0.07347845956683159
Época 7, Loss: 0.07365419734269381
Época 8, Loss: 0.04454803564585745
Época 9, Loss: 0.021316711558029056
Época 10, Loss: 0.011329704523086548
☐ Acurácia no teste após época 10: 79.76%

☐ Acurácia final: 79.76%
☐ F1-score macro: 0.78
☐ F1-score weighted: 0.79

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

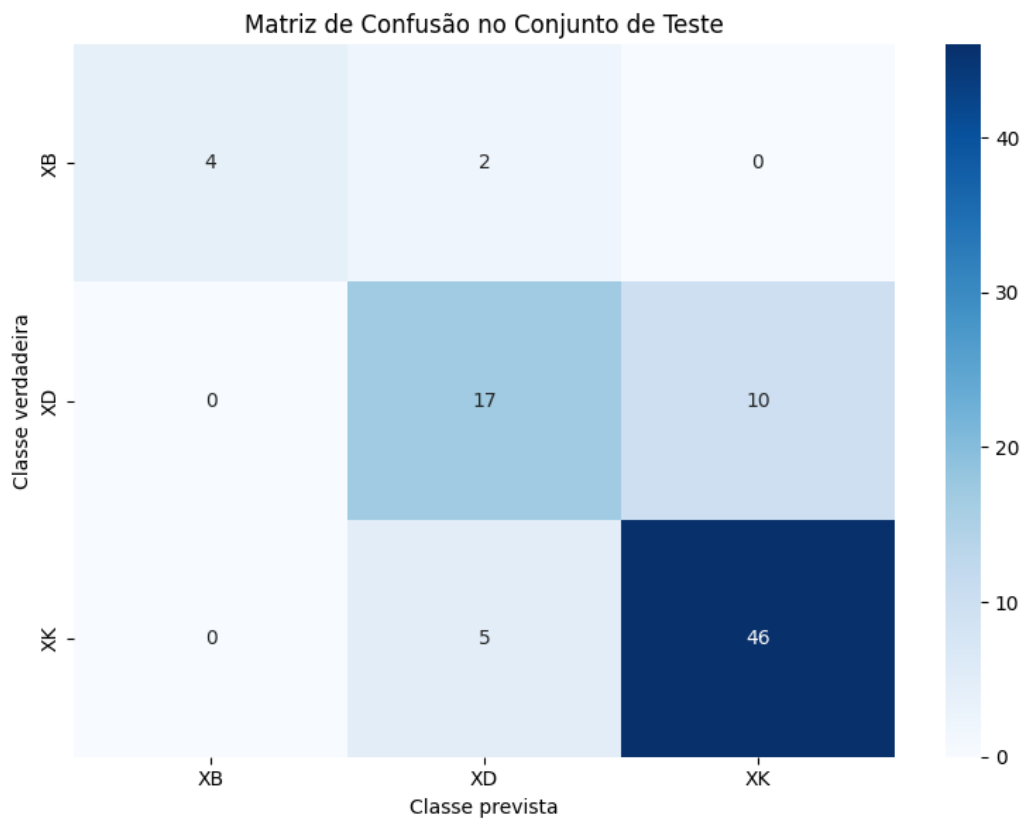
XB 1.00 0.67 0.80 6
XD 0.71 0.63 0.67 27
XK 0.82 0.90 0.86 51

accuracy 0.80 84
macro avg 0.84 0.73 0.78 84
weighted avg 0.80 0.80 0.79 84

```

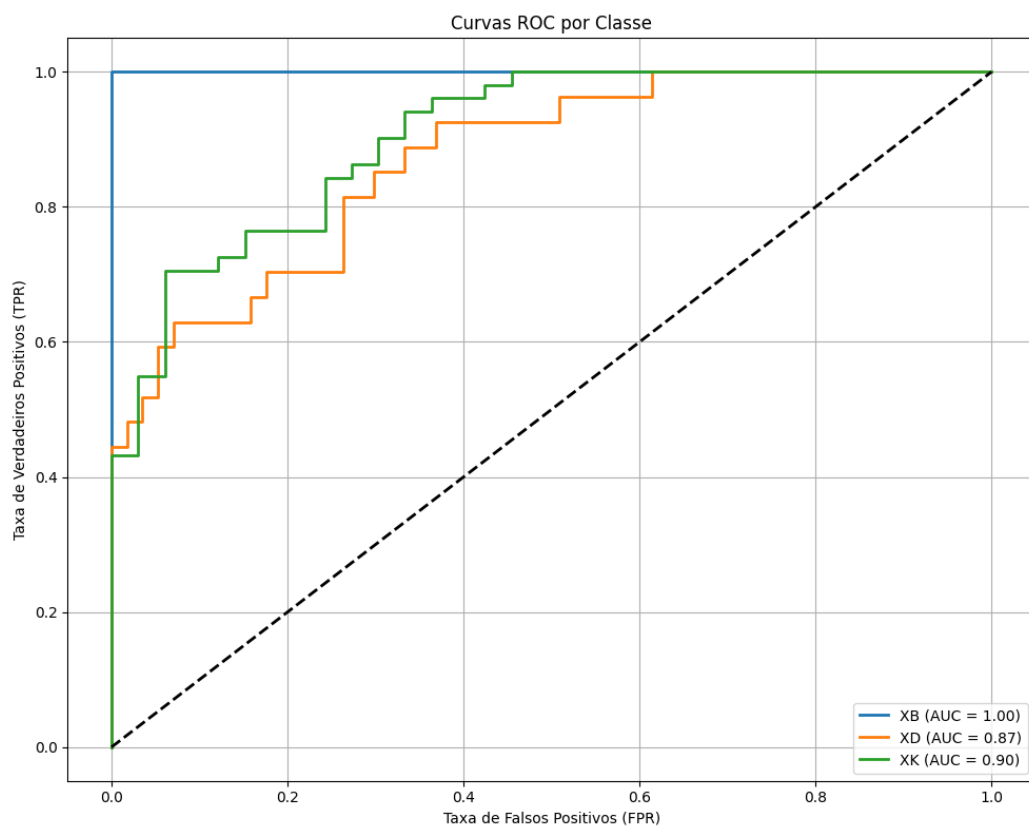
Interpretação rápida: acurácia reportada ~79.76.

Figura 4. Matriz de confusão.



Diagonal = acertos; fora da diagonal = confusões. As maiores confusões indicam classes visualmente similares ou necessidade de mais dados/curadoria.

Figura 5. Matriz de confusão.



Diagonal = acertos; fora da diagonal = confusões. As maiores confusões indicam classes visualmente similares ou necessidade de mais dados/curadoria.

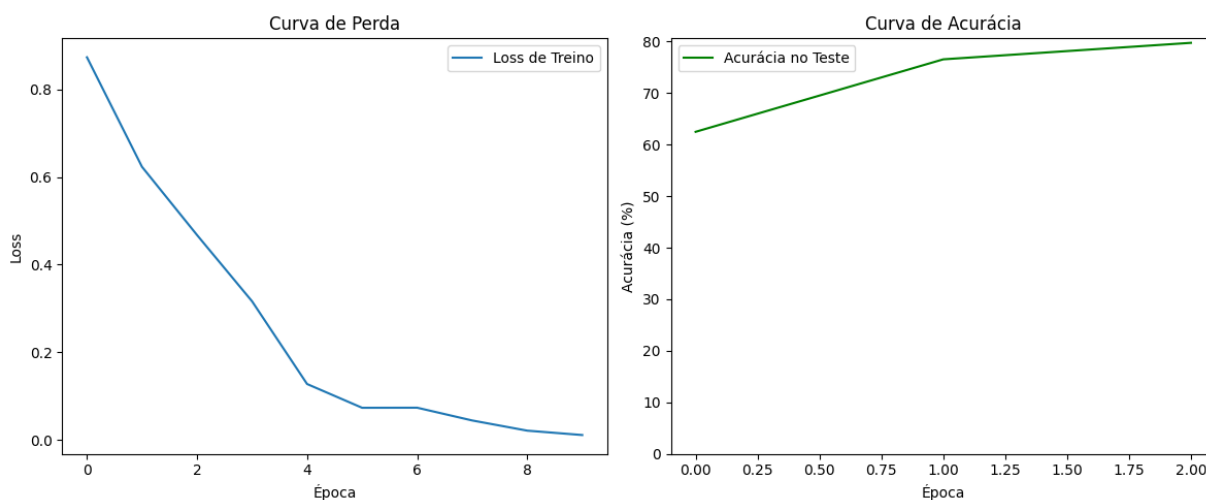
Linha	Código	Explicação
1	<code>torch.save(model.state_dict(), './models_gray/modelo_final_alex_net.pth')</code>	Instrução do script no contexto do bloco.

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

Linha	Código	Explicação
18	<code>plt.legend()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
19	<code>plt.ylim(bottom=0)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
20		Linha em branco; separa blocos lógicos.
21	<code>plt.tight_layout()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
22	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Figura 6. Curvas de loss e acurácia.



Use para verificar convergência e overfitting: se a loss de treino cai, mas a de validação/teste estagna ou piora, ajuste regularização, dados e hiperparâmetros.

4.5 MobileNetV2 Treinamento

Linha	Código	Explicação
1	<code>train_losses = []</code>	Atribuição: define/atualiza variável ou objeto.
2	<code>test_accuracies = []</code>	Atribuição: define/atualiza variável ou objeto.
3		Linha em branco; separa blocos lógicos.
4	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).

Linha	Código	Explicação
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 = "camaroetes_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([Atribuição: define/atualiza variável ou objeto.
16	transforms.Resize((224, 224)), # Todas as redes esperam 224x224	Define transformação/pré-processamento de imagens (torchvision.transforms).
17	transforms.ToTensor(),	Define transformação/pré-processamento de imagens (torchvision.transforms).
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).

Linha	Código	Explicação
23	# 3. DATALOADERS	Comentário; documenta a intenção do bloco (não executa).
24	# =====	Comentário; documenta a intenção do bloco (não executa).
25	train_dataset = datasets.ImageFolder(os.path.join(data_dir, "train"), transform=transform)	Atribuição: define/atualiza variável ou objeto.
26	test_dataset = datasets.ImageFolder(os.path.join(data_dir, "test"), transform=transform)	Atribuição: define/atualiza variável ou objeto.
27		Linha em branco; separa blocos lógicos.
28	train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)	Atribuição: define/atualiza variável ou objeto.
29	test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)	Atribuição: define/atualiza variável ou objeto.
30		Linha em branco; separa blocos lógicos.
31	num_classes = len(train_dataset.classes)	Atribuição: define/atualiza variável ou objeto.
32		Linha em branco; separa blocos lógicos.
33	# =====	Comentário; documenta a intenção do bloco (não executa).
34	# 4. MODELO	Comentário; documenta a intenção do bloco (não executa).
35	# =====	Comentário; documenta a intenção do bloco (não executa).
36		Linha em branco; separa blocos lógicos.
37	model = models.mobilenet_v2(pretrained=True)	Atribuição: define/atualiza variável ou objeto.
38	model.classifier[1] = nn.Linear(model.classifier[1].in_features, num_classes)	Atribuição: define/atualiza variável ou objeto.
39		Linha em branco; separa blocos lógicos.
40	model = model.to(device)	Atribuição: define/atualiza variável ou objeto.
41		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
42	# =====	Comentário; documenta a intenção do bloco (não executa).
43	# 5. LOSS E OTIMIZADOR	Comentário; documenta a intenção do bloco (não executa).
44	# =====	Comentário; documenta a intenção do bloco (não executa).
45	criterion = nn.CrossEntropyLoss()	Atribuição: define/atualiza variável ou objeto.
46	optimizer = optim.Adam(model.parameters(), lr=0.0001)	Atribuição: define/atualiza variável ou objeto.
47		Linha em branco; separa blocos lógicos.
48	# =====	Comentário; documenta a intenção do bloco (não executa).
49	# 6. TREINAMENTO	Comentário; documenta a intenção do bloco (não executa).
50	# =====	Comentário; documenta a intenção do bloco (não executa).
51	for epoch in range(num_epochs):	Inicia um laço de iteração.
52	model.train()	Coloca o modelo em modo de treino.
53	running_loss = 0.0	Atribuição: define/atualiza variável ou objeto.
54		Linha em branco; separa blocos lógicos.
55	for images, labels in train_loader:	Inicia um laço de iteração.
56	images, labels = images.to(device), labels.to(device)	Atribuição: define/atualiza variável ou objeto.
57		Linha em branco; separa blocos lógicos.
58	optimizer.zero_grad()	Configura o otimizador (atualiza pesos do modelo).
59	outputs = model(images)	Atribuição: define/atualiza variável ou objeto.
60	loss = criterion(outputs, labels)	Atribuição: define/atualiza variável ou objeto.
61	loss.backward()	Define função de perda (critério de otimização).

Linha	Código	Explicação
62	<code>optimizer.step()</code>	Configura o otimizador (atualiza pesos do modelo).
63		Linha em branco; separa blocos lógicos.
64	<code>running_loss += loss.item()</code>	Atribuição: define/atualiza variável ou objeto.
65		Linha em branco; separa blocos lógicos.
66	<code>print(f"Época {epoch+1}, Loss: {running_loss / len(train_loader)}")</code>	Imprime informação/diagnóstico no console.
67		Linha em branco; separa blocos lógicos.
68	<code>train_losses.append(running_loss / len(train_loader))</code>	Define função de perda (critério de otimização).
69		Linha em branco; separa blocos lógicos.
70	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
71	<code># 7. AVALIAÇÃO NO TESTE</code>	Comentário; documenta a intenção do bloco (não executa).
72	<code># =====</code>	Comentário; documenta a intenção do bloco (não executa).
73	<code>model.eval()</code>	Coloca o modelo em modo de avaliação (desativa dropout etc.).
74	<code>correct = 0</code>	Atribuição: define/atualiza variável ou objeto.
75	<code>total = 0</code>	Atribuição: define/atualiza variável ou objeto.
76		Linha em branco; separa blocos lógicos.
77	<code>all_preds = []</code>	Atribuição: define/atualiza variável ou objeto.
78	<code>all_labels = []</code>	Atribuição: define/atualiza variável ou objeto.
79	<code>all_probs = []</code>	Atribuição: define/atualiza variável ou objeto.
80	<code>with torch.no_grad():</code>	Instrução do script no contexto do bloco.

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

Linha	Código	Explicação
100	# === Relatório e Matriz de Confusão ===	Comentário; documenta a intenção do bloco (não executa).
101	class_names = train_dataset.classes	Atribuição: define/atualiza variável ou objeto.
102		Linha em branco; separa blocos lógicos.
103	print(f"\n Acurácia final: {accuracy_score(all_labels, all_preds)*100:.2f}%")	Imprime informação/diagnóstico no console.
104	print(f" F1-score macro: {f1_score(all_labels, all_preds, average='macro'):.2f}")	Imprime informação/diagnóstico no console.
105	print(f" F1-score weighted: {f1_score(all_labels, all_preds, average='weighted'):.2f}")	Imprime informação/diagnóstico no console.
106		Linha em branco; separa blocos lógicos.
107	print("\n=== Classification Report ===")	Imprime informação/diagnóstico no console.
108	print(classification_report(all_labels, all_preds, target_names=class_names))	Imprime informação/diagnóstico no console.
109		Linha em branco; separa blocos lógicos.
110	cm = confusion_matrix(all_labels, all_preds)	Atribuição: define/atualiza variável ou objeto.
111	plt.figure(figsize=(8, 6))	Comando de plotagem/ajuste de figura (Matplotlib).
112	sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)	Instrução do script no contexto do bloco.
113	plt.xlabel("Classe prevista")	Comando de plotagem/ajuste de figura (Matplotlib).
114	plt.ylabel("Classe verdadeira")	Comando de plotagem/ajuste de figura (Matplotlib).
115	plt.title("Matriz de Confusão no Conjunto de Teste")	Comando de plotagem/ajuste de figura (Matplotlib).
116	plt.tight_layout()	Comando de plotagem/ajuste de figura (Matplotlib).
117	plt.show()	Comando de plotagem/ajuste de figura (Matplotlib).
118		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
119	<code># One-hot encoding dos rótulos verdadeiros</code>	Comentário; documenta a intenção do bloco (não executa).
120	<code>y_true = label_binarize(all_labels, classes=list(range(num_classes)))</code>	Atribuição: define/atualiza variável ou objeto.
121	<code>y_score = np.array(all_probs)</code>	Atribuição: define/atualiza variável ou objeto.
122		Linha em branco; separa blocos lógicos.
123	<code># Plotar curva ROC para cada classe</code>	Comentário; documenta a intenção do bloco (não executa).
124	<code>plt.figure(figsize=(10, 8))</code>	Comando de plotagem/ajuste de figura (Matplotlib).
125	<code>for i in range(num_classes):</code>	Inicia um laço de iteração.
126	<code> fpr, tpr, _ = roc_curve(y_true[:, i], y_score[:, i])</code>	Atribuição: define/atualiza variável ou objeto.
127	<code> roc_auc = auc(fpr, tpr)</code>	Atribuição: define/atualiza variável ou objeto.
128	<code> plt.plot(fpr, tpr, lw=2, label=f'{class_names[i]} (AUC = {roc_auc:.2f})')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
129		Linha em branco; separa blocos lógicos.
130	<code>plt.plot([0, 1], [0, 1], 'k--', lw=2)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
131	<code>plt.xlabel('Taxa de Falsos Positivos (FPR)')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
132	<code>plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
133	<code>plt.title('Curvas ROC por Classe')</code>	Comando de plotagem/ajuste de figura (Matplotlib).
134	<code>plt.legend(loc="lower right")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
135	<code>plt.grid(True)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
136	<code>plt.tight_layout()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
137	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\camarao\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\camarao\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)

Época 1, Loss: 0.8536086678504944
Época 2, Loss: 0.44405471831560134
Época 3, Loss: 0.24857145100831984
Época 4, Loss: 0.08011382780969142
Época 5, Loss: 0.08074797205626964
Época 6, Loss: 0.02686897963285446
Época 7, Loss: 0.06776842176914215
Época 8, Loss: 0.046061249915510416
Época 9, Loss: 0.031345964223146436
Época 10, Loss: 0.06150011131539941
☐ Acurácia no teste após época 10: 86.90%

☐ Acurácia final: 86.90%
☐ F1-score macro: 0.87
☐ F1-score weighted: 0.87

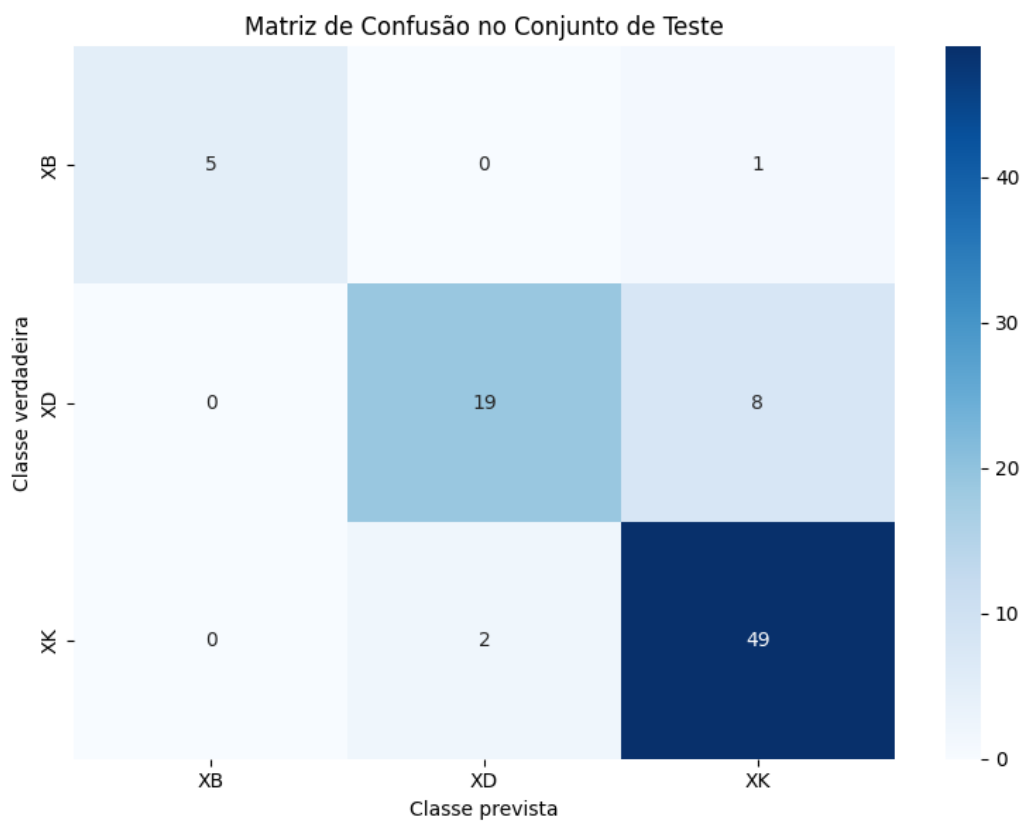
=== Classification Report ===
precision recall f1-score support

XB 1.00 0.83 0.91 6
XD 0.90 0.70 0.79 27
XK 0.84 0.96 0.90 51

accuracy 0.87 84
macro avg 0.92 0.83 0.87 84
weighted avg 0.88 0.87 0.87 84
```

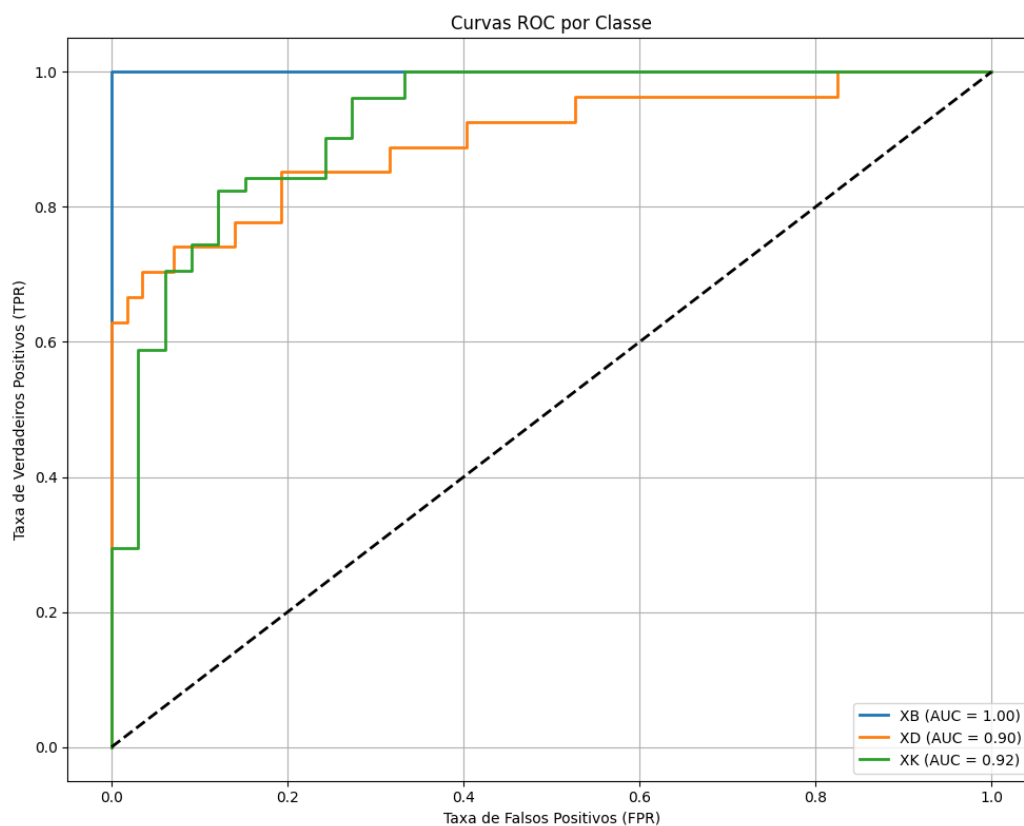
Interpretação rápida: acurácia reportada ~86.90.

Figura 7. Matriz de confusão.



Diagonal = acertos; fora da diagonal = confusões. As maiores confusões indicam classes visualmente similares ou necessidade de mais dados/curadoria.

Figura 8. Matriz de confusão.



Diagonal = acertos; fora da diagonal = confusões. As maiores confusões indicam classes visualmente similares ou necessidade de mais dados/curadoria.

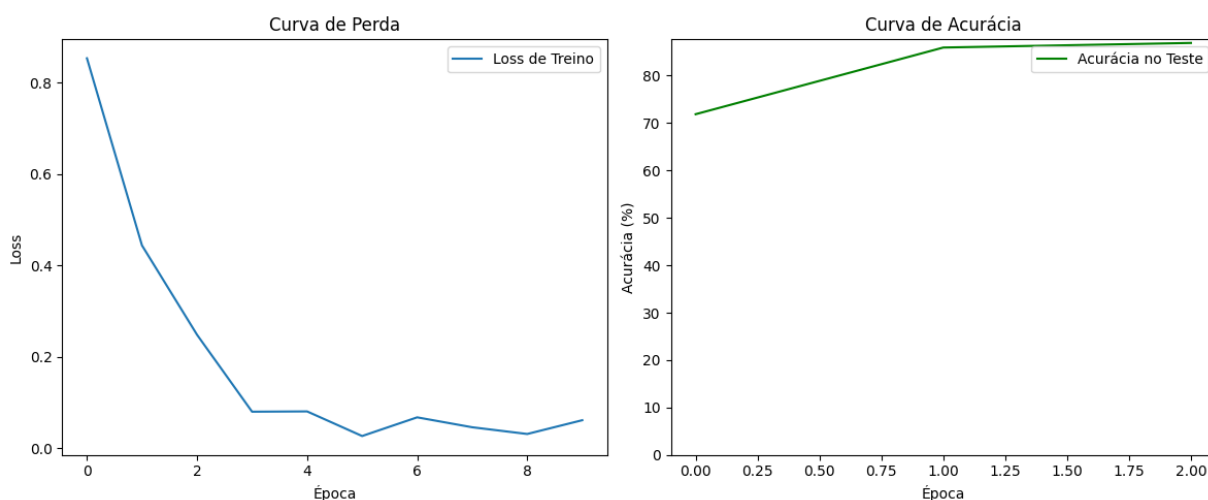
Linha	Código	Explicação
1	<code>torch.save(model.state_dict(), './models_gray/modelo_final_mobile_net.pth')</code>	Instrução do script no contexto do bloco.

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

Linha	Código	Explicação
18	<code>plt.ylim(bottom=0)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
19		Linha em branco; separa blocos lógicos.
20	<code>plt.tight_layout()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
21	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Figura 9. Curvas de loss e acurácia.



Use para verificar convergência e overfitting: se a loss de treino cai, mas a de validação/teste estagna ou piora, ajuste regularização, dados e hiperparâmetros.

4.6 ResNet avaliação

Linha	Código	Explicação
1	<code>import torch</code>	Importa bibliotecas/módulos necessários.
2	<code>import os</code>	Importa bibliotecas/módulos necessários.
3	<code>from PIL import Image</code>	Importa bibliotecas/módulos necessários.
4	<code>from torchvision import transforms, models</code>	Importa bibliotecas/módulos necessários.
5	<code>from sklearn.metrics import classification_report, confusion_matrix</code>	Importa bibliotecas/módulos necessários.

Linha	Código	Explicação
6	<code>import numpy as np</code>	Importa bibliotecas/módulos necessários.
7		Linha em branco; separa blocos lógicos.
8	<code># === CONFIGURAÇÕES ===</code>	Comentário; documenta a intenção do bloco (não executa).
9	<code>caminho_dataset = 'camaroes_dataset_split/external'</code>	Atribuição: define/atualiza variável ou objeto.
10	<code>classes = sorted(os.listdir(caminho_dataset)) # os nomes das pastas são os nomes das classes</code>	Atribuição: define/atualiza variável ou objeto.
11	<code>class_to_idx = {classe: idx for idx, classe in enumerate(classes)}</code>	Atribuição: define/atualiza variável ou objeto.
12	<code>caminho_modelo = './models_gray/modelo_final_res_net.pth' # Caminho para o modelo treinado</code>	Atribuição: define/atualiza variável ou objeto.
13		Linha em branco; separa blocos lógicos.
14	<code># === TRANSFORMAÇÕES ===</code>	Comentário; documenta a intenção do bloco (não executa).
15	<code>transform = transforms.Compose([</code>	Atribuição: define/atualiza variável ou objeto.
16	<code> transforms.Resize((224, 224)),</code>	Define transformação/pré-processamento de imagens (torchvision.transforms).
17	<code> transforms.ToTensor(),</code>	Define transformação/pré-processamento de imagens (torchvision.transforms).
18	<code> transforms.Normalize(mean=[0.485, 0.456, 0.496],</code>	Atribuição: define/atualiza variável ou objeto.
19	<code> std=[0.229, 0.224, 0.225]))</code>	Atribuição: define/atualiza variável ou objeto.
20	<code>])</code>	Instrução do script no contexto do bloco.
21		Linha em branco; separa blocos lógicos.
22	<code># === CARREGAR MODELO ===</code>	Comentário; documenta a intenção do bloco (não executa).
23	<code>model = models.resnet18(pretrained=False)</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
24	<code>model.fc = torch.nn.Linear(model.fc.in_features, len(classes))</code>	Atribuição: define/atualiza variável ou objeto.
25	<code>model.load_state_dict(torch.load(caminho_modelo, map_location='cpu'))</code>	Atribuição: define/atualiza variável ou objeto.
26	<code>model.eval()</code>	Coloca o modelo em modo de avaliação (desativa dropout etc.).
27		Linha em branco; separa blocos lógicos.
28	<code># === AVALIAÇÃO ===</code>	Comentário; documenta a intenção do bloco (não executa).
29	<code>y_true = []</code>	Atribuição: define/atualiza variável ou objeto.
30	<code>y_pred = []</code>	Atribuição: define/atualiza variável ou objeto.
31		Linha em branco; separa blocos lógicos.
32	<code>for classe in classes:</code>	Inicia um laço de iteração.
33	<code> pasta_classe = os.path.join(caminho_dataset, classe)</code>	Atribuição: define/atualiza variável ou objeto.
34	<code> for img_nome in os.listdir(pasta_classe):</code>	Inicia um laço de iteração.
35	<code> if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):</code>	Estrutura condicional (controle de fluxo).
36	<code> continue</code>	Instrução do script no contexto do bloco.
37	<code> img_path = os.path.join(pasta_classe, img_nome)</code>	Atribuição: define/atualiza variável ou objeto.
38	<code> img = Image.open(img_path).convert('RGB')</code>	Atribuição: define/atualiza variável ou objeto.
39	<code> img_tensor = transform(img).unsqueeze(0)</code>	Atribuição: define/atualiza variável ou objeto.
40		Linha em branco; separa blocos lógicos.
41	<code> with torch.no_grad():</code>	Instrução do script no contexto do bloco.
42	<code> output = model(img_tensor)</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
43	<code>_, pred = torch.max(output, 1)</code>	Atribuição: define/atualiza variável ou objeto.
44		Linha em branco; separa blocos lógicos.
45	<code>y_true.append(class_to_idx[classe])</code>	Instrução do script no contexto do bloco.
46	<code>y_pred.append(pred.item())</code>	Instrução do script no contexto do bloco.
47		Linha em branco; separa blocos lógicos.
48	<code># === RELATÓRIO ===</code>	Comentário; documenta a intenção do bloco (não executa).
49	<code>print("=== Classification Report ===")</code>	Imprime informação/diagnóstico no console.
50	<code>print(classification_report(y_true, y_pred, target_names=classes))</code>	Imprime informação/diagnóstico no console.
51		Linha em branco; separa blocos lógicos.
52	<code>print("=== Matriz de Confusão ===")</code>	Imprime informação/diagnóstico no console.
53	<code>cm = confusion_matrix(y_true, y_pred)</code>	Atribuição: define/atualiza variável ou objeto.
54	<code>plt.figure(figsize=(8, 6))</code>	Comando de plotagem/ajuste de figura (Matplotlib).
55	<code>sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=classes, yticklabels=classes)</code>	Instrução do script no contexto do bloco.
56	<code>plt.xlabel("Classe prevista")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
57	<code>plt.ylabel("Classe verdadeira")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
58	<code>plt.title("□ Matriz de Confusão")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
59	<code>plt.tight_layout()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
60	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\camarao\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\camarao\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)

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

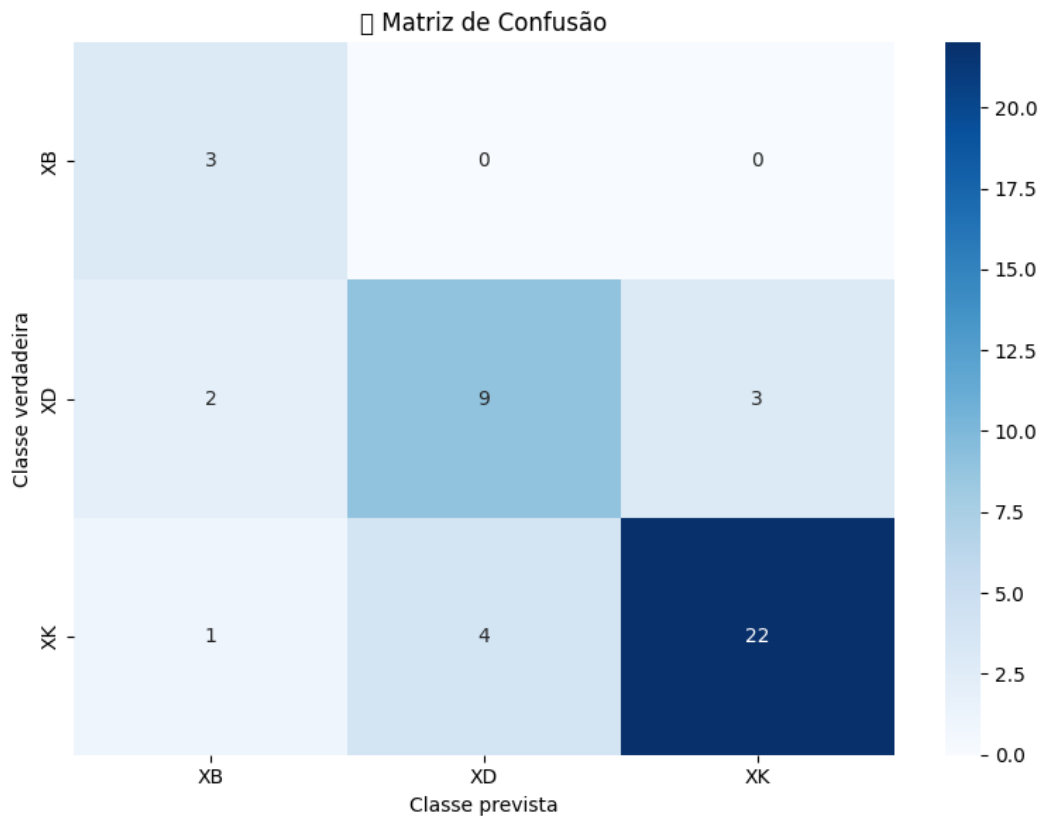
XB 0.50 1.00 0.67 3
XD 0.69 0.64 0.67 14
XK 0.88 0.81 0.85 27

accuracy 0.77 44
macro avg 0.69 0.82 0.73 44
weighted avg 0.79 0.77 0.78 44

=== Matriz de Confusão ===

C:\Users\vitin\AppData\Local\Temp\ipykernel_20368\893438013.py:59: UserWarning: Glyph
128269 (\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) DejaVu Sans.
plt.tight_layout()
c:\Users\vitin\Documents\LABEEC\camarao\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 10. Matriz de confusão.



Diagonal = acertos; fora da diagonal = confusões. As maiores confusões indicam classes visualmente similares ou necessidade de mais dados/curadoria.

4.7 Avaliação do Modelo AlexNet

Linha	Código	Explicação
1	<code>import torch</code>	Importa bibliotecas/módulos necessários.
2	<code>import os</code>	Importa bibliotecas/módulos necessários.
3	<code>from PIL import Image</code>	Importa bibliotecas/módulos necessários.
4	<code>from torchvision import transforms, models</code>	Importa bibliotecas/módulos necessários.
5	<code>from sklearn.metrics import classification_report, confusion_matrix</code>	Importa bibliotecas/módulos necessários.
6	<code>import numpy as np</code>	Importa bibliotecas/módulos necessários.
7		Linha em branco; separa blocos lógicos.
8	<code># === CONFIGURAÇÕES ===</code>	Comentário; documenta a intenção do bloco (não executa).
9	<code>caminho_dataset = 'camaroés_dataset_split/external'</code>	Atribuição: define/atualiza variável ou objeto.
10	<code>classes = sorted(os.listdir(caminho_dataset)) # os nomes das pastas são os nomes das classes</code>	Atribuição: define/atualiza variável ou objeto.
11	<code>class_to_idx = {classe: idx for idx, classe in enumerate(classes)}</code>	Atribuição: define/atualiza variável ou objeto.
12	<code>caminho_modelo = './models_gray/modelo_final_alex_net.pth' # Caminho para o modelo treinado</code>	Atribuição: define/atualiza variável ou objeto.
13		Linha em branco; separa blocos lógicos.
14	<code># === TRANSFORMAÇÕES ===</code>	Comentário; documenta a intenção do bloco (não executa).
15	<code>transform = transforms.Compose([</code>	Atribuição: define/atualiza variável ou objeto.
16	<code> transforms.Resize((224, 224)),</code>	Define transformação/pré-processamento de imagens (torchvision.transforms).
17	<code> transforms.ToTensor(),</code>	Define transformação/pré-processamento de imagens (torchvision.transforms).

Linha	Código	Explicação
18	<code>transforms.Normalize(mean=[0.485, 0.456, 0.406],</code>	Atribuição: define/atualiza variável ou objeto.
19	<code>std=[0.229, 0.224, 0.225])</code>	Atribuição: define/atualiza variável ou objeto.
20	<code>)</code>	Instrução do script no contexto do bloco.
21		Linha em branco; separa blocos lógicos.
22	<code># === CARREGAR MODELO ===</code>	Comentário; documenta a intenção do bloco (não executa).
23	<code>model = models.alexnet(pretrained=False)</code>	Atribuição: define/atualiza variável ou objeto.
24	<code>model.classifier[6] = torch.nn.Linear(model.classifier[6].in_features, len(classes))</code>	Atribuição: define/atualiza variável ou objeto.
25	<code>model.load_state_dict(torch.load(caminho_modelo, map_location='cpu'))</code>	Atribuição: define/atualiza variável ou objeto.
26	<code>model.eval()</code>	Coloca o modelo em modo de avaliação (desativa dropout etc.).
27		Linha em branco; separa blocos lógicos.
28	<code># === AVALIAÇÃO ===</code>	Comentário; documenta a intenção do bloco (não executa).
29	<code>y_true = []</code>	Atribuição: define/atualiza variável ou objeto.
30	<code>y_pred = []</code>	Atribuição: define/atualiza variável ou objeto.
31		Linha em branco; separa blocos lógicos.
32	<code>for classe in classes:</code>	Inicia um laço de iteração.
33	<code> pasta_classe = os.path.join(caminho_dataset, classe)</code>	Atribuição: define/atualiza variável ou objeto.
34	<code> for img_nome in os.listdir(pasta_classe):</code>	Inicia um laço de iteração.
35	<code> if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):</code>	Estrutura condicional (controle de fluxo).
36	<code> continue</code>	Instrução do script no contexto do bloco.

Linha	Código	Explicação
37	<code>img_path = os.path.join(pasta_classe, img_nome)</code>	Atribuição: define/atualiza variável ou objeto.
38	<code>img = Image.open(img_path).convert('RGB')</code>	Atribuição: define/atualiza variável ou objeto.
39	<code>img_tensor = transform(img).unsqueeze(0)</code>	Atribuição: define/atualiza variável ou objeto.
40		Linha em branco; separa blocos lógicos.
41	<code>with torch.no_grad():</code>	Instrução do script no contexto do bloco.
42	<code>output = model(img_tensor)</code>	Atribuição: define/atualiza variável ou objeto.
43	<code>_, pred = torch.max(output, 1)</code>	Atribuição: define/atualiza variável ou objeto.
44		Linha em branco; separa blocos lógicos.
45	<code>y_true.append(class_to_idx[classe])</code>	Instrução do script no contexto do bloco.
46	<code>y_pred.append(pred.item())</code>	Instrução do script no contexto do bloco.
47		Linha em branco; separa blocos lógicos.
48	<code># === RELATÓRIO ===</code>	Comentário; documenta a intenção do bloco (não executa).
49	<code>print("=== Classification Report ===")</code>	Imprime informação/diagnóstico no console.
50	<code>print(classification_report(y_true, y_pred, target_names=classes))</code>	Imprime informação/diagnóstico no console.
51		Linha em branco; separa blocos lógicos.
52	<code>print("=== Matriz de Confusão ===")</code>	Imprime informação/diagnóstico no console.
53	<code>cm = confusion_matrix(y_true, y_pred)</code>	Atribuição: define/atualiza variável ou objeto.
54	<code>plt.figure(figsize=(8, 6))</code>	Comando de plotagem/ajuste de figura (Matplotlib).
55	<code>sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=classes, yticklabels=classes)</code>	Instrução do script no contexto do bloco.

Linha	Código	Explicação
56	<code>plt.xlabel("Classe prevista")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
57	<code>plt.ylabel("Classe verdadeira")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
58	<code>plt.title("□ Matriz de Confusão")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
59	<code>plt.tight_layout()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
60	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\camarao\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\camarao\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)

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

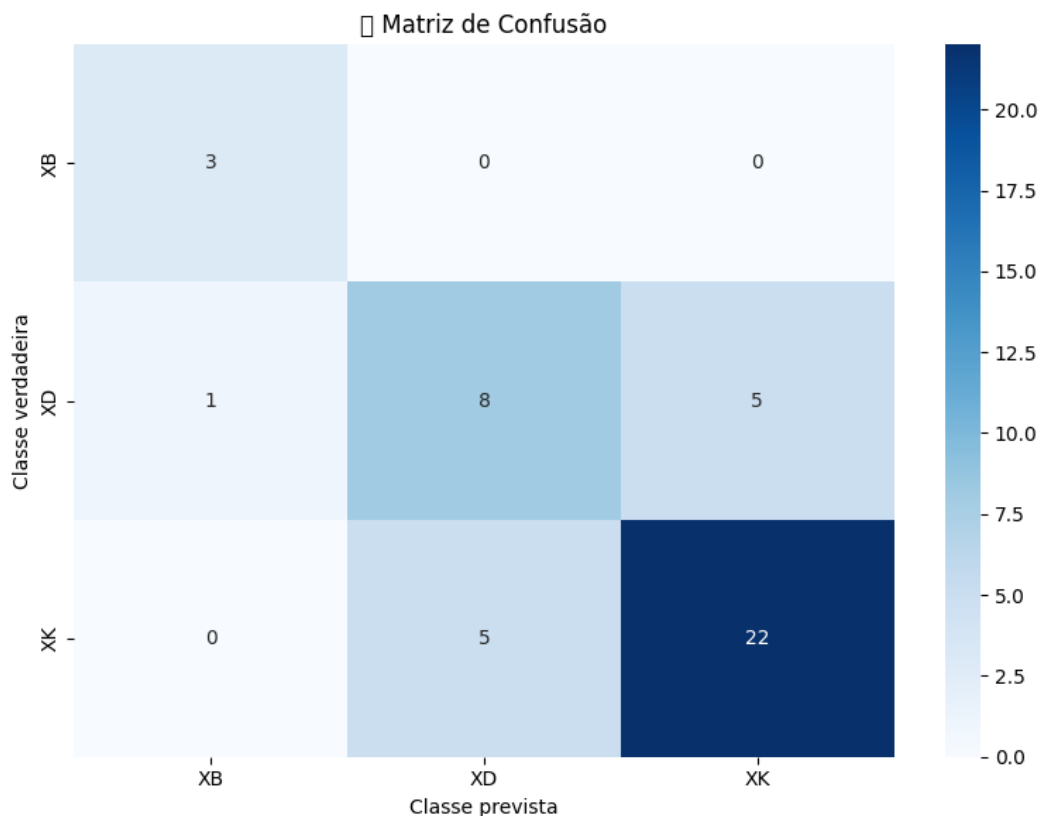
XB 0.75 1.00 0.86 3
XD 0.62 0.57 0.59 14
XK 0.81 0.81 0.81 27

accuracy 0.75 44
macro avg 0.73 0.80 0.75 44
weighted avg 0.75 0.75 0.75 44

=== Matriz de Confusão ===

C:\Users\vitin\AppData\Local\Temp\ipykernel_20368\231474395.py:59: UserWarning: Glyph
128269 (\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) DejaVu Sans.
plt.tight_layout()
c:\Users\vitin\Documents\LABEEC\camarao\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 11. Matriz de confusão.



Diagonal = acertos; fora da diagonal = confusões. As maiores confusões indicam classes visualmente similares ou necessidade de mais dados/curadoria.

4.8 Avaliação do modelo MobileNetV2

Linha	Código	Explicação
1	<code>import torch</code>	Importa bibliotecas/módulos necessários.
2	<code>import os</code>	Importa bibliotecas/módulos necessários.
3	<code>from PIL import Image</code>	Importa bibliotecas/módulos necessários.
4	<code>from torchvision import transforms, models</code>	Importa bibliotecas/módulos necessários.
5	<code>from sklearn.metrics import classification_report, confusion_matrix</code>	Importa bibliotecas/módulos necessários.
6	<code>import numpy as np</code>	Importa bibliotecas/módulos necessários.
7		Linha em branco; separa blocos lógicos.
8	<code># === CONFIGURAÇÕES ===</code>	Comentário; documenta a intenção do bloco (não executa).

Linha	Código	Explicação
9	<code>caminho_dataset = 'camaroes_dataset_split/external'</code>	Atribuição: define/atualiza variável ou objeto.
10	<code>classes = sorted(os.listdir(caminho_dataset)) # os nomes das pastas são os nomes das classes</code>	Atribuição: define/atualiza variável ou objeto.
11	<code>class_to_idx = {classe: idx for idx, classe in enumerate(classes)}</code>	Atribuição: define/atualiza variável ou objeto.
12	<code>caminho_modelo = './models_gray/modelo_final_mobile_net.pth' # Caminho para o modelo treinado</code>	Atribuição: define/atualiza variável ou objeto.
13		Linha em branco; separa blocos lógicos.
14	<code># === TRANSFORMAÇÕES ===</code>	Comentário; documenta a intenção do bloco (não executa).
15	<code>transform = transforms.Compose([</code>	Atribuição: define/atualiza variável ou objeto.
16	<code> transforms.Resize((224, 224)),</code>	Define transformação/pré-processamento de imagens (torchvision.transforms).
17	<code> transforms.ToTensor(),</code>	Define transformação/pré-processamento de imagens (torchvision.transforms).
18	<code> transforms.Normalize(mean=[0.485, 0.456, 0.406],</code>	Atribuição: define/atualiza variável ou objeto.
19	<code> std=[0.229, 0.224, 0.225])</code>	Atribuição: define/atualiza variável ou objeto.
20	<code>])</code>	Instrução do script no contexto do bloco.
21		Linha em branco; separa blocos lógicos.
22	<code># === CARREGAR MODELO ===</code>	Comentário; documenta a intenção do bloco (não executa).
23	<code>model = models.mobilenet_v2(pretrained=False)</code>	Atribuição: define/atualiza variável ou objeto.
24	<code>model.classifier[1] = torch.nn.Linear(model.last_channel, len(classes))</code>	Atribuição: define/atualiza variável ou objeto.
25	<code>model.load_state_dict(torch.load(caminho_modelo, map_location='cpu'))</code>	Atribuição: define/atualiza variável ou objeto.
26	<code>model.eval()</code>	Coloca o modelo em modo de avaliação (desativa dropout etc.).

Linha	Código	Explicação
27		Linha em branco; separa blocos lógicos.
28		Linha em branco; separa blocos lógicos.
29	<code># === AVALIAÇÃO ===</code>	Comentário; documenta a intenção do bloco (não executa).
30	<code>y_true = []</code>	Atribuição: define/atualiza variável ou objeto.
31	<code>y_pred = []</code>	Atribuição: define/atualiza variável ou objeto.
32		Linha em branco; separa blocos lógicos.
33	<code>for classe in classes:</code>	Inicia um laço de iteração.
34	<code> pasta_classe = os.path.join(caminho_dataset, classe)</code>	Atribuição: define/atualiza variável ou objeto.
35	<code> for img_nome in os.listdir(pasta_classe):</code>	Inicia um laço de iteração.
36	<code> if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):</code>	Estrutura condicional (controle de fluxo).
37	<code> continue</code>	Instrução do script no contexto do bloco.
38	<code> img_path = os.path.join(pasta_classe, img_nome)</code>	Atribuição: define/atualiza variável ou objeto.
39	<code> img = Image.open(img_path).convert('RGB')</code>	Atribuição: define/atualiza variável ou objeto.
40	<code> img_tensor = transform(img).unsqueeze(0)</code>	Atribuição: define/atualiza variável ou objeto.
41		Linha em branco; separa blocos lógicos.
42	<code> with torch.no_grad():</code>	Instrução do script no contexto do bloco.
43	<code> output = model(img_tensor)</code>	Atribuição: define/atualiza variável ou objeto.
44	<code> _, pred = torch.max(output, 1)</code>	Atribuição: define/atualiza variável ou objeto.
45		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
46	<code>y_true.append(class_to_idx[classe])</code>	Instrução do script no contexto do bloco.
47	<code>y_pred.append(pred.item())</code>	Instrução do script no contexto do bloco.
48		Linha em branco; separa blocos lógicos.
49	<code># === RELATÓRIO ===</code>	Comentário; documenta a intenção do bloco (não executa).
50	<code>print("=== Classification Report ===")</code>	Imprime informação/diagnóstico no console.
51	<code>print(classification_report(y_true, y_pred, target_names=classes))</code>	Imprime informação/diagnóstico no console.
52		Linha em branco; separa blocos lógicos.
53	<code>print("=== Matriz de Confusão ===")</code>	Imprime informação/diagnóstico no console.
54	<code>cm = confusion_matrix(y_true, y_pred)</code>	Atribuição: define/atualiza variável ou objeto.
55	<code>plt.figure(figsize=(8, 6))</code>	Comando de plotagem/ajuste de figura (Matplotlib).
56	<code>sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=classes, yticklabels=classes)</code>	Instrução do script no contexto do bloco.
57	<code>plt.xlabel("Classe prevista")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
58	<code>plt.ylabel("Classe verdadeira")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
59	<code>plt.title("□ Matriz de Confusão")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
60	<code>plt.tight_layout()</code>	Comando de plotagem/ajuste de figura (Matplotlib).
61	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

```

c:\Users\vitin\Documents\LABEEC\camarao\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\camarao\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)

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

XB 0.50 0.67 0.57 3
XD 0.80 0.29 0.42 14
XK 0.71 0.93 0.81 27

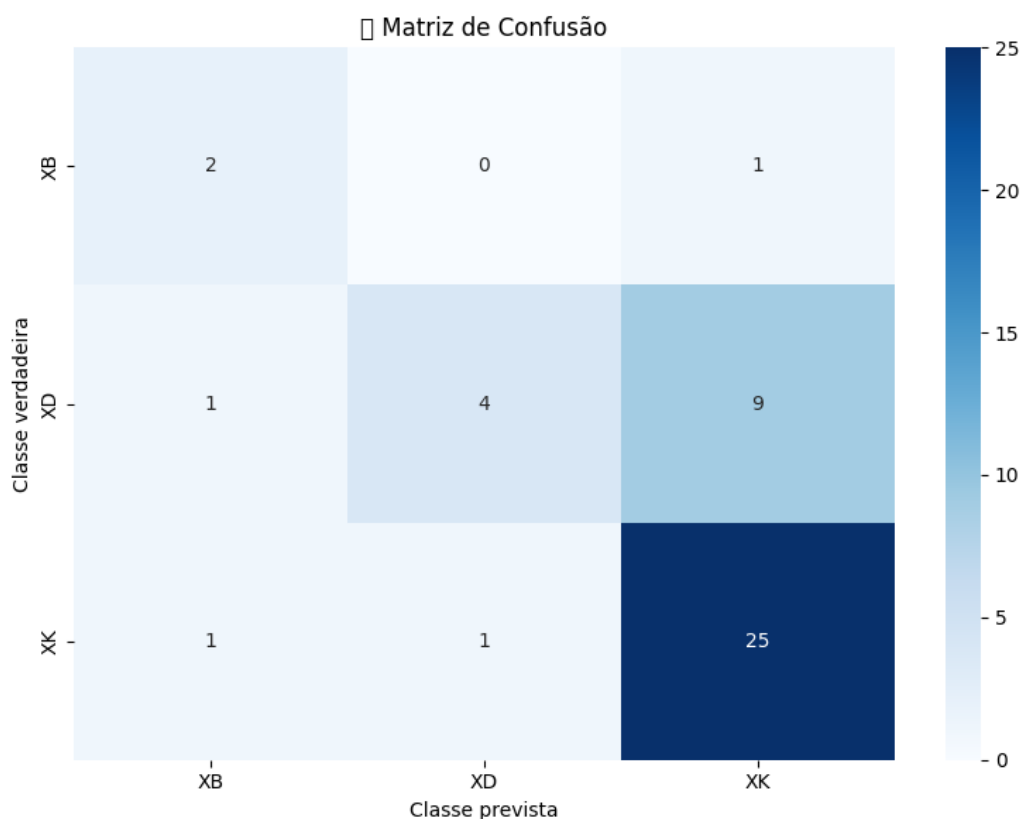
accuracy 0.70 44
macro avg 0.67 0.63 0.60 44
weighted avg 0.73 0.70 0.67 44

=== Matriz de Confusão ===

C:\Users\vitin\AppData\Local\Temp\ipykernel_20368\1114222817.py:60: UserWarning: Glyph
128269 (\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) DejaVu Sans.
plt.tight_layout()
c:\Users\vitin\Documents\LABEEC\camarao\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 12. Matriz de confusão.



Diagonal = acertos; fora da diagonal = confusões. As maiores confusões indicam classes visualmente similares ou necessidade de mais dados/curadoria.

Linha	Código	Explicação
1	<code>import os</code>	Importa bibliotecas/módulos necessários.
2	<code>import torch</code>	Importa bibliotecas/módulos necessários.
3	<code>from PIL import Image</code>	Importa bibliotecas/módulos necessários.
4	<code>import matplotlib.pyplot as plt</code>	Importa bibliotecas/módulos necessários.
5	<code>from torchvision import models, transforms</code>	Importa bibliotecas/módulos necessários.
6	<code>from torchcam.methods import GradCAM</code>	Importa bibliotecas/módulos necessários.
7	<code>from torchcam.utils import overlay_mask</code>	Importa bibliotecas/módulos necessários.
8	<code>from torchvision.transforms.functional import to_pil_image</code>	Importa bibliotecas/módulos necessários.
9		Linha em branco; separa blocos lógicos.
10	<code># === CONFIGURAÇÕES ===</code>	Comentário; documenta a intenção do bloco (não executa).
11	<code>device = torch.device("cuda" if torch.cuda.is_available() else "cpu")</code>	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
12	<code>modelo_path = "./models_gray/modelo_final_res_net.pth"</code>	Atribuição: define/atualiza variável ou objeto.
13	<code>caminho_dataset = 'camaroes_dataset_split/external'</code>	Atribuição: define/atualiza variável ou objeto.
14	<code>classes = sorted(os.listdir(caminho_dataset))</code>	Atribuição: define/atualiza variável ou objeto.
15		Linha em branco; separa blocos lógicos.
16	<code># === TRANSFORMAÇÕES ===</code>	Comentário; documenta a intenção do bloco (não executa).
17	<code>transform = transforms.Compose([</code>	Atribuição: define/atualiza variável ou objeto.
18	<code> transforms.Resize((224, 224)),</code>	Define transformação/pré-processamento de imagens (torchvision.transforms).

Linha	Código	Explicação
19	<code>transforms.ToTensor(),</code>	Define transformação/pré-processamento de imagens (torchvision.transforms).
20	<code>transforms.Normalize(mean=[0.485, 0.456,</code>	Atribuição: define/atualiza variável ou objeto.
21	<code>0.225]),</code> <code>std=[0.229, 0.224,</code>	Atribuição: define/atualiza variável ou objeto.
22	<code>1])</code>	Instrução do script no contexto do bloco.
23		Linha em branco; separa blocos lógicos.
24	<code># === CARREGAR MODELO ===</code>	Comentário; documenta a intenção do bloco (não executa).
25	<code>model = models.resnet18(pretrained=False)</code>	Atribuição: define/atualiza variável ou objeto.
26	<code>model.fc =</code> <code>torch.nn.Linear(model.fc.in_features,</code> <code>len(classes))</code>	Atribuição: define/atualiza variável ou objeto.
27	<code>model.load_state_dict(torch.load(modelo_path,</code> <code>map_location=device))</code>	Atribuição: define/atualiza variável ou objeto.
28	<code>model.to(device)</code>	Instrução do script no contexto do bloco.
29	<code>model.eval()</code>	Coloca o modelo em modo de avaliação (desativa dropout etc.).
30		Linha em branco; separa blocos lógicos.
31	<code>plt.figure(figsize=(15, 6))</code>	Comando de plotagem/ajuste de figura (Matplotlib).
32	<code>cont = 0</code>	Atribuição: define/atualiza variável ou objeto.
33	<code>for idx, classe in enumerate(classes):</code>	Inicia um laço de iteração.
34	<code>pasta_classe =</code> <code>os.path.join(caminho_dataset, classe)</code>	Atribuição: define/atualiza variável ou objeto.
35	<code>img_nome = os.listdir(pasta_classe)[1]</code>	Atribuição: define/atualiza variável ou objeto.
36		Linha em branco; separa blocos lógicos.
37	<code>if not img_nome.lower().endswith(('.png',</code> <code>'.jpg', '.jpeg', '.bmp')):</code>	Estrutura condicional (controle de fluxo).

Linha	Código	Explicação
38	<code>continue</code>	Instrução do script no contexto do bloco.
39		Linha em branco; separa blocos lógicos.
40	<code># === CARREGAR IMAGEM ===</code>	Comentário; documenta a intenção do bloco (não executa).
41	<code>img_path = os.path.join(pasta_classe, img_nome)</code>	Atribuição: define/atualiza variável ou objeto.
42	<code>img = Image.open(img_path).convert('RGB')</code>	Atribuição: define/atualiza variável ou objeto.
43		Linha em branco; separa blocos lógicos.
44	<code>img_tensor = transform(img).unsqueeze(0)</code>	Atribuição: define/atualiza variável ou objeto.
45		Linha em branco; separa blocos lógicos.
46	<code>output = model(img_tensor)</code>	Atribuição: define/atualiza variável ou objeto.
47	<code>_, pred = torch.max(output, 1)</code>	Atribuição: define/atualiza variável ou objeto.
48		Linha em branco; separa blocos lógicos.
49	<code># === APLICAR Grad-CAM ===</code>	Comentário; documenta a intenção do bloco (não executa).
50	<code>cam_extractor = GradCAM(model, target_layer='layer4')</code>	Atribuição: define/atualiza variável ou objeto.
51		Linha em branco; separa blocos lógicos.
52	<code>output = model(img_tensor)</code>	Atribuição: define/atualiza variável ou objeto.
53	<code>pred_class = output.argmax().item()</code>	Atribuição: define/atualiza variável ou objeto.
54		Linha em branco; separa blocos lógicos.
55	<code># Extrair ativação e sobrepor com imagem original</code>	Comentário; documenta a intenção do bloco (não executa).
56	<code>activation_map = cam_extractor(pred_class, output)[0].cpu()</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
57		Linha em branco; separa blocos lógicos.
58	<code># Converter imagem e ativação para PIL</code>	Comentário; documenta a intenção do bloco (não executa).
59	<code>activation_pil = to_pil_image(activation_map, mode='F')</code>	Atribuição: define/atualiza variável ou objeto.
60		Linha em branco; separa blocos lógicos.
61	<code># Gerar imagem com sobreposição (mapa de calor)</code>	Comentário; documenta a intenção do bloco (não executa).
62	<code>result = overlay_mask(img, activation_pil, alpha=0.5)</code>	Atribuição: define/atualiza variável ou objeto.
63	<code>print(img_nome)</code>	Imprime informação/diagnóstico no console.
64	<code># Mostrar com matplotlib</code>	Comentário; documenta a intenção do bloco (não executa).
65	<code>img_nome = img_nome.split('_')[0]</code>	Atribuição: define/atualiza variável ou objeto.
66	<code>cont+=1</code>	Atribuição: define/atualiza variável ou objeto.
67	<code>plt.subplot(3,6, idx + cont)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
68	<code>plt.title(f"Imagem: {img_nome}")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
69	<code>plt.imshow(img)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
70	<code>cont+=1</code>	Atribuição: define/atualiza variável ou objeto.
71	<code>plt.subplot(3,6, idx + cont)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
72	<code>plt.axis("off")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
73	<code>plt.imshow(result)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
74	<code>plt.title(f"Classe prevista: {classes[pred_class]}")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
75	<code>plt.axis("off")</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Linha	Código	Explicação
76	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

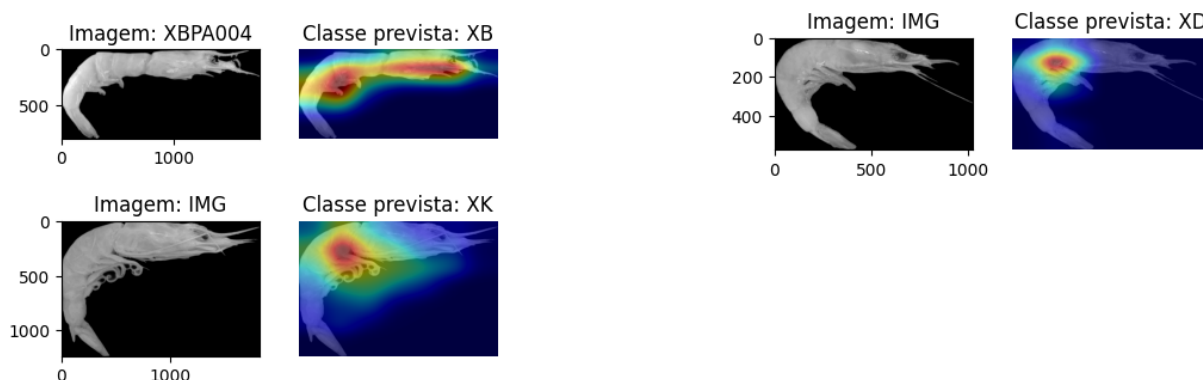
Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\camarao\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\camarao\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)

XBPA004_sem_fundo.png
IMG_0124_sem_fundo.png
IMG_0152_sem_fundo.png
```

Figura 13. Grad-CAM (mapa de atenção).



Indica onde o modelo está 'olhando'. Idealmente destaca regiões anatômicas do organismo; foco no fundo pode indicar viés do dataset (iluminação, suporte, marcações).

Linha	Código	Explicação
1	<code>import os</code>	Importa bibliotecas/módulos necessários.
2	<code>import torch</code>	Importa bibliotecas/módulos necessários.
3	<code>from PIL import Image</code>	Importa bibliotecas/módulos necessários.
4	<code>import matplotlib.pyplot as plt</code>	Importa bibliotecas/módulos necessários.
5	<code>from torchvision import models, transforms</code>	Importa bibliotecas/módulos necessários.
6	<code>from torchcam.methods import GradCAM</code>	Importa bibliotecas/módulos necessários.

Linha	Código	Explicação
25	<code>model = models.alexnet(pretrained=False)</code>	Atribuição: define/atualiza variável ou objeto.
26	<code>model.classifier[6] = torch.nn.Linear(model.classifier[6].in_features, len(classes))</code>	Atribuição: define/atualiza variável ou objeto.
27	<code>model.load_state_dict(torch.load(modelo_path, map_location='cpu'))</code>	Atribuição: define/atualiza variável ou objeto.
28	<code>model.eval()</code>	Coloca o modelo em modo de avaliação (desativa dropout etc.).
29		Linha em branco; separa blocos lógicos.
30	<code>plt.figure(figsize=(15, 6))</code>	Comando de plotagem/ajuste de figura (Matplotlib).
31	<code>cont = 0</code>	Atribuição: define/atualiza variável ou objeto.
32	<code>for idx, classe in enumerate(classes):</code>	Inicia um laço de iteração.
33	<code> pasta_classe = os.path.join(caminho_dataset, classe)</code>	Atribuição: define/atualiza variável ou objeto.
34	<code> img_nome = os.listdir(pasta_classe)[0]</code>	Atribuição: define/atualiza variável ou objeto.
35		Linha em branco; separa blocos lógicos.
36	<code> if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):</code>	Estrutura condicional (controle de fluxo).
37	<code> continue</code>	Instrução do script no contexto do bloco.
38		Linha em branco; separa blocos lógicos.
39	<code> # === CARREGAR IMAGEM ===</code>	Comentário; documenta a intenção do bloco (não executa).
40	<code> img_path = os.path.join(pasta_classe, img_nome)</code>	Atribuição: define/atualiza variável ou objeto.
41	<code> img = Image.open(img_path).convert('RGB')</code>	Atribuição: define/atualiza variável ou objeto.
42		Linha em branco; separa blocos lógicos.
43	<code> img_tensor = transform(img).unsqueeze(0)</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
44		Linha em branco; separa blocos lógicos.
45	<code>output = model(img_tensor)</code>	Atribuição: define/atualiza variável ou objeto.
46	<code>_, pred = torch.max(output, 1)</code>	Atribuição: define/atualiza variável ou objeto.
47		Linha em branco; separa blocos lógicos.
48	<code># === APLICAR Grad-CAM ===</code>	Comentário; documenta a intenção do bloco (não executa).
49	<code>cam_extractor = GradCAM(model, target_layer='features.11')</code>	Atribuição: define/atualiza variável ou objeto.
50		Linha em branco; separa blocos lógicos.
51	<code>output = model(img_tensor)</code>	Atribuição: define/atualiza variável ou objeto.
52	<code>pred_class = output.argmax().item()</code>	Atribuição: define/atualiza variável ou objeto.
53		Linha em branco; separa blocos lógicos.
54	<code># Extrair ativação e sobrepor com imagem original</code>	Comentário; documenta a intenção do bloco (não executa).
55	<code>activation_map = cam_extractor(pred_class, output)[0].cpu()</code>	Atribuição: define/atualiza variável ou objeto.
56		Linha em branco; separa blocos lógicos.
57	<code># Converter imagem e ativação para PIL</code>	Comentário; documenta a intenção do bloco (não executa).
58	<code>activation_pil = to_pil_image(activation_map, mode='F')</code>	Atribuição: define/atualiza variável ou objeto.
59		Linha em branco; separa blocos lógicos.
60	<code># Gerar imagem com sobreposição (mapa de calor)</code>	Comentário; documenta a intenção do bloco (não executa).
61	<code>result = overlay_mask(img, activation_pil, alpha=0.5)</code>	Atribuição: define/atualiza variável ou objeto.
62		Linha em branco; separa blocos lógicos.

Linha	Código	Explicação
63	<code># Mostrar com matplotlib</code>	Comentário; documenta a intenção do bloco (não executa).
64	<code>img_nome = img_nome.split('_')[0]</code>	Atribuição: define/atualiza variável ou objeto.
65	<code>cont+=1</code>	Atribuição: define/atualiza variável ou objeto.
66	<code>plt.subplot(3,6, idx + cont)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
67	<code>plt.title(f"Imagem: {img_nome}")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
68	<code>plt.imshow(img)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
69	<code>cont+=1</code>	Atribuição: define/atualiza variável ou objeto.
70	<code>plt.subplot(3,6, idx + cont)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
71	<code>plt.axis("off")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
72	<code>plt.imshow(result)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
73	<code>plt.title(f"Classe prevista: {classes[pred_class]}")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
74	<code>plt.axis("off")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
75	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

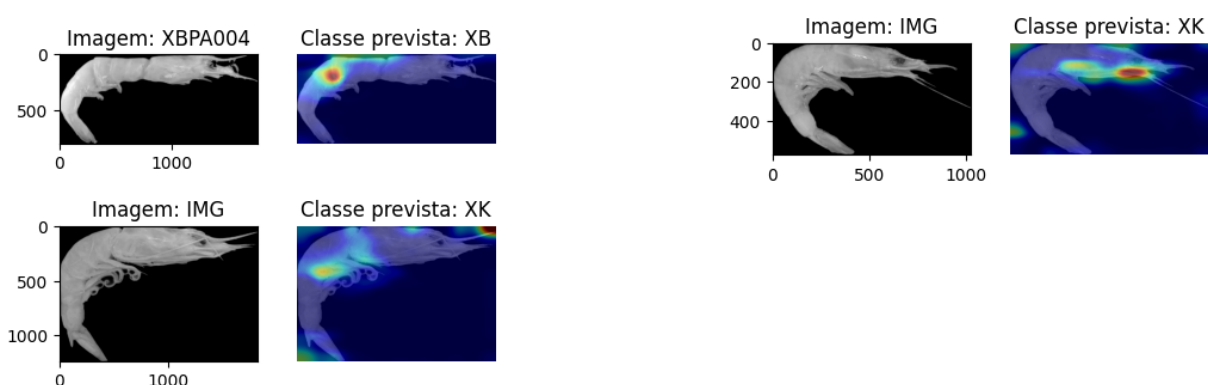
Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\camarao\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\camarao\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)

camaroes_dataset_split/external\XB
camaroes_dataset_split/external\XB\XBPA004_sem_fundo.png
camaroes_dataset_split/external\XD
camaroes_dataset_split/external\XD\IMG_0124_sem_fundo.png
camaroes_dataset_split/external\XK
camaroes_dataset_split/external\XK\IMG_0152_sem_fundo.png
```

Figura 14. Grad-CAM (mapa de atenção).



Indica onde o modelo está 'olhando'. Idealmente destaca regiões anatômicas do organismo; foco no fundo pode indicar viés do dataset (iluminação, suporte, marcações).

Linha	Código	Explicação
1	<code>import os</code>	Importa bibliotecas/módulos necessários.
2	<code>import torch</code>	Importa bibliotecas/módulos necessários.
3	<code>from PIL import Image</code>	Importa bibliotecas/módulos necessários.
4	<code>import matplotlib.pyplot as plt</code>	Importa bibliotecas/módulos necessários.
5	<code>from torchvision import models, transforms</code>	Importa bibliotecas/módulos necessários.
6	<code>from torchcam.methods import GradCAM</code>	Importa bibliotecas/módulos necessários.
7	<code>from torchcam.utils import overlay_mask</code>	Importa bibliotecas/módulos necessários.
8	<code>from torchvision.transforms.functional import to_pil_image</code>	Importa bibliotecas/módulos necessários.

Linha	Código	Explicação
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 = 'camaroetes_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([Atribuição: define/atualiza variável ou objeto.
18	transforms.Resize((224, 224)),	Define transformação/pré-processamento de imagens (torchvision.transforms).
19	transforms.ToTensor(),	Define transformação/pré-processamento de imagens (torchvision.transforms).
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)	Atribuição: define/atualiza variável ou objeto.
26	model.classifier[1] = torch.nn.Linear(model.last_channel, len(classes))	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
27	<code>model.load_state_dict(torch.load(modelo_path, map_location='cpu'))</code>	Atribuição: define/atualiza variável ou objeto.
28	<code>model.eval()</code>	Coloca o modelo em modo de avaliação (desativa dropout etc.).
29		Linha em branco; separa blocos lógicos.
30	<code>plt.figure(figsize=(15, 6))</code>	Comando de plotagem/ajuste de figura (Matplotlib).
31	<code>cont = 0</code>	Atribuição: define/atualiza variável ou objeto.
32	<code>for idx, classe in enumerate(classes):</code>	Inicia um laço de iteração.
33	<code> pasta_classe = os.path.join(caminho_dataset, classe)</code>	Atribuição: define/atualiza variável ou objeto.
34	<code> img_nome = os.listdir(pasta_classe)[0]</code>	Atribuição: define/atualiza variável ou objeto.
35		Linha em branco; separa blocos lógicos.
36	<code> if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):</code>	Estrutura condicional (controle de fluxo).
37	<code> continue</code>	Instrução do script no contexto do bloco.
38		Linha em branco; separa blocos lógicos.
39	<code> # === CARREGAR IMAGEM ===</code>	Comentário; documenta a intenção do bloco (não executa).
40	<code> img_path = os.path.join(pasta_classe, img_nome)</code>	Atribuição: define/atualiza variável ou objeto.
41	<code> img = Image.open(img_path).convert('RGB')</code>	Atribuição: define/atualiza variável ou objeto.
42		Linha em branco; separa blocos lógicos.
43	<code> img_tensor = transform(img).unsqueeze(0)</code>	Atribuição: define/atualiza variável ou objeto.
44		Linha em branco; separa blocos lógicos.
45	<code> output = model(img_tensor)</code>	Atribuição: define/atualiza variável ou objeto.

Linha	Código	Explicação
46	<code>_ , pred = torch.max(output, 1)</code>	Atribuição: define/atualiza variável ou objeto.
47		Linha em branco; separa blocos lógicos.
48	<code># === APLICAR Grad-CAM ===</code>	Comentário; documenta a intenção do bloco (não executa).
49	<code>cam_extractor = GradCAM(model, target_layer='features.18')</code>	Atribuição: define/atualiza variável ou objeto.
50		Linha em branco; separa blocos lógicos.
51	<code>output = model(img_tensor)</code>	Atribuição: define/atualiza variável ou objeto.
52	<code>pred_class = output.argmax().item()</code>	Atribuição: define/atualiza variável ou objeto.
53		Linha em branco; separa blocos lógicos.
54	<code># Extrair ativação e sobrepor com imagem original</code>	Comentário; documenta a intenção do bloco (não executa).
55	<code>activation_map = cam_extractor(pred_class, output)[0].cpu()</code>	Atribuição: define/atualiza variável ou objeto.
56		Linha em branco; separa blocos lógicos.
57	<code># Converter imagem e ativação para PIL</code>	Comentário; documenta a intenção do bloco (não executa).
58	<code>activation_pil = to_pil_image(activation_map, mode='F')</code>	Atribuição: define/atualiza variável ou objeto.
59		Linha em branco; separa blocos lógicos.
60	<code># Gerar imagem com sobreposição (mapa de calor)</code>	Comentário; documenta a intenção do bloco (não executa).
61	<code>result = overlay_mask(img, activation_pil, alpha=0.5)</code>	Atribuição: define/atualiza variável ou objeto.
62		Linha em branco; separa blocos lógicos.
63	<code># Mostrar com matplotlib</code>	Comentário; documenta a intenção do bloco (não executa).
64	<code>print(img_nome)</code>	Imprime informação/diagnóstico no console.

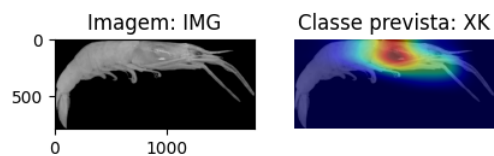
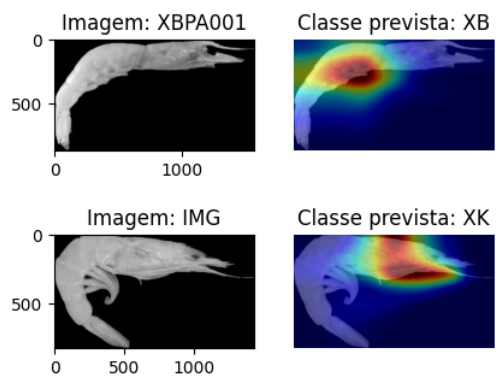
Linha	Código	Explicação
65	<code>img_nome = img_nome.split('_')[0]</code>	Atribuição: define/atualiza variável ou objeto.
66	<code>cont+=1</code>	Atribuição: define/atualiza variável ou objeto.
67	<code>plt.subplot(3,6, idx + cont)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
68	<code>plt.title(f"Imagem: {img_nome}")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
69	<code>plt.imshow(img)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
70	<code>cont+=1</code>	Atribuição: define/atualiza variável ou objeto.
71	<code>plt.subplot(3,6, idx + cont)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
72	<code>plt.axis("off")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
73	<code>plt.imshow(result)</code>	Comando de plotagem/ajuste de figura (Matplotlib).
74	<code>plt.title(f"Classe prevista: {classes[pred_class]}")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
75	<code>plt.axis("off")</code>	Comando de plotagem/ajuste de figura (Matplotlib).
76	<code>plt.show()</code>	Comando de plotagem/ajuste de figura (Matplotlib).

Resultados gerados nesta célula

Saída (texto)

XBPA001_sem_fundo.png
 IMG_0114_sem_fundo.png
 IMG_0148_sem_fundo.png

Figura 15. Grad-CAM (mapa de atenção).



Indica onde o modelo está 'olhando'. Idealmente destaca regiões anatômicas do organismo; foco no fundo pode indicar viés do dataset (iluminação, suporte, marcações).