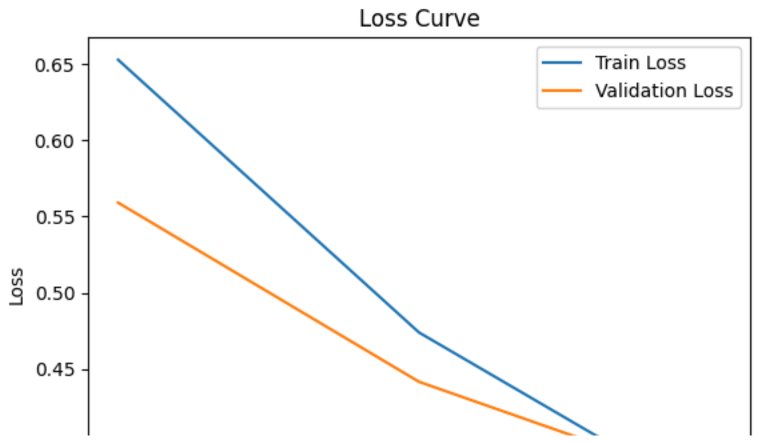
```
# Mount Google Drive and set paths
!pip install torchinfo
!pip install ptflops
from google.colab import drive
drive.mount('/content/drive')

import os
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, precision_score, recall_score, f1_score
from torchinfo import summary
from ptflops import get_model_complexity_info
import platform
import time
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
from itertools import product
import random

# Set dataset path
data_dir = '/content/drive/MyDrive/dataset'

# Transforms
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

train_dir = os.path.join(data_dir, 'train')
valid_dir = os.path.join(data_dir, 'valid')
test_dir = os.path.join(data_dir, 'test')

train_data = datasets.ImageFolder(train_dir, transform=transform)
valid_data = datasets.ImageFolder(valid_dir, transform=transform)
test_data = datasets.ImageFolder(test_dir, transform=transform)

# Dataloader for the validation set
valid_loader = DataLoader(valid_data, batch_size=32)

# Hyperparameter tuning configs
learning_rates = [0.001, 0.0005]
optimizers = ['adam', 'sgd']
batch_sizes = [32, 64]
epochs = 3

best_model = None
best_val_accuracy = 0
results = []
train_time = None
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

for lr, opt_type, batch_size in product(learning_rates, optimizers, batch_sizes):
    print(f"\nTraining with LR={lr}, Optimizer={opt_type}, Batch Size={batch_size}")

    train_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True)

    model = models.densenet121(pretrained=True)
    for param in model.parameters():
        param.requires_grad = False
    model.classifier = nn.Linear(model.classifier.in_features, 2)
    model = model.to(device)

    optimizer = optim.Adam(model.classifier.parameters(), lr=lr) if opt_type == 'adam' else optim.SGD(model.classifier.parameters(), lr=lr, m
    criterion = nn.CrossEntropyLoss()

    train_losses, val_losses = [], []
    train_accuracies = []
    start_time = time.time()

    best_val_loss = float('inf')
    early_stop_counter = 0
```

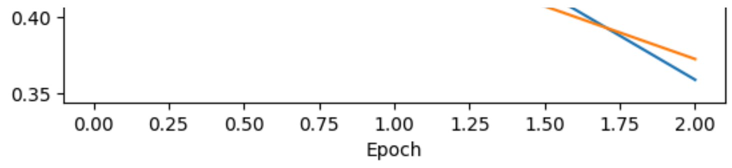
```
        patience = 2

        for epoch in range(epochs):
            model.train()
            train_loss, correct_train, total_train = 0, 0, 0
            for inputs, labels in train_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, 1)
                correct_train += (predicted == labels).sum().item()
                total_train += labels.size(0)

            avg_train_loss = train_loss / len(train_loader)
            train_losses.append(avg_train_loss)
            train_accuracy = 100 * correct_train / total_train
            train_accuracies.append(train_accuracy)

            model.eval()
            val_loss, correct, total = 0, 0, 0
            with torch.no_grad():
                for inputs, labels in valid_loader:
                    inputs, labels = inputs.to(device), labels.to(device)
                    outputs = model(inputs)
                    loss = criterion(outputs, labels)
                    val_loss += loss.item()
                    _, predicted = torch.max(outputs, 1)
                    correct += (predicted == labels).sum().item()
                    total += labels.size(0)

            avg_val_loss = val_loss / len(valid_loader)
            val_losses.append(avg_val_loss)
            val_accuracy = 100 * correct / total

            print(f"Epoch {epoch+1}: Train Acc={train_accuracy:.2f}%, Train Loss={avg_train_loss:.4f}, Val Loss={avg_val_loss:.4f}, Val Acc={val_

            if avg_val_loss < best_val_loss:
                best_val_loss = avg_val_loss
                early_stop_counter = 0
            else:
                early_stop_counter += 1
                if early_stop_counter >= patience:
                    print("Early stopping triggered.")
                    break

        if val_accuracy > best_val_accuracy:
            best_val_accuracy = val_accuracy
            best_model = model

        train_time = time.time() - start_time
        results.append((lr, opt_type, batch_size, val_accuracy, train_accuracy))

    # Plot Loss Curve
    plt.plot(train_losses, label='Train Loss')
    plt.plot(val_losses, label='Validation Loss')
    plt.legend()
    plt.title("Loss Curve")
    plt.xlabel("Epoch")
    plt.ylabel("Loss")
    plt.show()

    # Plot Accuracy Curve
    plt.plot(train_accuracies, label='Train Accuracy')
    plt.title("Train Accuracy Curve")
    plt.xlabel("Epoch")
    plt.ylabel("Accuracy")
    plt.legend()
    plt.show()

    print("\n=== Hyperparameter Tuning Summary ===")
    for r in results:
        print(f"LR: {r[0]}, Optimizer: {r[1]}, Batch Size: {r[2]}, Val Accuracy: {r[3]:.2f}%, Train Accuracy: {r[4]:.2f}%")
```

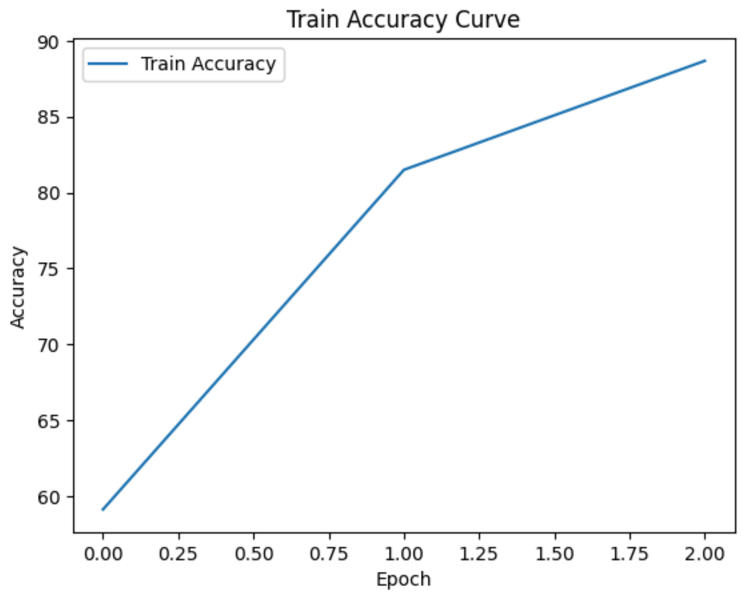
```
# ---------- Final Evaluation on Test Set ---------- #
model = best_model
model.eval()
correct, total = 0, 0
all_preds, all_labels, all_probs = [], [], []
start_test_time = time.time()

with torch.no_grad():
    for inputs, labels in DataLoader(test_data, batch_size=32):
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = model(inputs)
        probs = torch.nn.functional.softmax(outputs, dim=1)
        _, predicted = probs.max(1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
        all_preds.extend(predicted.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())
        all_probs.extend(probs[:, 1].cpu().numpy())

print(f"\nTest Accuracy: {100 * correct / total:.2f}%")

# ---------- Metrics ---------- #
precision = precision_score(all_labels, all_preds, average='macro')
recall = recall_score(all_labels, all_preds, average='macro')
f1 = f1_score(all_labels, all_preds, average='macro')
roc_auc = roc_auc_score(all_labels, all_probs)

print("\n📊 Final Test Performance:")
print(f"✓ Precision: {precision:.2f}")
print(f"✓ Recall: {recall:.2f}")
print(f"✓ F1 Score: {f1:.2f}")
print(f"✓ ROC-AUC: {roc_auc:.2f}")

print("\nClassification Report:\n", classification_report(all_labels, all_preds, target_names=train_data.classes))
print("Confusion Matrix:\n", confusion_matrix(all_labels, all_preds))

# ---------- Computational Parameters ---------- #
print("\n--- Model Info ---")
total_params = sum(p.numel() for p in model.parameters())
trainable_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print(f"Total Parameters: {total_params}")
print(f"Trainable Parameters: {trainable_params}")

summary(model, input_size=(1, 3, 224, 224))
with torch.cuda.device(0):
    macs, params = get_model_complexity_info(model, (3, 224, 224), as_strings=True, print_per_layer_stat=False)
    print(f"FLOPs: {macs}, Params: {params}")

print("Training Platform:", platform.platform())
print("CUDA Device:", torch.cuda.get_device_name(0) if torch.cuda.is_available() else "CPU")
print(f"Total Training Time: {train_time:.2f} seconds")
print(f"Inference Time on 1 image: {time.time() - start_test_time:.4f} seconds")

# ---------- Robustness Test (Noise and Lighting) ---------- #
transform_robust = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ColorJitter(brightness=0.5),
    transforms.ToTensor(),
    transforms.Lambda(lambda x: x + 0.05 * torch.randn_like(x)),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
robust_data = datasets.ImageFolder(test_dir, transform=transform_robust)
robust_loader = DataLoader(robust_data, batch_size=32)

correct, total = 0, 0
with torch.no_grad():
    for inputs, labels in robust_loader:
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = model(inputs)
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
print(f"Robustness (Noise & Lighting) Accuracy: {100 * correct / total:.2f}%")

# ---------- Error Analysis ---------- #
wrong_preds = [(img[0], p, l) for img, p, l in zip(test_data.imgs, all_preds, all_labels) if p != l]
```

```
for i in range(min(5, len(wrong_preds))):
    path, pred, label = wrong_preds[i]
    img = Image.open(path)
    plt.imshow(img)
    plt.title(f"True: {train_data.classes[label]}, Predicted: {train_data.classes[pred]}")
    plt.axis('off')
    plt.show()

# ---------- Save Model ---------- #
torch.save(model.state_dict(), '/content/drive/MyDrive/densenet121_best__model.pth')
```

```
Collecting torchinfo
  Downloading torchinfo-1.8.0-py3-none-any.whl.metadata (21 kB)
Downloading torchinfo-1.8.0-py3-none-any.whl (23 kB)
Installing collected packages: torchinfo
Successfully installed torchinfo-1.8.0
Collecting ptflops
  Downloading ptflops-0.7.4-py3-none-any.whl.metadata (9.4 kB)
Requirement already satisfied: torch>=2.0 in /usr/local/lib/python3.11/dist-packages (from ptflops) (2.6.0+cu124)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch>=2.0->ptflops) (3.18.0)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0->ptflops) (4.14.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=2.0->ptflops) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0->ptflops) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch>=2.0->ptflops) (2025.3.0)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=2.0->ptflops)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=2.0->ptflops)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=2.0->ptflops)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=2.0->ptflops)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=2.0->ptflops)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=2.0->ptflops)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=2.0->ptflops)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=2.0->ptflops)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=2.0->ptflops)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0->ptflops) (0.6.
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0->ptflops) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0->ptflops) (12.4.12
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=2.0->ptflops)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0->ptflops) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0->ptflops) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch>=2.0->ptflops) (
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch>=2.0->ptflops) (3.0.2)
Downloading ptflops-0.7.4-py3-none-any.whl (19 kB)
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 3.9 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 60.0 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 61.8 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 46.9 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.8 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.6 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 15.5 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.6 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.8 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 43.8 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-c
Mounted at /content/drive

Training with LR=0.001, Optimizer=adam, Batch Size=32
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/densenet121-a639ec97.pth" to /root/.cache/torch/hub/checkpoints/densenet121-a639ec97.p
100%|██████████| 30.8M/30.8M [00:00<00:00, 211MB/s]
Epoch 1: Train Acc=74.17%, Train Loss=0.5293, Val Loss=0.4046, Val Acc=83.52%
Epoch 2: Train Acc=91.57%, Train Loss=0.2789, Val Loss=0.2873, Val Acc=87.91%
Epoch 3: Train Acc=93.23%, Train Loss=0.2159, Val Loss=0.2563, Val Acc=87.91%

Training with LR=0.001, Optimizer=adam, Batch Size=64
Epoch 1: Train Acc=74.72%, Train Loss=0.5322, Val Loss=0.4399, Val Acc=85.71%
Epoch 2: Train Acc=90.47%, Train Loss=0.3249, Val Loss=0.3097, Val Acc=89.01%
Epoch 3: Train Acc=91.57%, Train Loss=0.2270, Val Loss=0.3141, Val Acc=91.21%

Training with LR=0.001, Optimizer=sgd, Batch Size=32
Epoch 1: Train Acc=72.38%, Train Loss=0.5351, Val Loss=0.3611, Val Acc=81.32%
Epoch 2: Train Acc=88.95%, Train Loss=0.2829, Val Loss=0.2540, Val Acc=91.21%
Epoch 3: Train Acc=92.96%, Train Loss=0.1991, Val Loss=0.2154, Val Acc=93.41%

Training with LR=0.001, Optimizer=sgd, Batch Size=64
Epoch 1: Train Acc=70.30%, Train Loss=0.5825, Val Loss=0.4598, Val Acc=83.52%
Epoch 2: Train Acc=87.15%, Train Loss=0.3620, Val Loss=0.3317, Val Acc=90.11%
Epoch 3: Train Acc=91.02%, Train Loss=0.2844, Val Loss=0.2912, Val Acc=92.31%

Training with LR=0.0005, Optimizer=adam, Batch Size=32
Epoch 1: Train Acc=72.51%, Train Loss=0.5596, Val Loss=0.4293, Val Acc=81.32%
Epoch 2: Train Acc=88.67%, Train Loss=0.3508, Val Loss=0.3148, Val Acc=91.21%
Epoch 3: Train Acc=91.16%, Train Loss=0.2902, Val Loss=0.2678, Val Acc=92.31%

Training with LR=0.0005, Optimizer=adam, Batch Size=64
Epoch 1: Train Acc=62.57%, Train Loss=0.6487, Val Loss=0.6297, Val Acc=74.73%
Epoch 2: Train Acc=81.91%, Train Loss=0.4596, Val Loss=0.4221, Val Acc=87.91%
Epoch 3: Train Acc=90.75%, Train Loss=0.3535, Val Loss=0.3731, Val Acc=89.01%

Training with LR=0.0005, Optimizer=sgd, Batch Size=32
Epoch 1: Train Acc=61.33%, Train Loss=0.6315, Val Loss=0.4532, Val Acc=85.71%
Epoch 2: Train Acc=88.67%, Train Loss=0.3486, Val Loss=0.3182, Val Acc=91.21%
Epoch 3: Train Acc=91.44%, Train Loss=0.2593, Val Loss=0.2713, Val Acc=93.41%

Training with LR=0.0005, Optimizer=sgd, Batch Size=64
Epoch 1: Train Acc=59.12%, Train Loss=0.6528, Val Loss=0.5591, Val Acc=74.73%
Epoch 2: Train Acc=81.49%, Train Loss=0.4739, Val Loss=0.4416, Val Acc=86.81%
Epoch 3: Train Acc=88.67%, Train Loss=0.3589, Val Loss=0.3724, Val Acc=90.11%
```





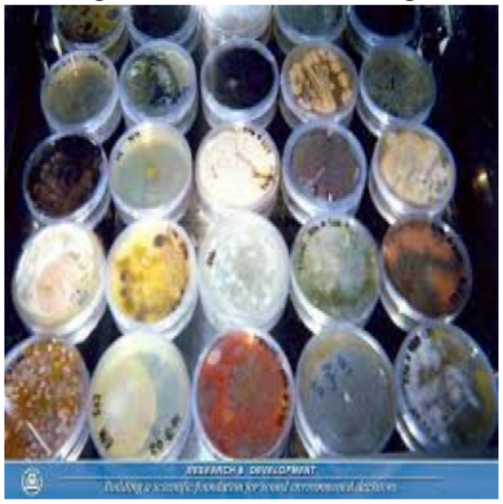


```
=== Hyperparameter Tuning Summary ===
LR: 0.001, Optimizer: adam, Batch Size: 32, Val Accuracy: 87.91%, Train Accuracy: 93.23%
LR: 0.001, Optimizer: adam, Batch Size: 64, Val Accuracy: 91.21%, Train Accuracy: 91.57%
LR: 0.001, Optimizer: sgd, Batch Size: 32, Val Accuracy: 93.41%, Train Accuracy: 92.96%
LR: 0.001, Optimizer: sgd, Batch Size: 64, Val Accuracy: 92.31%, Train Accuracy: 91.02%
LR: 0.0005, Optimizer: adam, Batch Size: 32, Val Accuracy: 92.31%, Train Accuracy: 91.16%
LR: 0.0005, Optimizer: adam, Batch Size: 64, Val Accuracy: 89.01%, Train Accuracy: 90.75%
LR: 0.0005, Optimizer: sgd, Batch Size: 32, Val Accuracy: 93.41%, Train Accuracy: 91.44%
LR: 0.0005, Optimizer: sgd, Batch Size: 64, Val Accuracy: 90.11%, Train Accuracy: 88.67%

Test Accuracy: 90.00%

📊 Final Test Performance:
✓ Precision: 0.91
✓ Recall: 0.90
✓ F1 Score: 0.90
✓ ROC-AUC: 0.97

Classification Report:
                precision    recall  f1-score   support

    Degradable       0.97      0.82      0.89        45
Non degradable       0.85      0.98      0.91        45

      accuracy                           0.90        90
     macro avg       0.91      0.90      0.90        90
  weighted avg       0.91      0.90      0.90        90

Confusion Matrix:
 [[37  8]
 [ 1 44]]

--- Model Info ---
Total Parameters: 6955906
Trainable Parameters: 2050
FLOPs: 2.9 GMac, Params: 2.05 k
Training Platform: Linux-6.1.123+-x86_64-with-glibc2.35
CUDA Device: Tesla T4
Total Training Time: 31.10 seconds
Inference Time on 1 image: 74.1569 seconds
Robustness (Noise & Lighting) Accuracy: 81.11%
```

True: Degradable, Predicted: Non degradable

True: Degradable, Predicted: Non degradable

True: Degradable, Predicted: Non degradable



True: Degradable, Predicted: Non degradable

True: Degradable, Predicted: Non degradable