```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.
import kagglehub
jawadali1045_20k_multi_class_crop_disease_images_path = kagglehub.dataset_download('jawadali1045/20k-multi-class-crop-disease-images')

print('Data source import complete.')
```

```
Data source import complete.
```

```
jawadali1045_20k_multi_class_crop_disease_images_path
```

```
'/root/.cache/kagglehub/datasets/jawadali1045/20k-multi-class-crop-disease-images/versions/1'
```

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save &
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
!rm -rf /kaggle/working/*
```

```
!pip install imagehash
!pip install kagglehub
!pip install kaggle
!pip install torchinfo
!pip install seaborn
```

```
Requirement already satisfied: imagehash in /usr/local/lib/python3.11/dist-packages (4.3.2)
Requirement already satisfied: PyWavelets in /usr/local/lib/python3.11/dist-packages (from imagehash) (1.8.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from imagehash) (2.0.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from imagehash) (11.1.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from imagehash) (1.14.1)
Requirement already satisfied: kagglehub in /usr/local/lib/python3.11/dist-packages (0.3.10)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from kagglehub) (24.2)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages (from kagglehub) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from kagglehub) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from kagglehub) (4.67.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (2025.1.31)
Requirement already satisfied: kaggle in /usr/local/lib/python3.11/dist-packages (1.7.4.2)
Requirement already satisfied: bleach in /usr/local/lib/python3.11/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2025.1.31)
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.4.1)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.10)
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packages (from kaggle) (5.29.4)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.11/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.11/dist-packages (from kaggle) (75.1.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from kaggle) (4.67.1)
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.3.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from kaggle) (0.5.1)
Requirement already satisfied: torchinfo in /usr/local/lib/python3.11/dist-packages (1.8.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.0.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.11/dist-packages (from seaborn) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.5
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->
```

```
import os
import cv2
import imagehash
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from PIL import Image
from concurrent.futures import ProcessPoolExecutor

import torch
import torch.nn as nn
import torch.nn.functional as F
from torchinfo import summary
from torchvision import transforms, models, datasets
from torch.utils.data import Dataset, DataLoader, random_split
from torch.optim.lr_scheduler import ReduceLROnPlateau

from sklearn.metrics import recall_score, f1_score, roc_auc_score, confusion_matrix
from sklearn.preprocessing import label_binarize
```

```
corrupted = []
df_list = []

root_dir = '/kaggle/working/combined_dataset'
```

```
import os
import shutil

# Path ke direktori dataset Train dan Validation
train_dir = '/root/.cache/kagglehub/datasets/jawadali1045/20k-multi-class-crop-disease-images/versions/1/Train'
validation_dir = '/root/.cache/kagglehub/datasets/jawadali1045/20k-multi-class-crop-disease-images/versions/1/Validation'

# Folder tujuan untuk menyimpan semua gambar gabungan
destination_base_folder = '/kaggle/working/combined_dataset'

# Buat folder tujuan jika belum ada
os.makedirs(destination_base_folder, exist_ok=True)

# Fungsi untuk membaca gambar dari direktori dan menyalinnya ke folder tujuan dengan struktur yang sama
def copy_files_with_structure(source_directory, destination_base_folder):
    for root, dirs, files in os.walk(source_directory):
        for file in files:
            if file.endswith(('png', 'jpg', 'jpeg')):  # Filter file gambar
                source_file_path = os.path.join(root, file)
                relative_path = os.path.relpath(root, source_directory)
                destination_folder = os.path.join(destination_base_folder, relative_path)
                destination_file_path = os.path.join(destination_folder, file)

                # Buat folder tujuan jika belum ada
                os.makedirs(destination_folder, exist_ok=True)

                # Salin file
                shutil.copy2(source_file_path, destination_file_path)
                print(f"File {file} berhasil disalin ke {destination_file_path}.")

# Salin gambar dari folder Train dan Validation ke folder tujuan dengan struktur yang sama
copy_files_with_structure(train_dir, destination_base_folder)
copy_files_with_structure(validation_dir, destination_base_folder)
```

Show hidden output

```
def process_image(image_path, animal):
    if not os.path.isfile(image_path):
        return [image_path, True, None, None, None, None, None, animal]

    img = cv2.imread(image_path)
    if img is None:
        return [image_path, True, None, None, None, None, None, animal]

    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img_pil = Image.fromarray(img_rgb)
    img_hash = imagehash.phash(img_pil)
    width, height, channels = img.shape
    image_format = os.path.splitext(image_path)[-1]

    return [image_path, False, img_hash, image_format, width, height, channels, animal]
```

```
def process_animal(animal_dir, animal):
    image_paths = [os.path.join(animal_dir, image) for image in os.listdir(animal_dir) if os.path.isfile(os.path.join(animal_dir, image))]

    with ProcessPoolExecutor() as executor:
        results = executor.map(process_image, image_paths, [animal] * len(image_paths))

    valid_results = []
    for result in results:
        if result:
            if result[1]:
                corrupted.append(result[0])
            valid_results.append(result)
    return valid_results
```

```
for animal in os.listdir(root_dir):
    animal_dir = os.path.join(root_dir, animal)
    if os.path.isdir(animal_dir):
        results = process_animal(animal_dir, animal)
        df_list.extend(results)
```

```
print(f'Total corrupted images: {len(corrupted)}\n'
      f'Corrupted Images:\n{corrupted}')
```

```
Total corrupted images: 0
Corrupted Images:
[]
```

```
df = pd.DataFrame(columns=['image_path', 'corrupted', 'image_hash', 'image_format', 'width', 'height', 'channels', 'label'], data = df_list)
df.head()
```

	image_path	corrupted	image_hash	image_format	width	height	channels	label
0	/kaggle/working/combined_dataset/bacterial_bli...	False	cfd82c1e7da84551	.jpg	428	571	3	bacterial_blight in Cotton
1	/kaggle/working/combined_dataset/bacterial_bli...	False	acc8e8e4d985d955	.png	173	153	3	bacterial_blight in Cotton
2	/kaggle/working/combined_dataset/bacterial_bli...	False	f178dea569c32189	.png	165	220	3	bacterial_blight in Cotton
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	bacterial_blight in

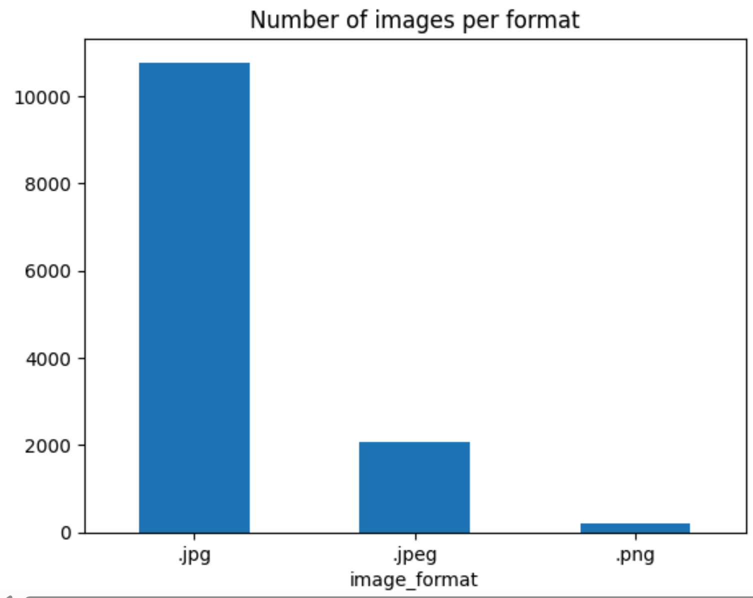
Next steps: Generate code with df View recommended plots New interactive sheet

```
df.describe().loc[['mean', 'std', 'min', 'max']]
```

	width	height	channels
mean	490.144291	479.341609	3.0
std	412.816607	417.908929	0.0
min	31.000000	36.000000	3.0
max	5504.000000	8256.000000	3.0

```
df['image_format'].value_counts().plot(kind='bar', title='Number of images per format')
plt.xticks(rotation=0)
```

```
(array([0, 1, 2]),
 [Text(0, 0, '.jpg'), Text(1, 0, '.jpeg'), Text(2, 0, '.png')])
```



```
class CONFIG:
    IMAGE_HEIGHT = 224
    IMAGE_WIDTH = 224
    CHANNELS = 3

    DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    N_CLASSES = df['label'].nunique()

    BATCH_SIZE = 16
    N_EPOCHS = 50
    PATIENCE_EPOCHS = 10
    LR = 0.001
    L2 = 0.0001
    DROPOUT = 0.3

    PATIENCE_SCHEDULER = 5

    MIN_VALUE_ACCURACY = 80
```

```
transform = transforms.Compose([
    transforms.Resize((CONFIG.IMAGE_HEIGHT, CONFIG.IMAGE_WIDTH)),
    transforms.ToTensor(),
    transforms.ColorJitter(brightness=0.3, contrast=0.2, saturation=0.2),
    transforms.RandomResizedCrop(224, scale=(0.8, 1.0), ratio=(0.9, 1.1)),
    transforms.RandomVerticalFlip(p=0.5),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

```
dataset = datasets.ImageFolder(root=root_dir, transform=transform)

train_size = int(0.7 * len(dataset))
val_size = int(0.15 * len(dataset))
test_size = len(dataset) - train_size - val_size
train_dataset, val_dataset, test_dataset = random_split(dataset, [train_size, val_size, test_size])
```

```
train_loader = DataLoader(train_dataset, batch_size=CONFIG.BATCH_SIZE, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=CONFIG.BATCH_SIZE, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=CONFIG.BATCH_SIZE, shuffle=False)
```

```
model = models.efficientnet_b2(weights=models.EfficientNet_B2_Weights.DEFAULT)
model.classifier = nn.Sequential(
    nn.Dropout(
        p=CONFIG.DROPOUT,
        inplace=True
        ),
    nn.Linear(
        model.classifier[1].in_features,
        CONFIG.N_CLASSES,
        bias=True
        )
    )
```

```
Downloading: "https://download.pytorch.org/models/efficientnet_b2_rwightman-c35c1473.pth" to /root/.cache/torch/hub/checkpoints/efficier
100%|██████████| 35.2M/35.2M [00:00<00:00, 156MB/s]
```

```
efficientnet = models.efficientnet_b2(weights=models.EfficientNet_B2_Weights.DEFAULT)

print(f'Default: {efficientnet.classifier}\n\n CustomModel: {model.classifier}')
```

```
Default: Sequential(
  (0): Dropout(p=0.3, inplace=True)
  (1): Linear(in_features=1408, out_features=1000, bias=True)
)

 CustomModel: Sequential(
  (0): Dropout(p=0.3, inplace=True)
  (1): Linear(in_features=1408, out_features=44, bias=True)
)
```

```
summary(model, input_size=(CONFIG.BATCH_SIZE, CONFIG.CHANNELS, CONFIG.IMAGE_HEIGHT, CONFIG.IMAGE_WIDTH))
```

```
==========================================================================================================
Layer (type:depth-idx)                                  Output Shape              Param #
==========================================================================================================
EfficientNet                                            [16, 44]                  --
├─Sequential: 1-1                                       [16, 1408, 7, 7]          --
│    └─Conv2dNormActivation: 2-1                        [16, 32, 112, 112]        --
│    │    └─Conv2d: 3-1                                 [16, 32, 112, 112]        864
│    │    └─BatchNorm2d: 3-2                            [16, 32, 112, 112]        64
│    │    └─SiLU: 3-3                                   [16, 32, 112, 112]        --
│    └─Sequential: 2-2                                  [16, 16, 112, 112]        --
│    │    └─MBConv: 3-4                                 [16, 16, 112, 112]        1,448
│    │    └─MBConv: 3-5                                 [16, 16, 112, 112]        612
│    └─Sequential: 2-3                                  [16, 24, 56, 56]          --
│    │    └─MBConv: 3-6                                 [16, 24, 56, 56]          6,004
│    │    └─MBConv: 3-7                                 [16, 24, 56, 56]          10,710
│    │    └─MBConv: 3-8                                 [16, 24, 56, 56]          10,710
│    └─Sequential: 2-4                                  [16, 48, 28, 28]          --
│    │    └─MBConv: 3-9                                 [16, 48, 28, 28]          16,518
│    │    └─MBConv: 3-10                                [16, 48, 28, 28]          43,308
│    │    └─MBConv: 3-11                                [16, 48, 28, 28]          43,308
│    └─Sequential: 2-5                                  [16, 88, 14, 14]          --
│    │    └─MBConv: 3-12                                [16, 88, 14, 14]          50,300
│    │    └─MBConv: 3-13                                [16, 88, 14, 14]          123,750
│    │    └─MBConv: 3-14                                [16, 88, 14, 14]          123,750
│    │    └─MBConv: 3-15                                [16, 88, 14, 14]          123,750
│    └─Sequential: 2-6                                  [16, 120, 14, 14]         --
│    │    └─MBConv: 3-16                                [16, 120, 14, 14]         149,158
│    │    └─MBConv: 3-17                                [16, 120, 14, 14]         237,870
│    │    └─MBConv: 3-18                                [16, 120, 14, 14]         237,870
│    │    └─MBConv: 3-19                                [16, 120, 14, 14]         237,870
│    └─Sequential: 2-7                                  [16, 208, 7, 7]           --
│    │    └─MBConv: 3-20                                [16, 208, 7, 7]           301,406
│    │    └─MBConv: 3-21                                [16, 208, 7, 7]           686,868
│    │    └─MBConv: 3-22                                [16, 208, 7, 7]           686,868
│    │    └─MBConv: 3-23                                [16, 208, 7, 7]           686,868
│    │    └─MBConv: 3-24                                [16, 208, 7, 7]           686,868
│    └─Sequential: 2-8                                  [16, 352, 7, 7]           --
│    │    └─MBConv: 3-25                                [16, 352, 7, 7]           846,900
│    │    └─MBConv: 3-26                                [16, 352, 7, 7]           1,888,920
│    └─Conv2dNormActivation: 2-9                        [16, 1408, 7, 7]          --
│    │    └─Conv2d: 3-27                                [16, 1408, 7, 7]          495,616
│    │    └─BatchNorm2d: 3-28                           [16, 1408, 7, 7]          2,816
│    │    └─SiLU: 3-29                                  [16, 1408, 7, 7]          --
├─AdaptiveAvgPool2d: 1-2                                [16, 1408, 1, 1]          --
├─Sequential: 1-3                                       [16, 44]                  --
│    └─Dropout: 2-10                                    [16, 1408]                --
│    └─Linear: 2-11                                     [16, 44]                  61,996
==========================================================================================================
Total params: 7,762,990
Trainable params: 7,762,990
Non-trainable params: 0
```

```
Total mult-adds (Units.GIGABYTES): 10.52
==========================================================================================
Input size (MB): 9.63
Forward/backward pass size (MB): 2508.77
Params size (MB): 31.05
Estimated Total Size (MB): 2549.46
==========================================================================================
```

```
class EarlyStopping:
    def __init__(self, patience=0, verbose=False, min_delta=0, path="checkpoint.pth", min_val_acc=0):
        self.patience = patience
        self.verbose = verbose
        self.min_delta = min_delta
        self.path = path
        self.min_val_acc = min_val_acc
        self.counter = 0
        self.best_score = None
        self.early_stop = False

    def __call__(self, model, optimizer, scheduler, train_losses, val_losses, train_accuracies, val_accuracies, learning_rate, l2_norm, epoc
        if val_accuracies[-1] < self.min_val_acc:
            if self.verbose:
                print(f"Validation accuracy {val_accuracies[-1]:.4f} did not reach minimum {self.min_val_acc:.4f}. Not saving.")
            return

        score = val_accuracies[-1]

        if self.best_score is None:
            self.best_score = score
            self.save_checkpoint(model, optimizer, scheduler, train_losses, val_losses, train_accuracies, val_accuracies, learning_rate, l2_
        elif score < self.best_score + self.min_delta:
            self.counter += 1
            if self.verbose:
                print(f"EarlyStopping counter: {self.counter} out of {self.patience}")
            if self.counter >= self.patience:
                self.early_stop = True
        else:
            self.best_score = score
            self.save_checkpoint(model, optimizer, scheduler, train_losses, val_losses, train_accuracies, val_accuracies, learning_rate, l2_
            self.counter = 0

    def save_checkpoint(self, model, optimizer, scheduler, train_losses, val_losses, train_accuracies, val_accuracies, learning_rate, l2_nor
        if self.verbose:
            print(f"Validation accuracy improved. Saving model with val_loss {val_losses[-1]:.6f}...")

        training_state = {
            "model_state_dict": model.state_dict(),
            "optimizer_state_dict": optimizer.state_dict(),
            "scheduler_state_dict": scheduler.state_dict() if scheduler else None,
            "train_losses": train_losses,
            "val_losses": val_losses,
            "train_accuracies": train_accuracies,
            "val_accuracies": val_accuracies,
            "hyperparameters": {
                "learning_rate": learning_rate,
                "L2 norm": l2_norm,
                "batch_size": 32,
                "epochs": epochs,
            }
        }
        torch.save(training_state, self.path)
```

```
criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=CONFIG.LR, weight_decay=CONFIG.L2)

scheduler = ReduceLROnPlateau(optimizer, mode='min', factor=0.1, patience=CONFIG.PATIENCE_SCHEDULER)
```

```
def train_model(model, data_loader, criterion, optimizer, device):
    model.train()
    train_loss, train_correct, train_total = 0, 0, 0
    for inputs, labels in data_loader:
        inputs, labels = inputs.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        train_loss += loss.item()
        _, predicted = torch.max(outputs.data, 1)
        train_correct += (predicted == labels).sum().item()
        train_total += labels.size(0)

    avg_loss = train_loss / len(data_loader)
    train_accuracy = 100 * train_correct / train_total

    return avg_loss, train_accuracy
```

```
def evaluate_model(model, data_loader, criterion, device):
    model.eval()
    val_loss, val_correct, val_total = 0, 0, 0
    with torch.no_grad():
        for inputs, labels in data_loader:
            inputs, labels = inputs.to(device), labels.to(device)

            outputs = model(inputs)
            loss = criterion(outputs, labels)
            val_loss += loss.item()

            _, predicted = torch.max(outputs.data, 1)
            val_correct += (predicted == labels).sum().item()
            val_total += labels.size(0)

    avg_loss = val_loss / len(data_loader)
    val_accuracy = 100 * val_correct / val_total

    scheduler.step(val_loss)
    current_lr = scheduler.get_last_lr()[0]

    return avg_loss, val_accuracy, current_lr
```

```
train_accuracies = []
val_accuracies = []
train_losses = []
val_losses = []

early_stopping = EarlyStopping(patience=CONFIG.PATIENCE_EPOCHS, verbose=True, min_val_acc=CONFIG.MIN_VALUE_ACCURACY, path="best_model.pth")

for epoch in range(CONFIG.N_EPOCHS):
    train_loss, train_accuracy = train_model(model, train_loader, criterion, optimizer, CONFIG.DEVICE)
    val_loss, val_accuracy, current_lr = evaluate_model(model, val_loader, criterion, CONFIG.DEVICE)

    train_losses.append(train_loss)
    val_losses.append(val_loss)
    train_accuracies.append(train_accuracy)
    val_accuracies.append(val_accuracy)

    print(f"Epoch {epoch+1}/{CONFIG.N_EPOCHS}")
    print(f"Train Loss: {train_loss:.4f} | Train Accuracy: {train_accuracy:.2f}%")
    print(f"Validation Loss: {val_loss:.4f} | Validation Accuracy: {val_accuracy:.2f}%")
    print(f"Learning Rate: {current_lr}")

    early_stopping(model, optimizer, scheduler, train_losses, val_losses, train_accuracies, val_accuracies, CONFIG.LR, CONFIG.L2, CONFIG.N_EP
    if early_stopping.early_stop:
        print("Early stopping activated. Stopping training.")
        break
```

```
/usr/local/lib/python3.11/dist-packages/PIL/TiffImagePlugin.py:949: UserWarning: Corrupt EXIF data.  Expecting to read 4 bytes but only
  warnings.warn(str(msg))
/usr/local/lib/python3.11/dist-packages/PIL/Image.py:1045: UserWarning: Palette images with Transparency expressed in bytes should be cc
  warnings.warn(
Epoch 1/50
Train Loss: 1.0619 | Train Accuracy: 72.08%
Validation Loss: 0.6445 | Validation Accuracy: 83.33%
Learning Rate: 0.001
Validation accuracy improved. Saving model with val_loss 0.644461...
Epoch 2/50
Train Loss: 0.5673 | Train Accuracy: 84.01%
Validation Loss: 0.4556 | Validation Accuracy: 88.60%
Learning Rate: 0.001
Validation accuracy improved. Saving model with val_loss 0.455553...
Epoch 3/50
Train Loss: 0.4548 | Train Accuracy: 87.08%
Validation Loss: 0.4897 | Validation Accuracy: 87.18%
Learning Rate: 0.001
EarlyStopping counter: 1 out of 10
Epoch 4/50
Train Loss: 0.3990 | Train Accuracy: 88.03%
Validation Loss: 0.4304 | Validation Accuracy: 88.70%
Learning Rate: 0.001
Validation accuracy improved. Saving model with val_loss 0.430402...
Epoch 5/50
Train Loss: 0.3911 | Train Accuracy: 88.57%
Validation Loss: 0.3816 | Validation Accuracy: 89.31%
Learning Rate: 0.001
Validation accuracy improved. Saving model with val_loss 0.381586...
Epoch 6/50
Train Loss: 0.3518 | Train Accuracy: 89.68%
Validation Loss: 0.4105 | Validation Accuracy: 89.51%
Learning Rate: 0.001
Validation accuracy improved. Saving model with val_loss 0.410536...
Epoch 7/50
Train Loss: 0.3065 | Train Accuracy: 90.99%
Validation Loss: 0.3753 | Validation Accuracy: 90.43%
Learning Rate: 0.001
Validation accuracy improved. Saving model with val_loss 0.375293...
Epoch 8/50
Train Loss: 0.3160 | Train Accuracy: 90.30%
Validation Loss: 0.3539 | Validation Accuracy: 91.03%
Learning Rate: 0.001
Validation accuracy improved. Saving model with val_loss 0.353915...
Epoch 9/50
Train Loss: 0.3065 | Train Accuracy: 90.97%
Validation Loss: 0.3364 | Validation Accuracy: 91.69%
Learning Rate: 0.001
Validation accuracy improved. Saving model with val_loss 0.336401...
Epoch 10/50
Train Loss: 0.2983 | Train Accuracy: 91.04%
Validation Loss: 0.4675 | Validation Accuracy: 88.15%
Learning Rate: 0.001
EarlyStopping counter: 1 out of 10
Epoch 11/50
Train Loss: 0.2914 | Train Accuracy: 91.36%
Validation Loss: 0.3332 | Validation Accuracy: 90.78%
Learning Rate: 0.001
EarlyStopping counter: 2 out of 10
Epoch 12/50
Train Loss: 0.2676 | Train Accuracy: 92.00%
Validation Loss: 0.3602 | Validation Accuracy: 90.17%
Learning Rate: 0.001
EarlyStopping counter: 3 out of 10
Epoch 13/50
Train Loss: 0.2618 | Train Accuracy: 92.02%
Validation Loss: 0.3600 | Validation Accuracy: 91.19%
Learning Rate: 0.001
EarlyStopping counter: 4 out of 10
Epoch 14/50
Train Loss: 0.2453 | Train Accuracy: 92.46%
Validation Loss: 0.3577 | Validation Accuracy: 90.93%
Learning Rate: 0.001
EarlyStopping counter: 5 out of 10
Epoch 15/50
Train Loss: 0.2547 | Train Accuracy: 92.31%
Validation Loss: 0.3342 | Validation Accuracy: 92.10%
Learning Rate: 0.001
Validation accuracy improved. Saving model with val_loss 0.334246...
Epoch 16/50
Train Loss: 0.2327 | Train Accuracy: 92.93%
Validation Loss: 0.4234 | Validation Accuracy: 89.46%
Learning Rate: 0.001
EarlyStopping counter: 1 out of 10
```

```
Epoch 17/50
Train Loss: 0.2325 | Train Accuracy: 93.09%
Validation Loss: 0.3466 | Validation Accuracy: 91.24%
Learning Rate: 0.0001
EarlyStopping counter: 2 out of 10
Epoch 18/50
Train Loss: 0.1230 | Train Accuracy: 96.35%
Validation Loss: 0.2337 | Validation Accuracy: 93.92%
Learning Rate: 0.0001
Validation accuracy improved. Saving model with val_loss 0.233669...
Epoch 19/50
Train Loss: 0.0793 | Train Accuracy: 97.70%
Validation Loss: 0.2261 | Validation Accuracy: 94.38%
Learning Rate: 0.0001
Validation accuracy improved. Saving model with val_loss 0.226085...
Epoch 20/50
Train Loss: 0.0683 | Train Accuracy: 97.89%
Validation Loss: 0.2244 | Validation Accuracy: 94.58%
Learning Rate: 0.0001
Validation accuracy improved. Saving model with val_loss 0.224373...
Epoch 21/50
Train Loss: 0.0549 | Train Accuracy: 98.33%
Validation Loss: 0.2199 | Validation Accuracy: 94.83%
Learning Rate: 0.0001
Validation accuracy improved. Saving model with val_loss 0.219881...
Epoch 22/50
Train Loss: 0.0503 | Train Accuracy: 98.50%
Validation Loss: 0.2300 | Validation Accuracy: 95.04%
Learning Rate: 0.0001
Validation accuracy improved. Saving model with val_loss 0.230044...
Epoch 23/50
Train Loss: 0.0422 | Train Accuracy: 98.73%
Validation Loss: 0.2298 | Validation Accuracy: 94.83%
Learning Rate: 0.0001
EarlyStopping counter: 1 out of 10
Epoch 24/50
Train Loss: 0.0439 | Train Accuracy: 98.70%
Validation Loss: 0.2296 | Validation Accuracy: 94.93%
Learning Rate: 0.0001
EarlyStopping counter: 2 out of 10
Epoch 25/50
Train Loss: 0.0372 | Train Accuracy: 98.89%
Validation Loss: 0.2503 | Validation Accuracy: 95.09%
Learning Rate: 0.0001
Validation accuracy improved. Saving model with val_loss 0.250314...
Epoch 26/50
Train Loss: 0.0348 | Train Accuracy: 98.85%
Validation Loss: 0.2339 | Validation Accuracy: 95.19%
Learning Rate: 0.0001
Validation accuracy improved. Saving model with val_loss 0.233914...
Epoch 27/50
Train Loss: 0.0358 | Train Accuracy: 98.89%
Validation Loss: 0.2723 | Validation Accuracy: 94.83%
Learning Rate: 1e-05
EarlyStopping counter: 1 out of 10
Epoch 28/50
Train Loss: 0.0301 | Train Accuracy: 99.17%
Validation Loss: 0.2388 | Validation Accuracy: 94.68%
Learning Rate: 1e-05
EarlyStopping counter: 2 out of 10
Epoch 29/50
Train Loss: 0.0305 | Train Accuracy: 99.03%
Validation Loss: 0.2595 | Validation Accuracy: 95.04%
Learning Rate: 1e-05
EarlyStopping counter: 3 out of 10
Epoch 30/50
Train Loss: 0.0265 | Train Accuracy: 99.16%
Validation Loss: 0.2587 | Validation Accuracy: 95.14%
Learning Rate: 1e-05
EarlyStopping counter: 4 out of 10
Epoch 31/50
Train Loss: 0.0273 | Train Accuracy: 99.25%
Validation Loss: 0.2528 | Validation Accuracy: 95.24%
Learning Rate: 1e-05
Validation accuracy improved. Saving model with val_loss 0.252793...
Epoch 32/50
Train Loss: 0.0259 | Train Accuracy: 99.27%
Validation Loss: 0.2502 | Validation Accuracy: 95.14%
Learning Rate: 1e-05
EarlyStopping counter: 1 out of 10
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-60-33650011ec85> in <cell line: 0>()
      7
      8 for epoch in range(CONFIG.N_EPOCHS):
```

```
----> 9     train_loss, train_accuracy = train_model(model, train_loader, criterion, optimizer, CONFIG.DEVICE)
     10     val_loss, val_accuracy, current_lr = evaluate_model(model, val_loader, criterion, CONFIG.DEVICE)
     11
```

14 frames

```
/usr/local/lib/python3.11/dist-packages/torch/nn/functional.py in interpolate(input, size, scale_factor, mode, align_corners,
recompute_scale_factor, antialias)
   4676         assert align_corners is not None
   4677         if antialias:
-> 4678             return torch._C._nn._upsample_bilinear2d_aa(
   4679                 input, output_size, align_corners, scale_factors
   4680             )

KeyboardInterrupt:
```

```
import torch
from torchvision import models
import torch.nn as nn

# Load EfficientNet-B2 architecture without pre-trained weights
model = models.efficientnet_b2(weights=None)

# Modify classifier to match trained architecture
model.classifier = nn.Sequential(
    nn.Dropout(p=CONFIG.DROPOUT, inplace=True),
    nn.Linear(model.classifier[1].in_features, CONFIG.N_CLASSES, bias=True)
)

# Load the checkpoint and extract only the model state_dict
checkpoint = torch.load("best_model.pth", map_location=torch.device("cpu"))
model.load_state_dict(checkpoint["model_state_dict"])  # Correct way to load weights
model.eval()  # Set model to evaluation mode

print("Model successfully loaded and ready for inference!")
```

```
Model successfully loaded and ready for inference!
```

```
# Convert to TorchScript for optimized inference
scripted_model = torch.jit.script(model)
scripted_model.save("efficientnet_b2_scripted.pt")
```

```
# Load checkpoint
checkpoint = torch.load("best_model.pth", map_location=torch.device("cpu"))

# Get class-to-index mapping (if saved)
if "class_to_idx" in checkpoint:
    class_to_idx = checkpoint["class_to_idx"]
    idx_to_class = {v: k for k, v in class_to_idx.items()}  # Reverse mapping
    class_names = list(idx_to_class.values())
    print(class_names)
else:
    print("Class information not saved in the checkpoint.")
```

```
Class information not saved in the checkpoint.
```