Shrimp Vision Classifier

Classificação de espécies por Visão Computacional (Livro + Notebook)

https://github.com/fulvioamf/shrimp_vision_classifier

Vitor Carvalho Silva & Fulvio Aurélio de Morais 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.

Your text here 1

Sumário

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

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

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

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

4.2 Separando dataset

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

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

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

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

Resultados gerados nesta célula

Saída (texto)

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

4.3 Treinamento do modelo ResNet18

Linha	Código	Explicação
1	`train_losses = []`	Atribuição: define/atualiza variável ou objeto.
2	`test_accuracies = []`	Atribuição: define/atualiza variável ou objeto.
3		Linha em branco; separa blocos lógicos.
4	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
5	`# 1. CONFIGURAÇÕES`	Comentário; documenta a intenção do bloco (não executa).
6	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
7	`device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
8	`batch_size = 32`	Atribuição: define/atualiza variável ou objeto.
9	`num_epochs = 10`	Atribuição: define/atualiza variável ou objeto.
10	`data_dir = "camaroes_dataset_split" # Caminho para dataset com train/ e test/`	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	`train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True, generator=generator)`	Atribuição: define/atualiza variável ou objeto.
30	`test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False, generator=generator)`	Atribuição: define/atualiza variável ou objeto.
31		Linha em branco; separa blocos lógicos.
32	`num_classes = len(train_dataset.classes)`	Atribuição: define/atualiza variável ou objeto.
33	`class_names = train_dataset.classes`	Atribuição: define/atualiza variável ou objeto.
34		Linha em branco; separa blocos lógicos.
35	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
36	`# 4. MODELO (troque aqui)`	Comentário; documenta a intenção do bloco (não executa).
37	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
38	`model = models.resnet18(pretrained=True)`	Atribuição: define/atualiza variável ou objeto.
39	`model.fc = nn.Linear(model.fc.in_features, num_classes) # Para ResNet18`	Atribuição: define/atualiza variável ou objeto.
40		Linha em branco; separa blocos lógicos.
41	`model = model.to(device)`	Atribuição: define/atualiza variável ou objeto.
42		Linha em branco; separa blocos lógicos.
43	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
44	`# 5. LOSS E OTIMIZADOR`	Comentário; documenta a intenção do bloco (não executa).
45	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
46	`criterion = nn.CrossEntropyLoss()`	Atribuição: define/atualiza variável ou objeto.
47	`optimizer = optim.Adam(model.parameters(), lr=0.0001)`	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	`    train_losses.append(running_loss / len(train_loader))`	Define função de perda (critério de otimização).
70		Linha em branco; separa blocos lógicos.
71	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
72	`# 7. AVALIAÇÃO NO TESTE`	Comentário; documenta a intenção do bloco (não executa).
73	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
74	`model.eval()`	Coloca o modelo em modo de avaliação (desativa dropout etc.).
75	`correct = 0`	Atribuição: define/atualiza variável ou objeto.
76	`total = 0`	Atribuição: define/atualiza variável ou objeto.
77		Linha em branco; separa blocos lógicos.
78	`all_preds = []`	Atribuição: define/atualiza variável ou objeto.
79	`all_labels = []`	Atribuição: define/atualiza variável ou objeto.
80	`all_probs = []`	Atribuição: define/atualiza variável ou objeto.
81		Linha em branco; separa blocos lógicos.
82	`with torch.no_grad():`	Instrução do script no contexto do bloco.
83	`    for images, labels in test_loader:`	Inicia um laço de iteração.
84	`        images, labels = images.to(device), labels.to(device)`	Atribuição: define/atualiza variável ou objeto.
85	`        outputs = model(images)`	Atribuição: define/atualiza variável ou objeto.
86	`        _, predicted = torch.max(outputs, 1)`	Atribuição: define/atualiza variável ou objeto.

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

Linha	Código	Explicação
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	`# One-hot encoding dos rótulos verdadeiros`	Comentário; documenta a intenção do bloco (não executa).
111	`y_true = label_binarize(all_labels, classes=list(range(num_classes)))`	Atribuição: define/atualiza variável ou objeto.
112	`y_score = np.array(all_probs)`	Atribuição: define/atualiza variável ou objeto.
113		Linha em branco; separa blocos lógicos.
114	`# Plotar curva ROC para cada classe`	Comentário; documenta a intenção do bloco (não executa).
115	`plt.figure(figsize=(10, 8))`	Comando de plotagem/ajuste de figura (Matplotlib).
116	`for i in range(num_classes):`	Inicia um laço de iteração.
117	`    fpr, tpr, _ = roc_curve(y_true[:, i], y_score[:, i])`	Atribuição: define/atualiza variável ou objeto.
118	`    roc_auc = auc(fpr, tpr)`	Atribuição: define/atualiza variável ou objeto.
119	`    plt.plot(fpr, tpr, lw=2, label=f'{class_names[i]} (AUC = {roc_auc:.2f})')`	Comando de plotagem/ajuste de figura (Matplotlib).
120		Linha em branco; separa blocos lógicos.
121	`plt.plot([0, 1], [0, 1], 'k--', lw=2)`	Comando de plotagem/ajuste de figura (Matplotlib).
122	`plt.xlabel('Taxa de Falsos Positivos (FPR)')`	Comando de plotagem/ajuste de figura (Matplotlib).
123	`plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')`	Comando de plotagem/ajuste de figura (Matplotlib).
124	`plt.title('Curvas ROC por Classe')`	Comando de plotagem/ajuste de figura (Matplotlib).

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

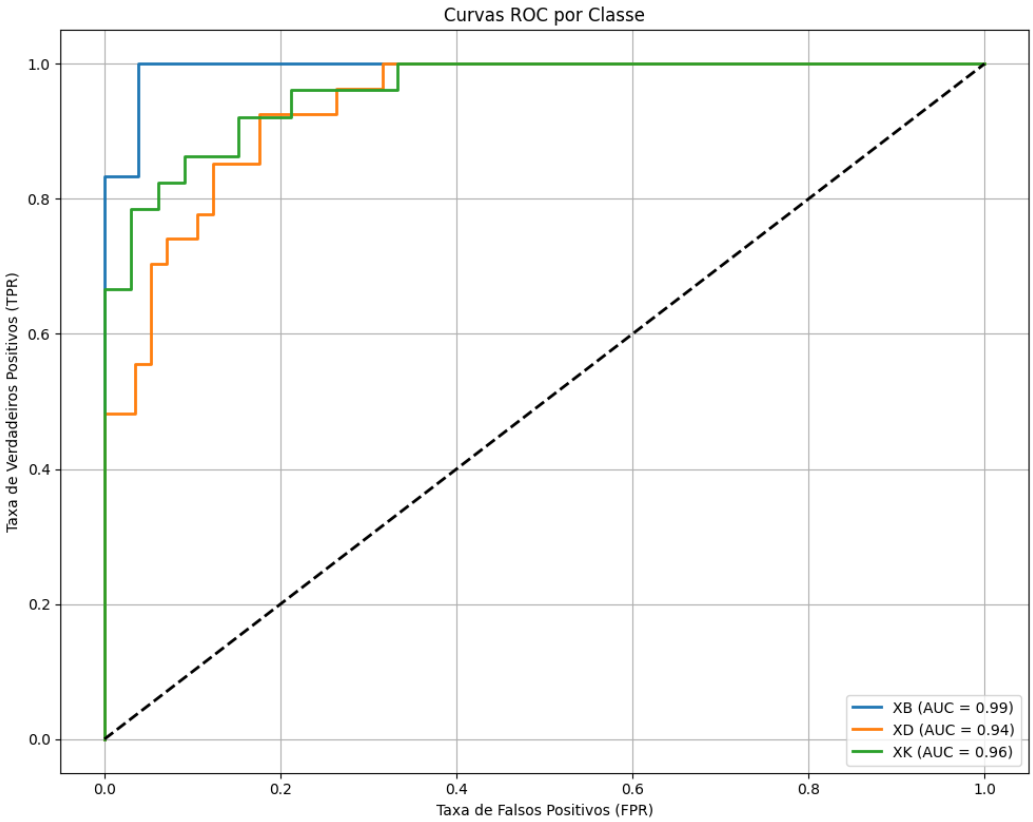
Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\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.

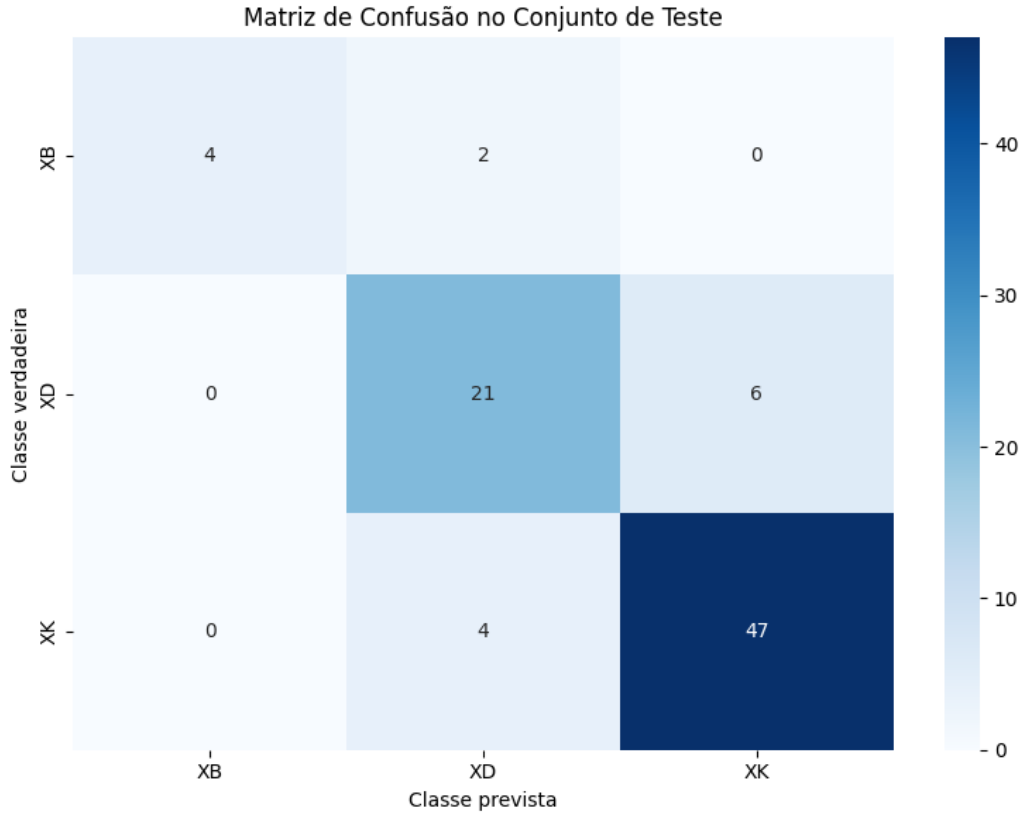


Diagonal = acertos; fora da diagonal = confusões. As maiores confusões indicam classes visualmente similares ou necessidade de mais dados/curadoria.

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

Resultados gerados nesta célula

Figura 2. Matriz de confusão.



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

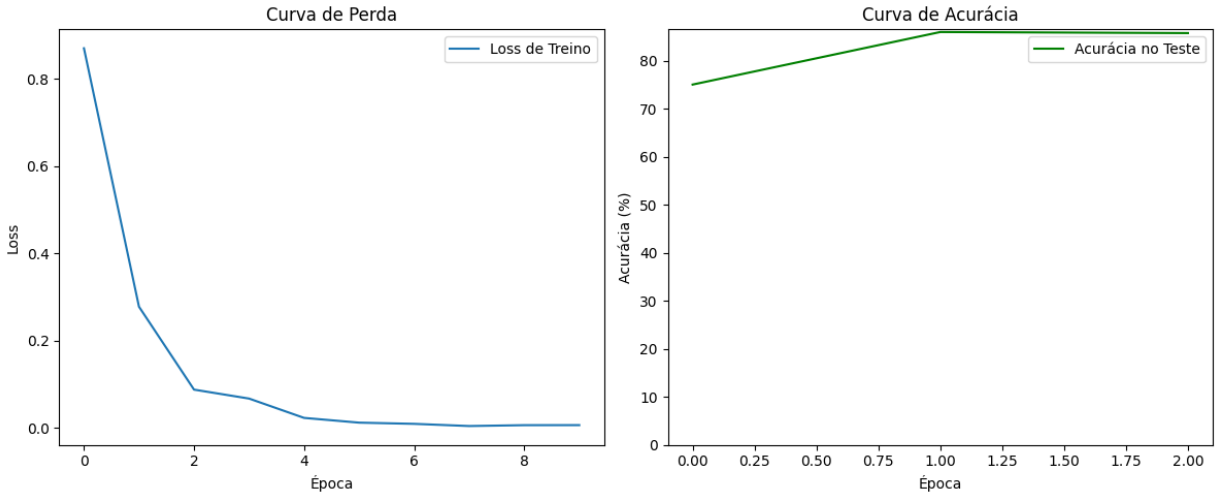
Linha	Código	Explicação
1	`torch.save(model.state_dict(), './models_gray/modelo_final_res_net.pth')`	Instrução do script no contexto do bloco.

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

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

Resultados gerados nesta célula

Figura 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	`train_losses = []`	Atribuição: define/atualiza variável ou objeto.
2	`test_accuracies = []`	Atribuição: define/atualiza variável ou objeto.
3		Linha em branco; separa blocos lógicos.
4	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
5	`# 1. CONFIGURAÇÕES`	Comentário; documenta a intenção do bloco (não executa).
6	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
7	`device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
8	`batch_size = 32`	Atribuição: define/atualiza variável ou objeto.
9	`num_epochs = 10`	Atribuição: define/atualiza variável ou objeto.
10	`data_dir = "camaroes_dataset_split" # Caminho para dataset com train/ e test/`	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	`test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False, generator=generator)`	Atribuição: define/atualiza variável ou objeto.
31		Linha em branco; separa blocos lógicos.
32	`num_classes = len(train_dataset.classes)`	Atribuição: define/atualiza variável ou objeto.
33	`class_names = train_dataset.classes`	Atribuição: define/atualiza variável ou objeto.
34		Linha em branco; separa blocos lógicos.
35	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
36	`# 4. MODELO`	Comentário; documenta a intenção do bloco (não executa).
37	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
38		Linha em branco; separa blocos lógicos.
39	`model = models.alexnet(pretrained=True)`	Atribuição: define/atualiza variável ou objeto.
40	`model.classifier[6] = nn.Linear(model.classifier[6].in_features, num_classes)`	Atribuição: define/atualiza variável ou objeto.
41		Linha em branco; separa blocos lógicos.
42	`model = model.to(device)`	Atribuição: define/atualiza variável ou objeto.
43		Linha em branco; separa blocos lógicos.
44	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
45	`# 5. LOSS E OTIMIZADOR`	Comentário; documenta a intenção do bloco (não executa).
46	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
47	`criterion = nn.CrossEntropyLoss()`	Atribuição: define/atualiza variável ou objeto.
48	`optimizer = optim.Adam(model.parameters(), lr=0.0001)`	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	`    train_losses.append(running_loss / len(train_loader))`	Define função de perda (critério de otimização).
71		Linha em branco; separa blocos lógicos.
72	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
73	`# 7. AVALIAÇÃO NO TESTE`	Comentário; documenta a intenção do bloco (não executa).
74	`# ======================`	Comentário; documenta a intenção do bloco (não executa).
75	`model.eval()`	Coloca o modelo em modo de avaliação (desativa dropout etc.).
76	`correct = 0`	Atribuição: define/atualiza variável ou objeto.
77	`total = 0`	Atribuição: define/atualiza variável ou objeto.
78		Linha em branco; separa blocos lógicos.
79	`all_preds = []`	Atribuição: define/atualiza variável ou objeto.
80	`all_labels = []`	Atribuição: define/atualiza variável ou objeto.
81	`all_probs = []`	Atribuição: define/atualiza variável ou objeto.
82	`with torch.no_grad():`	Instrução do script no contexto do bloco.
83	`    for images, labels in test_loader:`	Inicia um laço de iteração.
84	`        images, labels = images.to(device), labels.to(device)`	Atribuição: define/atualiza variável ou objeto.
85	`        outputs = model(images)`	Atribuição: define/atualiza variável ou objeto.
86	`        _, predicted = torch.max(outputs, 1)`	Atribuição: define/atualiza variável ou objeto.
87		Linha em branco; separa blocos lógicos.

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

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

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

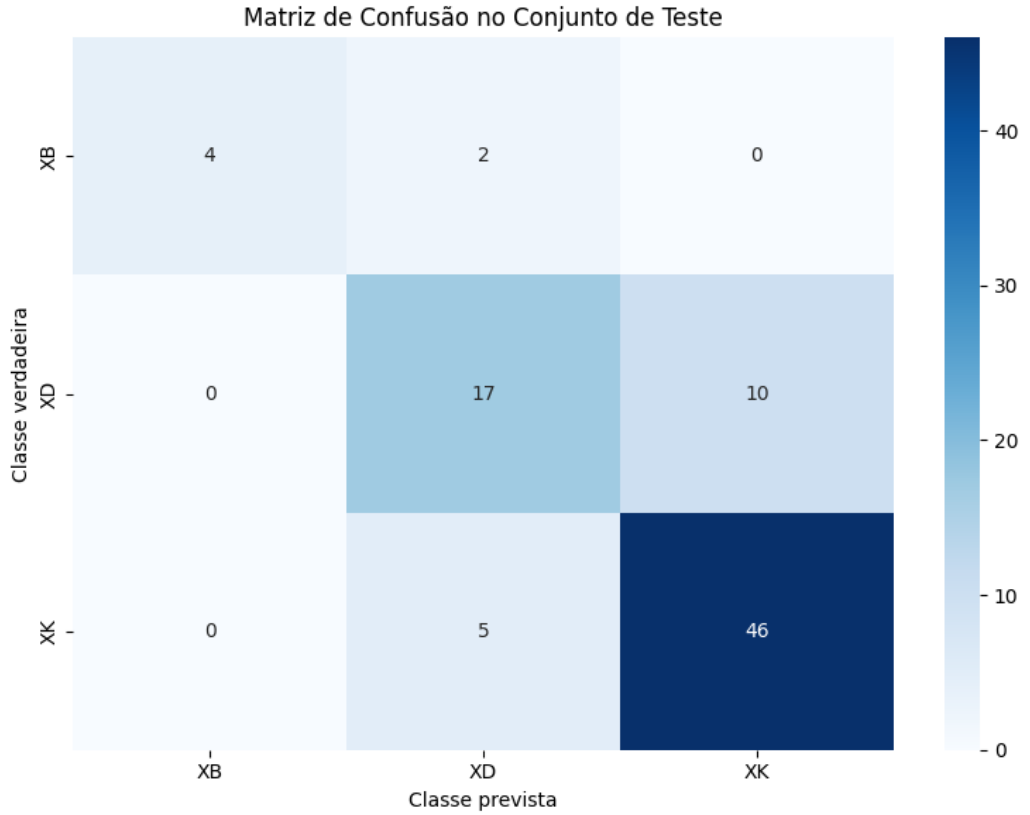
Resultados gerados nesta célula

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
```

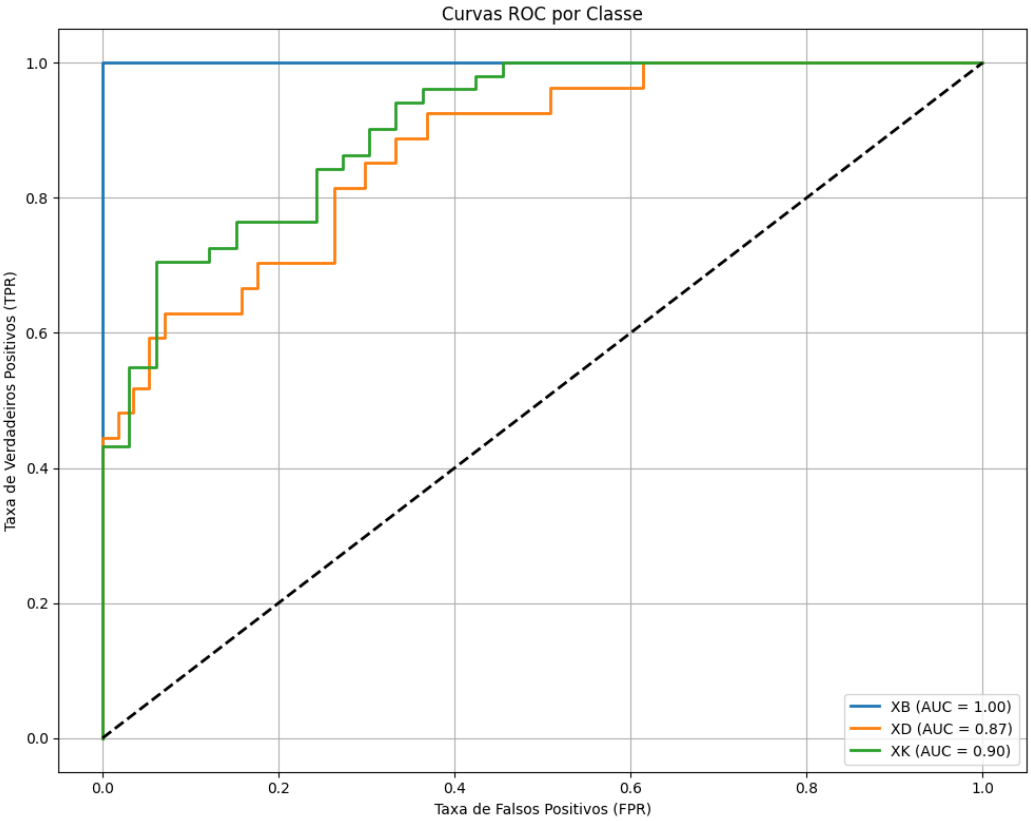
Interpretação rápida: acurácia reportada ~79.76.

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.

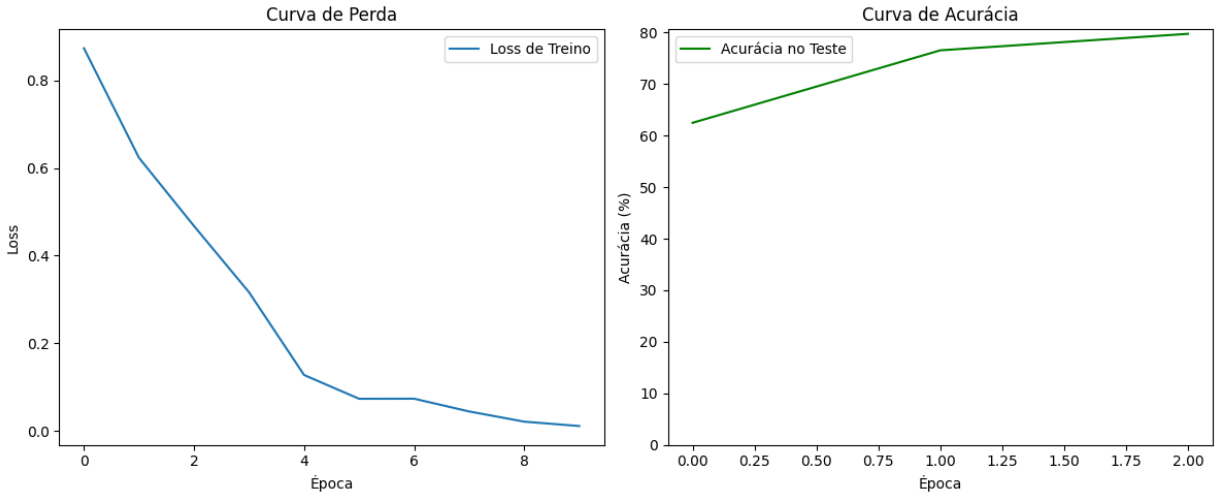
Linha	Código	Explicação
1	`torch.save(model.state_dict(), './models_gray/modelo_final_alex_net.pth')`	Instrução do script no contexto do bloco.

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

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

Resultados gerados nesta célula

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

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

Linha	Código	Explicação
81	`    for images, labels in test_loader:`	Inicia um laço de iteração.
82	`        images, labels = images.to(device), labels.to(device)`	Atribuição: define/atualiza variável ou objeto.
83	`        outputs = model(images)`	Atribuição: define/atualiza variável ou objeto.
84	`        _, predicted = torch.max(outputs, 1)`	Atribuição: define/atualiza variável ou objeto.
85		Linha em branco; separa blocos lógicos.
86	`        total += labels.size(0)`	Atribuição: define/atualiza variável ou objeto.
87	`        correct += (predicted == labels).sum().item()`	Instrução do script no contexto do bloco.
88		Linha em branco; separa blocos lógicos.
89	`        # Armazena previsões e rótulos verdadeiros`	Comentário; documenta a intenção do bloco (não executa).
90	`        probs = torch.softmax(outputs, dim=1).cpu().numpy()`	Atribuição: define/atualiza variável ou objeto.
91	`        all_probs.extend(probs)`	Instrução do script no contexto do bloco.
92	`        all_preds.extend(predicted.cpu().numpy())`	Instrução do script no contexto do bloco.
93	`        all_labels.extend(labels.cpu().numpy())`	Instrução do script no contexto do bloco.
94		Linha em branco; separa blocos lógicos.
95	`        # Acurácia final`	Comentário; documenta a intenção do bloco (não executa).
96	`        acc = 100 * correct / total`	Atribuição: define/atualiza variável ou objeto.
97	`        test_accuracies.append(acc)`	Instrução do script no contexto do bloco.
98	`    print(f" Acurácia no teste após época {epoch+1}: {acc:.2f}%")`	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	`# One-hot encoding dos rótulos verdadeiros`	Comentário; documenta a intenção do bloco (não executa).
120	`y_true = label_binarize(all_labels, classes=list(range(num_classes)))`	Atribuição: define/atualiza variável ou objeto.
121	`y_score = np.array(all_probs)`	Atribuição: define/atualiza variável ou objeto.
122		Linha em branco; separa blocos lógicos.
123	`# Plotar curva ROC para cada classe`	Comentário; documenta a intenção do bloco (não executa).
124	`plt.figure(figsize=(10, 8))`	Comando de plotagem/ajuste de figura (Matplotlib).
125	`for i in range(num_classes):`	Inicia um laço de iteração.
126	`    fpr, tpr, _ = roc_curve(y_true[:, i], y_score[:, i])`	Atribuição: define/atualiza variável ou objeto.
127	`    roc_auc = auc(fpr, tpr)`	Atribuição: define/atualiza variável ou objeto.
128	`    plt.plot(fpr, tpr, lw=2, label=f'{class_names[i]} (AUC = {roc_auc:.2f})')`	Comando de plotagem/ajuste de figura (Matplotlib).
129		Linha em branco; separa blocos lógicos.
130	`plt.plot([0, 1], [0, 1], 'k--', lw=2)`	Comando de plotagem/ajuste de figura (Matplotlib).
131	`plt.xlabel('Taxa de Falsos Positivos (FPR)')`	Comando de plotagem/ajuste de figura (Matplotlib).
132	`plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')`	Comando de plotagem/ajuste de figura (Matplotlib).
133	`plt.title('Curvas ROC por Classe')`	Comando de plotagem/ajuste de figura (Matplotlib).
134	`plt.legend(loc="lower right")`	Comando de plotagem/ajuste de figura (Matplotlib).
135	`plt.grid(True)`	Comando de plotagem/ajuste de figura (Matplotlib).
136	`plt.tight_layout()`	Comando de plotagem/ajuste de figura (Matplotlib).
137	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

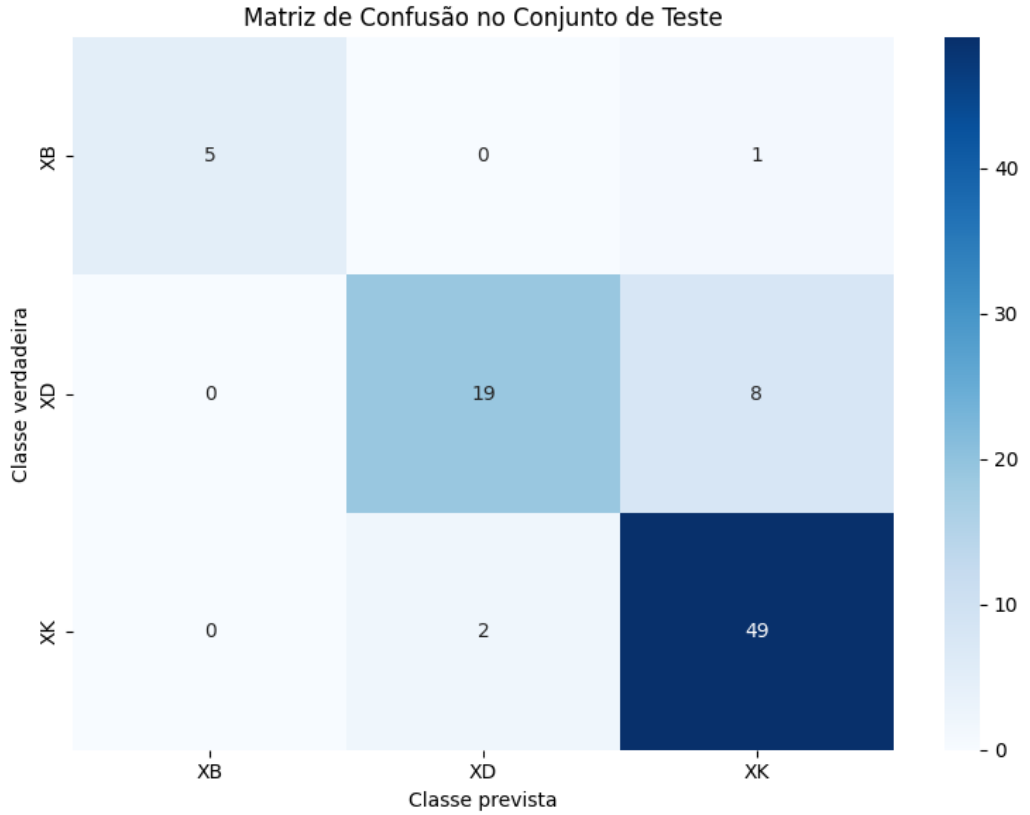
Resultados gerados nesta célula

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
```

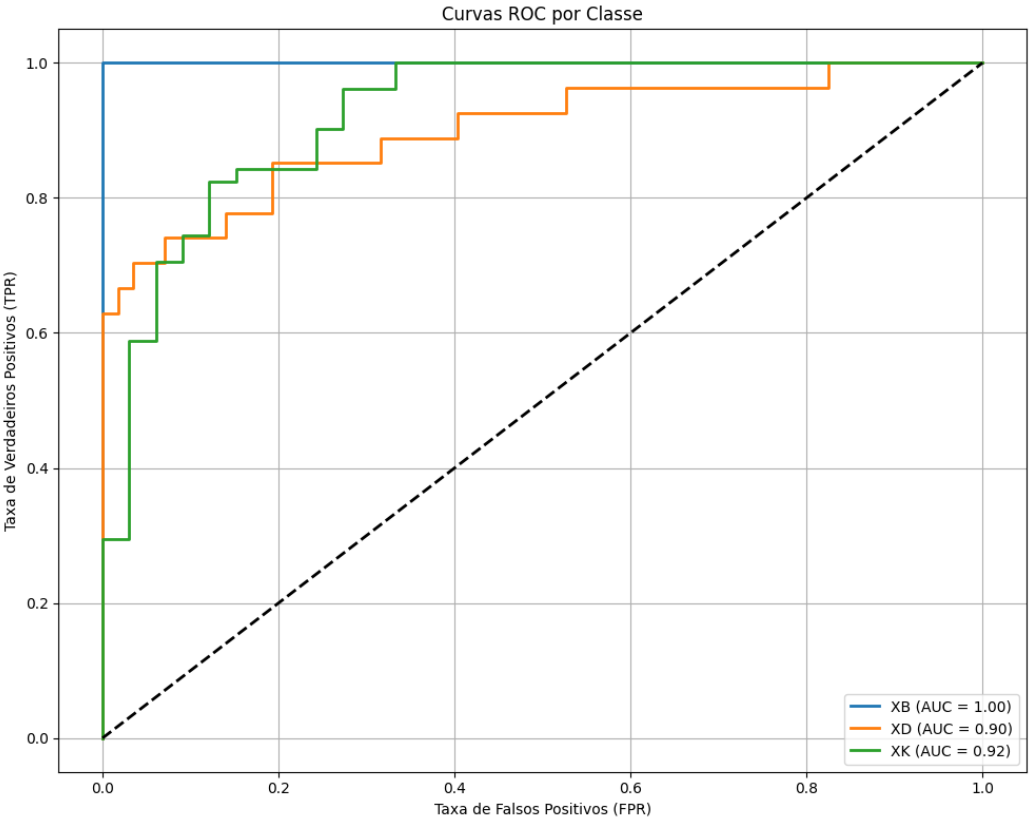
Interpretação rápida: acurácia reportada ~86.90.

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.

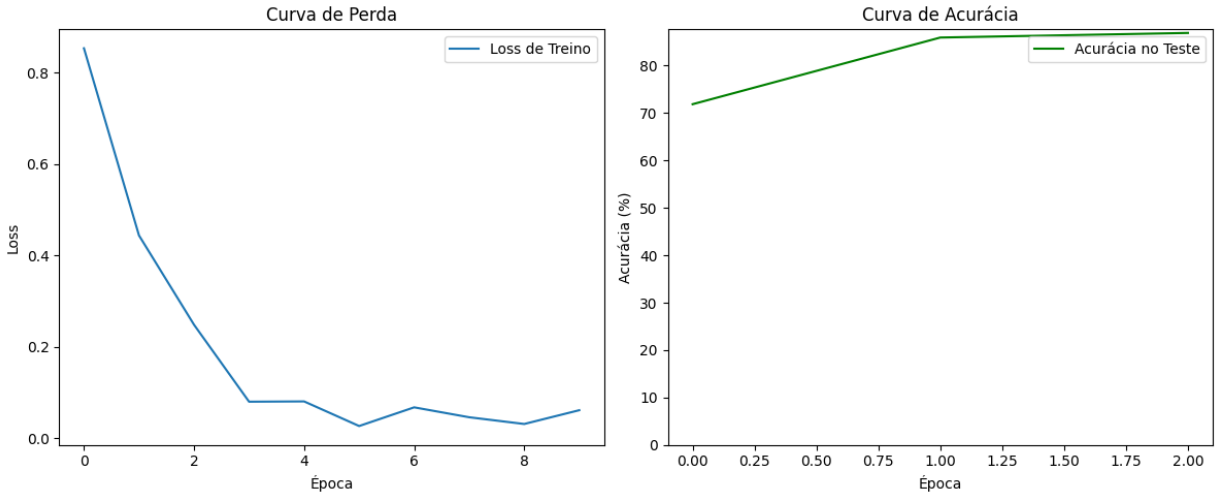
Linha	Código	Explicação
1	`torch.save(model.state_dict(), './models_gray/modelo_final_mobile_net.pth')`	Instrução do script no contexto do bloco.

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

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

Resultados gerados nesta célula

Figura 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	`import torch`	Importa bibliotecas/módulos necessários.
2	`import os`	Importa bibliotecas/módulos necessários.
3	`from PIL import Image`	Importa bibliotecas/módulos necessários.
4	`from torchvision import transforms, models`	Importa bibliotecas/módulos necessários.
5	`from sklearn.metrics import classification_report, confusion_matrix`	Importa bibliotecas/módulos necessários.

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

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

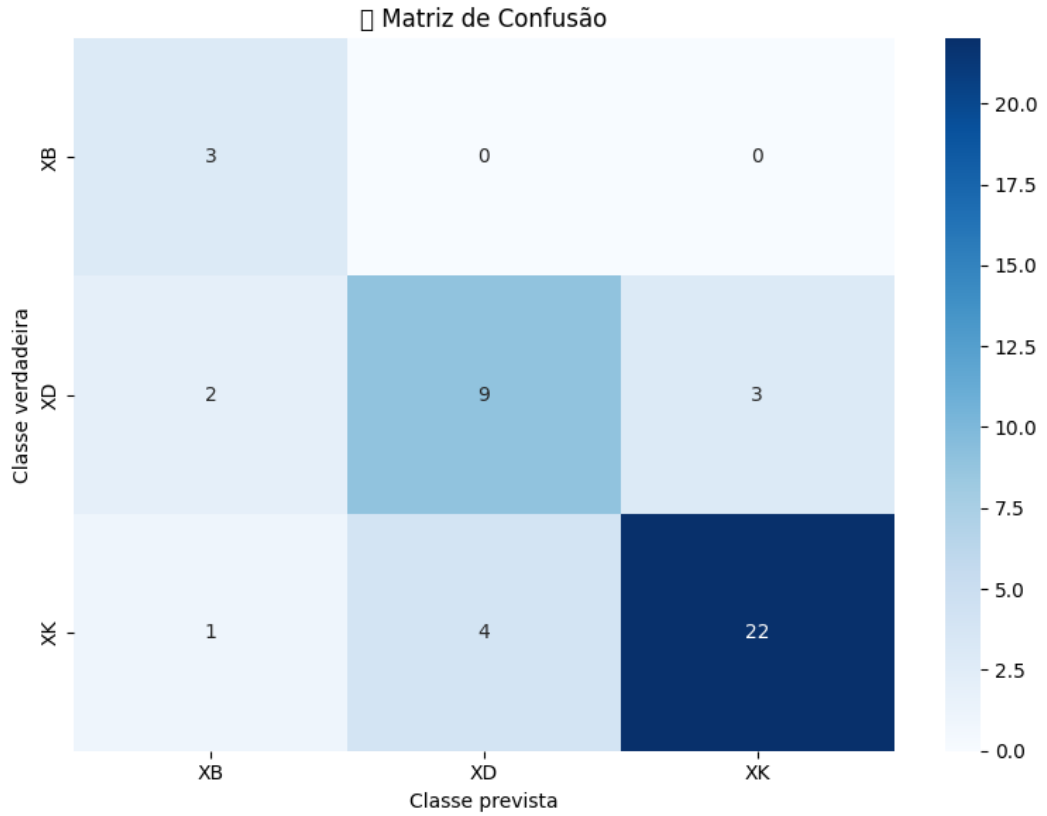
Linha	Código	Explicação
43	`            _, pred = torch.max(output, 1)`	Atribuição: define/atualiza variável ou objeto.
44		Linha em branco; separa blocos lógicos.
45	`        y_true.append(class_to_idx[classe])`	Instrução do script no contexto do bloco.
46	`        y_pred.append(pred.item())`	Instrução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

Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\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 Avalição do Modelo AlexNet

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

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

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

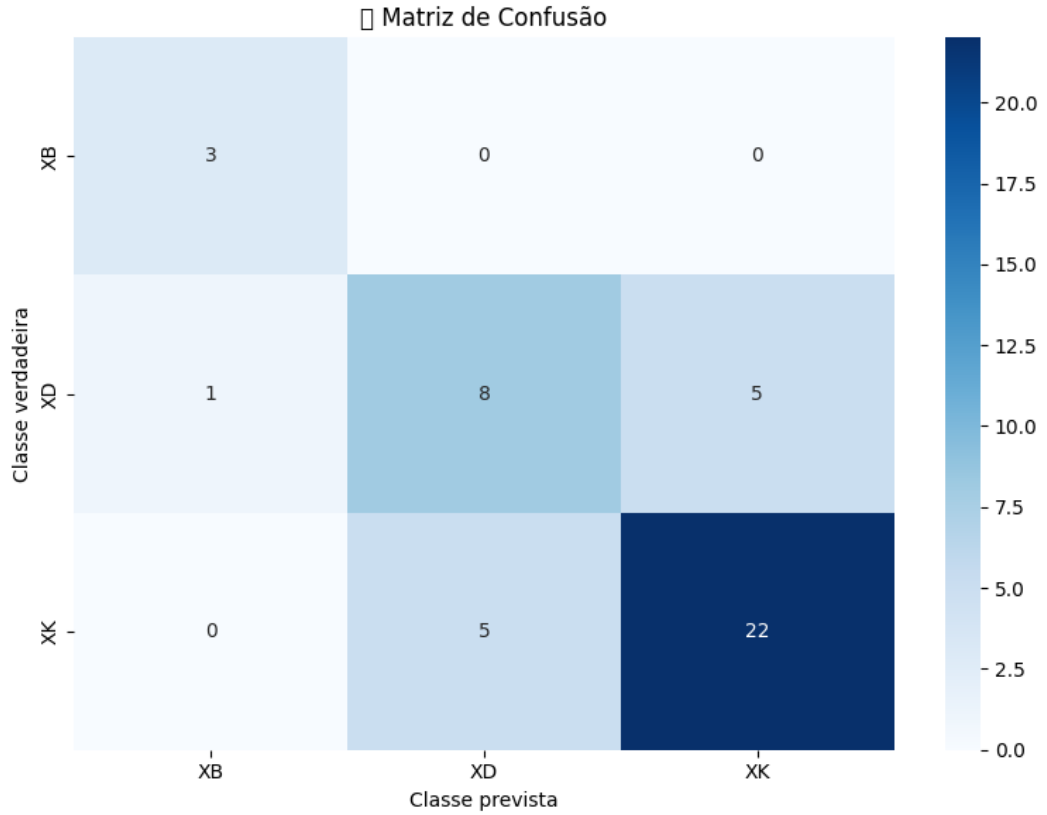
Linha	Código	Explicação
56	`plt.xlabel("Classe prevista")`	Comando de plotagem/ajuste de figura (Matplotlib).
57	`plt.ylabel("Classe verdadeira")`	Comando de plotagem/ajuste de figura (Matplotlib).
58	`plt.title("□ Matriz de Confusão")`	Comando de plotagem/ajuste de figura (Matplotlib).
59	`plt.tight_layout()`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

Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\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	`import torch`	Importa bibliotecas/módulos necessários.
2	`import os`	Importa bibliotecas/módulos necessários.
3	`from PIL import Image`	Importa bibliotecas/módulos necessários.
4	`from torchvision import transforms, models`	Importa bibliotecas/módulos necessários.
5	`from sklearn.metrics import classification_report, confusion_matrix`	Importa bibliotecas/módulos necessários.
6	`import numpy as np`	Importa bibliotecas/módulos necessários.
7		Linha em branco; separa blocos lógicos.
8	`# === CONFIGURAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).

Linha	Código	Explicação
9	`caminho_dataset = 'camaroes_dataset_split/external'`	Atribuição: define/atualiza variável ou objeto.
10	`classes = sorted(os.listdir(caminho_dataset)) # os nomes das pastas são os nomes das classes`	Atribuição: define/atualiza variável ou objeto.
11	`class_to_idx = {classe: idx for idx, classe in enumerate(classes)}`	Atribuição: define/atualiza variável ou objeto.
12	`caminho_modelo = './models_gray/modelo_final_mobile_net.pth' # Caminho para o modelo treinado`	Atribuição: define/atualiza variável ou objeto.
13		Linha em branco; separa blocos lógicos.
14	`# === TRANSFORMAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
15	`transform = transforms.Compose([`	Atribuição: define/atualiza variável ou objeto.
16	`    transforms.Resize((224, 224)),`	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],`	Atribuição: define/atualiza variável ou objeto.
19	`                         std=[0.229, 0.224, 0.225])`	Atribuição: define/atualiza variável ou objeto.
20	`])`	Instrução do script no contexto do bloco.
21		Linha em branco; separa blocos lógicos.
22	`# === CARREGAR MODELO ===`	Comentário; documenta a intenção do bloco (não executa).
23	`model = models.mobilenet_v2(pretrained=False)`	Atribuição: define/atualiza variável ou objeto.
24	`model.classifier[1] = torch.nn.Linear(model.last_channel, len(classes))`	Atribuição: define/atualiza variável ou objeto.
25	`model.load_state_dict(torch.load(caminho_modelo, map_location='cpu'))`	Atribuição: define/atualiza variável ou objeto.
26	`model.eval()`	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	`# === AVALIAÇÃO ===`	Comentário; documenta a intenção do bloco (não executa).
30	`y_true = []`	Atribuição: define/atualiza variável ou objeto.
31	`y_pred = []`	Atribuição: define/atualiza variável ou objeto.
32		Linha em branco; separa blocos lógicos.
33	`for classe in classes:`	Inicia um laço de iteração.
34	`    pasta_classe = os.path.join(caminho_dataset, classe)`	Atribuição: define/atualiza variável ou objeto.
35	`    for img_nome in os.listdir(pasta_classe):`	Inicia um laço de iteração.
36	`        if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):`	Estrutura condicional (controle de fluxo).
37	`            continue`	Instrução do script no contexto do bloco.
38	`        img_path = os.path.join(pasta_classe, img_nome)`	Atribuição: define/atualiza variável ou objeto.
39	`        img = Image.open(img_path).convert('RGB')`	Atribuição: define/atualiza variável ou objeto.
40	`        img_tensor = transform(img).unsqueeze(0)`	Atribuição: define/atualiza variável ou objeto.
41		Linha em branco; separa blocos lógicos.
42	`        with torch.no_grad():`	Instrução do script no contexto do bloco.
43	`            output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.
44	`            _, pred = torch.max(output, 1)`	Atribuição: define/atualiza variável ou objeto.
45		Linha em branco; separa blocos lógicos.

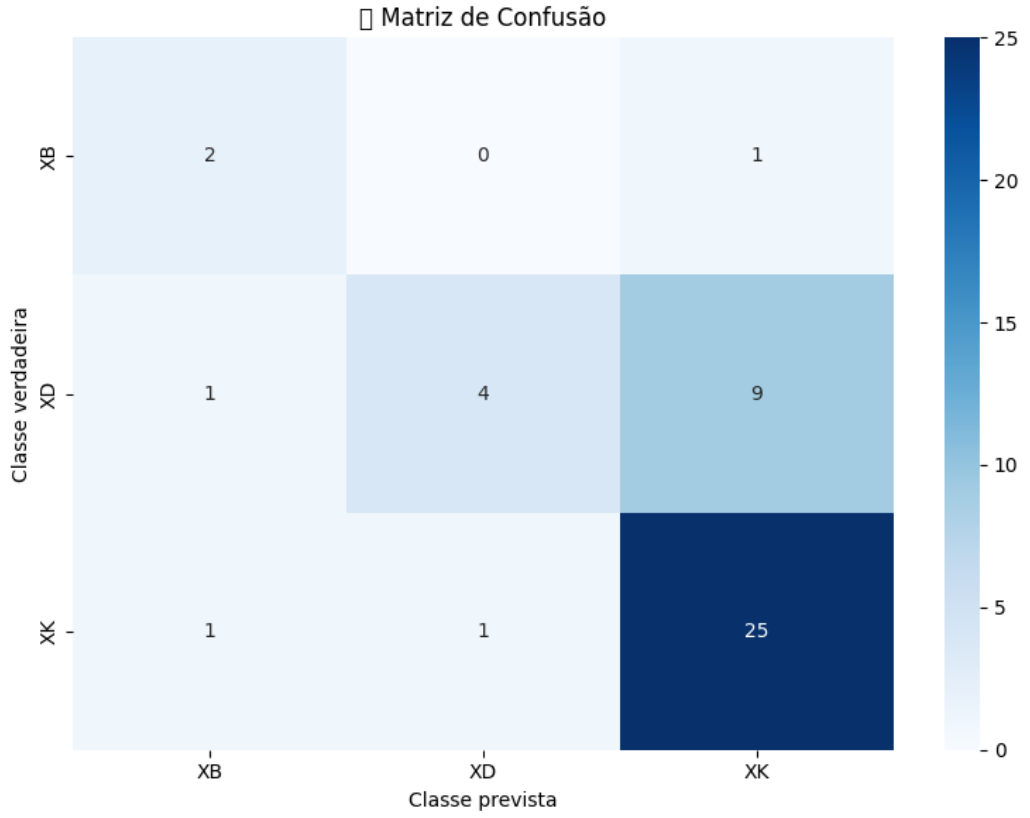
Linha	Código	Explicação
46	`        y_true.append(class_to_idx[classe])`	Instrução do script no contexto do bloco.
47	`        y_pred.append(pred.item())`	Instrução do script no contexto do bloco.
48		Linha em branco; separa blocos lógicos.
49	`# === RELATÓRIO ===`	Comentário; documenta a intenção do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

Resultados gerados nesta célula

Saída (texto)

```
c:\Users\vitin\Documents\LABEEC\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	`import os`	Importa bibliotecas/módulos necessários.
2	`import torch`	Importa bibliotecas/módulos necessários.
3	`from PIL import Image`	Importa bibliotecas/módulos necessários.
4	`import matplotlib.pyplot as plt`	Importa bibliotecas/módulos necessários.
5	`from torchvision import models, transforms`	Importa bibliotecas/módulos necessários.
6	`from torchcam.methods import GradCAM`	Importa bibliotecas/módulos necessários.
7	`from torchcam.utils import overlay_mask`	Importa bibliotecas/módulos necessários.
8	`from torchvision.transforms.functional import to_pil_image`	Importa bibliotecas/módulos necessários.
9		Linha em branco; separa blocos lógicos.
10	`# === CONFIGURAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
11	`device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
12	`modelo_path = "./models_gray/modelo_final_res_net.pth"`	Atribuição: define/atualiza variável ou objeto.
13	`caminho_dataset = 'camaroes_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).

Linha	Código	Explicação
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.resnet18(pretrained=False)`	Atribuição: define/atualiza variável ou objeto.
26	`model.fc = torch.nn.Linear(model.fc.in_features, len(classes))`	Atribuição: define/atualiza variável ou objeto.
27	`model.load_state_dict(torch.load(modelo_path, map_location=device))`	Atribuição: define/atualiza variável ou objeto.
28	`model.to(device)`	Instrução do script no contexto do bloco.
29	`model.eval()`	Coloca o modelo em modo de avaliação (desativa dropout etc.).
30		Linha em branco; separa blocos lógicos.
31	`plt.figure(figsize=(15, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
32	`cont = 0`	Atribuição: define/atualiza variável ou objeto.
33	`for idx, classe in enumerate(classes):`	Inicia um laço de iteração.
34	`    pasta_classe = os.path.join(caminho_dataset, classe)`	Atribuição: define/atualiza variável ou objeto.
35	`    img_nome = os.listdir(pasta_classe)[1]`	Atribuição: define/atualiza variável ou objeto.
36		Linha em branco; separa blocos lógicos.
37	`    if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):`	Estrutura condicional (controle de fluxo).

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

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

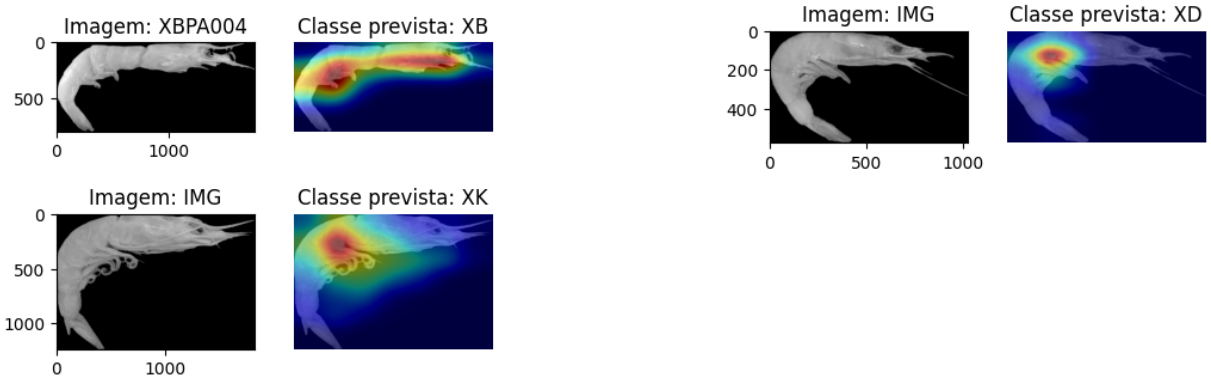
Linha	Código	Explicação
76	`plt.show()`	Comando de plotagem/ajuste de figura (Matplotlib).

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	`import os`	Importa bibliotecas/módulos necessários.
2	`import torch`	Importa bibliotecas/módulos necessários.
3	`from PIL import Image`	Importa bibliotecas/módulos necessários.
4	`import matplotlib.pyplot as plt`	Importa bibliotecas/módulos necessários.
5	`from torchvision import models, transforms`	Importa bibliotecas/módulos necessários.
6	`from torchcam.methods import GradCAM`	Importa bibliotecas/módulos necessários.

Linha	Código	Explicação
7	`from torchcam.utils import overlay_mask`	Importa bibliotecas/módulos necessários.
8	`from torchvision.transforms.functional import to_pil_image`	Importa bibliotecas/módulos necessários.
9		Linha em branco; separa blocos lógicos.
10	`# === CONFIGURAÇÕES ===`	Comentário; documenta a intenção do bloco (não executa).
11	`device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`	Seleciona GPU (cuda) se disponível; caso contrário, CPU.
12	`modelo_path = "./models_gray/modelo_final_alex_net.pth"`	Atribuição: define/atualiza variável ou objeto.
13	`caminho_dataset = 'camaroes_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).

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

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

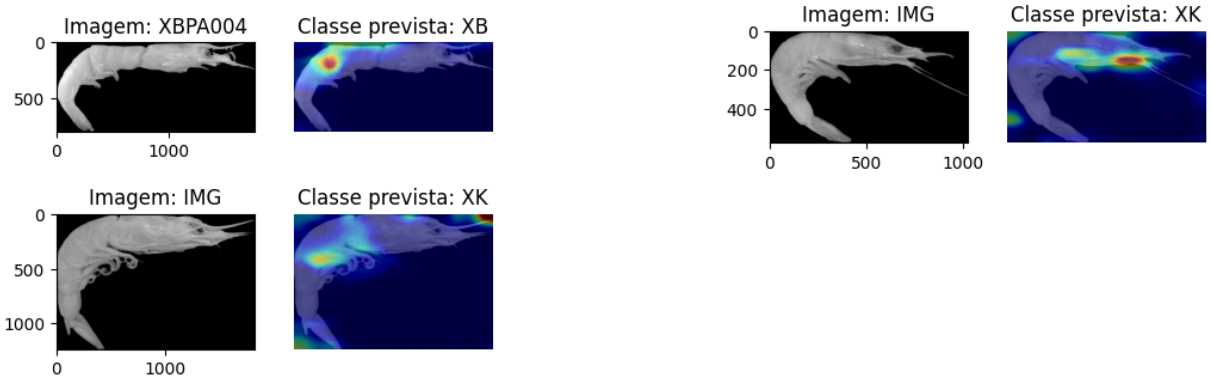
Linha	Código	Explicação
63	`    # Mostrar com matplotlib`	Comentário; documenta a intenção do bloco (não executa).
64	`    img_nome = img_nome.split('_')[0]`	Atribuição: define/atualiza variável ou objeto.
65	`    cont+=1`	Atribuição: define/atualiza variável ou objeto.
66	`    plt.subplot(3,6, idx +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

Resultados gerados nesta célula

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	`import os`	Importa bibliotecas/módulos necessários.
2	`import torch`	Importa bibliotecas/módulos necessários.
3	`from PIL import Image`	Importa bibliotecas/módulos necessários.
4	`import matplotlib.pyplot as plt`	Importa bibliotecas/módulos necessários.
5	`from torchvision import models, transforms`	Importa bibliotecas/módulos necessários.
6	`from torchcam.methods import GradCAM`	Importa bibliotecas/módulos necessários.
7	`from torchcam.utils import overlay_mask`	Importa bibliotecas/módulos necessários.
8	`from torchvision.transforms.functional import to_pil_image`	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 = 'camaroes_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	`model.load_state_dict(torch.load(modelo_path, map_location='cpu'))`	Atribuição: define/atualiza variável ou objeto.
28	`model.eval()`	Coloca o modelo em modo de avaliação (desativa dropout etc.).
29		Linha em branco; separa blocos lógicos.
30	`plt.figure(figsize=(15, 6))`	Comando de plotagem/ajuste de figura (Matplotlib).
31	`cont = 0`	Atribuição: define/atualiza variável ou objeto.
32	`for idx, classe in enumerate(classes):`	Inicia um laço de iteração.
33	`    pasta_classe = os.path.join(caminho_dataset, classe)`	Atribuição: define/atualiza variável ou objeto.
34	`    img_nome = os.listdir(pasta_classe)[0]`	Atribuição: define/atualiza variável ou objeto.
35		Linha em branco; separa blocos lógicos.
36	`    if not img_nome.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):`	Estrutura condicional (controle de fluxo).
37	`        continue`	Instrução do script no contexto do bloco.
38		Linha em branco; separa blocos lógicos.
39	`    # === CARREGAR IMAGEM ===`	Comentário; documenta a intenção do bloco (não executa).
40	`    img_path = os.path.join(pasta_classe, img_nome)`	Atribuição: define/atualiza variável ou objeto.
41	`    img = Image.open(img_path).convert('RGB')`	Atribuição: define/atualiza variável ou objeto.
42		Linha em branco; separa blocos lógicos.
43	`    img_tensor = transform(img).unsqueeze(0)`	Atribuição: define/atualiza variável ou objeto.
44		Linha em branco; separa blocos lógicos.
45	`    output = model(img_tensor)`	Atribuição: define/atualiza variável ou objeto.

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

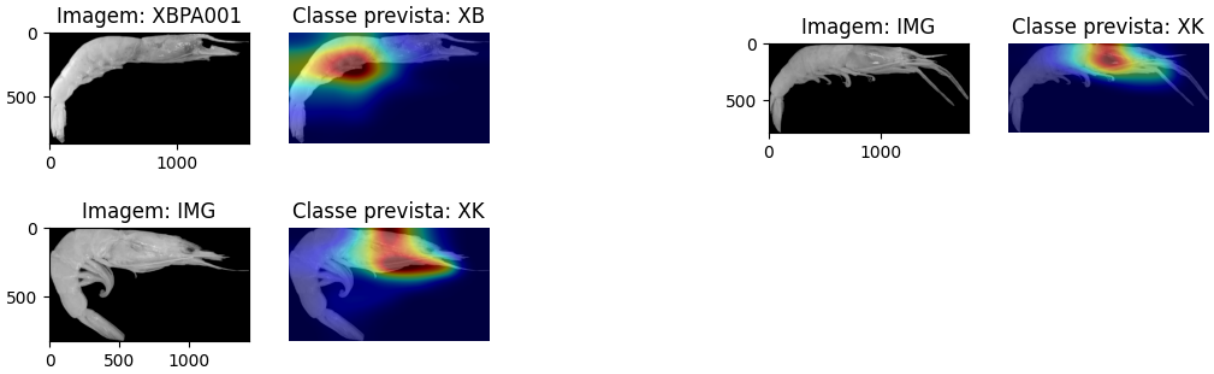
Linha	Código	Explicação
65	`    img_nome = img_nome.split('_')[0]`	Atribuição: define/atualiza variável ou objeto.
66	`    cont+=1`	Atribuição: define/atualiza variável ou objeto.
67	`    plt.subplot(3,6, idx + cont)`	Comando de plotagem/ajuste de figura (Matplotlib).
68	`    plt.title(f"Imagem: {img_nome}")`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

Resultados gerados nesta célula

Saída (texto)

```
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).