import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import confusion\_matrix, precision\_score, recall\_score, f1\_score, accuracy\_score

from tensorflow.keras.applications import Xception, ResNet101, DenseNet121, EfficientNetB0, ResNet50

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.regularizers import l2

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import EarlyStopping

# Verilerin bulunduğu dizinler

train\_dir = '/content/drive/MyDrive/Cilt Hastalıkları Teşhisi Projesi. /train'

test\_dir = '/content/drive/MyDrive/Cilt Hastalıkları Teşhisi Projesi. /test'

IMG\_SIZE = 224

NUM\_CLASSES = 2

class\_labels = {class\_name: idx for idx, class\_name in enumerate(sorted(os.listdir(train\_dir))) if os.path.isdir(os.path.join(train\_dir, class\_name))}

class\_labels = {k: v - 1 for k, v in class\_labels.items()}

def load\_images\_from\_folder(folder):

images = []

labels = []

for label\_name in os.listdir(folder):

if label\_name.startswith('.'):

continue

class\_folder = os.path.join(folder, label\_name)

if os.path.isdir(class\_folder):

label = class\_labels[label\_name]

for filename in os.listdir(class\_folder):

img\_path = os.path.join(class\_folder, filename)

img = cv2.imread(img\_path)

if img is not None:

img = cv2.resize(img, (IMG\_SIZE, IMG\_SIZE))

images.append(img)

labels.append(label)

return np.array(images), np.array(labels)

X\_train, y\_train = load\_images\_from\_folder(train\_dir)

X\_test, y\_test = load\_images\_from\_folder(test\_dir)

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

y\_train\_cat = to\_categorical(y\_train, num\_classes=NUM\_CLASSES)

y\_test\_cat = to\_categorical(y\_test, num\_classes=NUM\_CLASSES)

# Data augmentation

train\_datagen = ImageDataGenerator(

rotation\_range=20,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True

)

train\_generator = train\_datagen.flow(X\_train, y\_train\_cat, batch\_size=32)

models = {

"Xception": Xception,

"ResNet101": ResNet101,

"DenseNet121": DenseNet121,

"EfficientNetB0": EfficientNetB0,

"ResNet50": ResNet50

}

results = {}

for model\_name, model\_func in models.items():

print(f"\n--- {model\_name} Modeli Eğitiliyor ---\n")

base\_model = model\_func(weights='imagenet', include\_top=False, input\_shape=(IMG\_SIZE, IMG\_SIZE, 3))

base\_model.trainable = False

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(128, activation='relu', kernel\_regularizer=l2(0.01))(x)

x = Dropout(0.6)(x)

predictions = Dense(NUM\_CLASSES, activation='softmax')(x)

model = Model(inputs=base\_model.input, outputs=predictions)

model.compile(optimizer=Adam(learning\_rate=1e-4), loss='categorical\_crossentropy', metrics=['accuracy'])

early\_stop = EarlyStopping(monitor='val\_loss', patience=10, restore\_best\_weights=True)

history = model.fit(

train\_generator,

epochs=100,

validation\_data=(X\_test, y\_test\_cat),

callbacks=[early\_stop],

verbose=1

)

# Fine-tuning

base\_model.trainable = True

model.compile(optimizer=Adam(learning\_rate=1e-5), loss='categorical\_crossentropy', metrics=['accuracy'])

fine\_tune\_history = model.fit(

train\_generator,

epochs=10,

validation\_data=(X\_test, y\_test\_cat),

callbacks=[early\_stop],

verbose=1

)

y\_pred = model.predict(X\_test)

y\_pred\_classes = np.argmax(y\_pred, axis=1)

conf\_matrix = confusion\_matrix(y\_test, y\_pred\_classes)

precision = precision\_score(y\_test, y\_pred\_classes, average='macro')

recall = recall\_score(y\_test, y\_pred\_classes, average='macro')

f1 = f1\_score(y\_test, y\_pred\_classes, average='macro')

accuracy = accuracy\_score(y\_test, y\_pred\_classes)

TN = conf\_matrix.sum() - (conf\_matrix.sum(axis=1) + conf\_matrix.sum(axis=0) - np.diag(conf\_matrix))

FP = conf\_matrix.sum(axis=0) - np.diag(conf\_matrix)

specificity = np.mean(TN / (TN + FP))

results[model\_name] = {

"accuracy": accuracy,

"precision": precision,

"recall": recall,

"f1\_score": f1,

"specificity": specificity

}

plt.plot(history.history['accuracy'], label='Eğitim Doğruluğu')

plt.plot(history.history['val\_accuracy'], label='Doğrulama Doğruluğu')

plt.xlabel('Epoch')

plt.ylabel('Doğruluk')

plt.legend()

plt.title(f'{model\_name} - Doğruluk Grafiği')

plt.show()

plt.plot(history.history['loss'], label='Eğitim Kaybı')

plt.plot(history.history['val\_loss'], label='Doğrulama Kaybı')

plt.xlabel('Epoch')

plt.ylabel('Kayıp')

plt.legend()

plt.title(f'{model\_name} - Kayıp Grafiği')

plt.show()

for model\_name, metrics in results.items():

print(f"\n{model\_name} Model Sonuçları:")

for metric\_name, value in metrics.items():

print(f"{metric\_name.capitalize()}: {value:.4f}")

--- Xception Modeli Eğitiliyor ---

Downloading data from <https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5>

**83683744/83683744** ━━━━━━━━━━━━━━━━━━━━ **0s** 0us/step

/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data\_adapters/py\_dataset\_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().\_\_init\_\_(\*\*kwargs)` in its constructor. `\*\*kwargs` can include `workers`, `use\_multiprocessing`, `max\_queue\_size`. Do not pass these arguments to `fit()`, as they will be ignored.

self.\_warn\_if\_super\_not\_called()

Epoch 1/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **52s** 788ms/step - accuracy: 0.8101 - loss: 2.7419 - val\_accuracy: 0.9086 - val\_loss: 2.3181

Epoch 2/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 437ms/step - accuracy: 0.9195 - loss: 2.2664 - val\_accuracy: 0.9608 - val\_loss: 1.9789

Epoch 3/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 441ms/step - accuracy: 0.9635 - loss: 1.9225 - val\_accuracy: 0.9843 - val\_loss: 1.6977

Epoch 4/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 465ms/step - accuracy: 0.9664 - loss: 1.6698 - val\_accuracy: 0.9843 - val\_loss: 1.4725

Epoch 5/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 446ms/step - accuracy: 0.9687 - loss: 1.4608 - val\_accuracy: 0.9922 - val\_loss: 1.2783

Epoch 6/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **42s** 466ms/step - accuracy: 0.9827 - loss: 1.2621 - val\_accuracy: 0.9922 - val\_loss: 1.1178

Epoch 7/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 411ms/step - accuracy: 0.9809 - loss: 1.1129 - val\_accuracy: 0.9974 - val\_loss: 0.9822

Epoch 8/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 439ms/step - accuracy: 0.9796 - loss: 0.9758 - val\_accuracy: 0.9948 - val\_loss: 0.8672

Epoch 9/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 433ms/step - accuracy: 0.9823 - loss: 0.8753 - val\_accuracy: 0.9948 - val\_loss: 0.7683

Epoch 10/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **43s** 482ms/step - accuracy: 0.9860 - loss: 0.7763 - val\_accuracy: 0.9948 - val\_loss: 0.6844

Epoch 11/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 420ms/step - accuracy: 0.9836 - loss: 0.6794 - val\_accuracy: 0.9948 - val\_loss: 0.6105

Epoch 12/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 450ms/step - accuracy: 0.9888 - loss: 0.6080 - val\_accuracy: 0.9948 - val\_loss: 0.5465

Epoch 13/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 419ms/step - accuracy: 0.9909 - loss: 0.5457 - val\_accuracy: 0.9948 - val\_loss: 0.4919

Epoch 14/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 435ms/step - accuracy: 0.9879 - loss: 0.4929 - val\_accuracy: 0.9974 - val\_loss: 0.4441

Epoch 15/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 458ms/step - accuracy: 0.9957 - loss: 0.4414 - val\_accuracy: 0.9948 - val\_loss: 0.4030

Epoch 16/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 447ms/step - accuracy: 0.9919 - loss: 0.4090 - val\_accuracy: 0.9948 - val\_loss: 0.3663

Epoch 17/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 437ms/step - accuracy: 0.9858 - loss: 0.3738 - val\_accuracy: 0.9948 - val\_loss: 0.3335

Epoch 18/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 409ms/step - accuracy: 0.9860 - loss: 0.3420 - val\_accuracy: 0.9948 - val\_loss: 0.3059

Epoch 19/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 460ms/step - accuracy: 0.9878 - loss: 0.3111 - val\_accuracy: 0.9948 - val\_loss: 0.2822

Epoch 20/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 447ms/step - accuracy: 0.9859 - loss: 0.2893 - val\_accuracy: 0.9948 - val\_loss: 0.2580

Epoch 21/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **41s** 445ms/step - accuracy: 0.9915 - loss: 0.2604 - val\_accuracy: 0.9948 - val\_loss: 0.2392

Epoch 22/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 416ms/step - accuracy: 0.9927 - loss: 0.2427 - val\_accuracy: 0.9974 - val\_loss: 0.2204

Epoch 23/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 469ms/step - accuracy: 0.9858 - loss: 0.2348 - val\_accuracy: 0.9948 - val\_loss: 0.2056

Epoch 24/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 448ms/step - accuracy: 0.9926 - loss: 0.2083 - val\_accuracy: 0.9948 - val\_loss: 0.1912

Epoch 25/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **41s** 441ms/step - accuracy: 0.9908 - loss: 0.1969 - val\_accuracy: 0.9948 - val\_loss: 0.1785

Epoch 26/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **42s** 469ms/step - accuracy: 0.9917 - loss: 0.1835 - val\_accuracy: 0.9948 - val\_loss: 0.1657

Epoch 27/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 443ms/step - accuracy: 0.9901 - loss: 0.1721 - val\_accuracy: 0.9948 - val\_loss: 0.1541

Epoch 28/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **41s** 448ms/step - accuracy: 0.9897 - loss: 0.1607 - val\_accuracy: 0.9974 - val\_loss: 0.1439

Epoch 29/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 441ms/step - accuracy: 0.9923 - loss: 0.1532 - val\_accuracy: 0.9948 - val\_loss: 0.1368

Epoch 30/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 444ms/step - accuracy: 0.9951 - loss: 0.1478 - val\_accuracy: 0.9974 - val\_loss: 0.1276

Epoch 31/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 415ms/step - accuracy: 0.9893 - loss: 0.1404 - val\_accuracy: 0.9948 - val\_loss: 0.1228

Epoch 32/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 439ms/step - accuracy: 0.9916 - loss: 0.1358 - val\_accuracy: 0.9948 - val\_loss: 0.1171

Epoch 33/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 413ms/step - accuracy: 0.9917 - loss: 0.1223 - val\_accuracy: 0.9948 - val\_loss: 0.1096

Epoch 34/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 439ms/step - accuracy: 0.9955 - loss: 0.1144 - val\_accuracy: 0.9974 - val\_loss: 0.1037

Epoch 35/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 459ms/step - accuracy: 0.9965 - loss: 0.1085 - val\_accuracy: 0.9974 - val\_loss: 0.0983

Epoch 36/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 442ms/step - accuracy: 0.9878 - loss: 0.1126 - val\_accuracy: 0.9974 - val\_loss: 0.0945

Epoch 37/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 463ms/step - accuracy: 0.9967 - loss: 0.0982 - val\_accuracy: 0.9974 - val\_loss: 0.0903

Epoch 38/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 451ms/step - accuracy: 0.9915 - loss: 0.0995 - val\_accuracy: 0.9974 - val\_loss: 0.0858

Epoch 39/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 413ms/step - accuracy: 0.9952 - loss: 0.0887 - val\_accuracy: 0.9974 - val\_loss: 0.0808

Epoch 40/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 441ms/step - accuracy: 0.9943 - loss: 0.0855 - val\_accuracy: 0.9974 - val\_loss: 0.0784

Epoch 41/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 419ms/step - accuracy: 0.9870 - loss: 0.0942 - val\_accuracy: 0.9974 - val\_loss: 0.0766

Epoch 42/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 464ms/step - accuracy: 0.9942 - loss: 0.0810 - val\_accuracy: 0.9974 - val\_loss: 0.0724

Epoch 43/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 413ms/step - accuracy: 0.9954 - loss: 0.0812 - val\_accuracy: 0.9974 - val\_loss: 0.0717

Epoch 44/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 460ms/step - accuracy: 0.9945 - loss: 0.0779 - val\_accuracy: 0.9948 - val\_loss: 0.0706

Epoch 45/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 413ms/step - accuracy: 0.9888 - loss: 0.0847 - val\_accuracy: 0.9974 - val\_loss: 0.0670

Epoch 46/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 413ms/step - accuracy: 0.9929 - loss: 0.0723 - val\_accuracy: 0.9974 - val\_loss: 0.0652

Epoch 47/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 426ms/step - accuracy: 0.9963 - loss: 0.0703 - val\_accuracy: 0.9974 - val\_loss: 0.0618

Epoch 48/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 439ms/step - accuracy: 0.9844 - loss: 0.0834 - val\_accuracy: 0.9948 - val\_loss: 0.0640

Epoch 49/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 441ms/step - accuracy: 0.9957 - loss: 0.0704 - val\_accuracy: 0.9974 - val\_loss: 0.0607

Epoch 50/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 412ms/step - accuracy: 0.9947 - loss: 0.0680 - val\_accuracy: 0.9948 - val\_loss: 0.0584

Epoch 51/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 438ms/step - accuracy: 0.9911 - loss: 0.0732 - val\_accuracy: 0.9948 - val\_loss: 0.0587

Epoch 52/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 438ms/step - accuracy: 0.9949 - loss: 0.0641 - val\_accuracy: 0.9974 - val\_loss: 0.0558

Epoch 53/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 434ms/step - accuracy: 0.9933 - loss: 0.0637 - val\_accuracy: 0.9948 - val\_loss: 0.0550

Epoch 54/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 439ms/step - accuracy: 0.9893 - loss: 0.0701 - val\_accuracy: 0.9974 - val\_loss: 0.0526

Epoch 55/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 407ms/step - accuracy: 0.9938 - loss: 0.0623 - val\_accuracy: 0.9948 - val\_loss: 0.0543

Epoch 56/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 456ms/step - accuracy: 0.9961 - loss: 0.0649 - val\_accuracy: 0.9948 - val\_loss: 0.0541

Epoch 57/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 411ms/step - accuracy: 0.9906 - loss: 0.0648 - val\_accuracy: 0.9974 - val\_loss: 0.0521

Epoch 58/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 463ms/step - accuracy: 0.9885 - loss: 0.0610 - val\_accuracy: 0.9974 - val\_loss: 0.0497

Epoch 59/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **41s** 468ms/step - accuracy: 0.9968 - loss: 0.0485 - val\_accuracy: 0.9974 - val\_loss: 0.0483

Epoch 60/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 434ms/step - accuracy: 0.9930 - loss: 0.0572 - val\_accuracy: 0.9948 - val\_loss: 0.0521

Epoch 61/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **41s** 441ms/step - accuracy: 0.9900 - loss: 0.0692 - val\_accuracy: 0.9974 - val\_loss: 0.0479

Epoch 62/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 433ms/step - accuracy: 0.9978 - loss: 0.0467 - val\_accuracy: 0.9974 - val\_loss: 0.0476

Epoch 63/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 435ms/step - accuracy: 0.9965 - loss: 0.0489 - val\_accuracy: 0.9948 - val\_loss: 0.0487

Epoch 64/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 411ms/step - accuracy: 0.9955 - loss: 0.0487 - val\_accuracy: 0.9974 - val\_loss: 0.0456

Epoch 65/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 466ms/step - accuracy: 0.9933 - loss: 0.0565 - val\_accuracy: 0.9974 - val\_loss: 0.0443

Epoch 66/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 409ms/step - accuracy: 0.9968 - loss: 0.0502 - val\_accuracy: 0.9974 - val\_loss: 0.0451

Epoch 67/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 417ms/step - accuracy: 0.9961 - loss: 0.0471 - val\_accuracy: 0.9974 - val\_loss: 0.0430

Epoch 68/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 414ms/step - accuracy: 0.9924 - loss: 0.0600 - val\_accuracy: 0.9974 - val\_loss: 0.0439

Epoch 69/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 413ms/step - accuracy: 0.9937 - loss: 0.0477 - val\_accuracy: 0.9974 - val\_loss: 0.0427

Epoch 70/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 461ms/step - accuracy: 0.9964 - loss: 0.0449 - val\_accuracy: 0.9948 - val\_loss: 0.0437

Epoch 71/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 410ms/step - accuracy: 0.9928 - loss: 0.0492 - val\_accuracy: 0.9974 - val\_loss: 0.0423

Epoch 72/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 453ms/step - accuracy: 0.9960 - loss: 0.0423 - val\_accuracy: 0.9974 - val\_loss: 0.0416

Epoch 73/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 411ms/step - accuracy: 0.9972 - loss: 0.0442 - val\_accuracy: 0.9974 - val\_loss: 0.0415

Epoch 74/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 457ms/step - accuracy: 0.9905 - loss: 0.0513 - val\_accuracy: 0.9948 - val\_loss: 0.0431

Epoch 75/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **19s** 404ms/step - accuracy: 0.9923 - loss: 0.0515 - val\_accuracy: 0.9974 - val\_loss: 0.0436

Epoch 76/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 437ms/step - accuracy: 0.9971 - loss: 0.0421 - val\_accuracy: 0.9974 - val\_loss: 0.0400

Epoch 77/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 407ms/step - accuracy: 0.9958 - loss: 0.0426 - val\_accuracy: 0.9974 - val\_loss: 0.0405

Epoch 78/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 431ms/step - accuracy: 0.9939 - loss: 0.0443 - val\_accuracy: 0.9974 - val\_loss: 0.0405

Epoch 79/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 430ms/step - accuracy: 0.9942 - loss: 0.0463 - val\_accuracy: 0.9974 - val\_loss: 0.0407

Epoch 80/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 415ms/step - accuracy: 0.9963 - loss: 0.0434 - val\_accuracy: 0.9974 - val\_loss: 0.0394

Epoch 81/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 434ms/step - accuracy: 0.9969 - loss: 0.0380 - val\_accuracy: 0.9948 - val\_loss: 0.0401

Epoch 82/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **19s** 404ms/step - accuracy: 0.9952 - loss: 0.0458 - val\_accuracy: 0.9948 - val\_loss: 0.0405

Epoch 83/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 468ms/step - accuracy: 0.9938 - loss: 0.0462 - val\_accuracy: 0.9974 - val\_loss: 0.0400

Epoch 84/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 409ms/step - accuracy: 0.9923 - loss: 0.0513 - val\_accuracy: 0.9948 - val\_loss: 0.0424

Epoch 85/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 432ms/step - accuracy: 0.9913 - loss: 0.0511 - val\_accuracy: 0.9948 - val\_loss: 0.0406

Epoch 86/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 410ms/step - accuracy: 0.9913 - loss: 0.0499 - val\_accuracy: 0.9948 - val\_loss: 0.0398

Epoch 87/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 457ms/step - accuracy: 0.9915 - loss: 0.0512 - val\_accuracy: 0.9974 - val\_loss: 0.0397

Epoch 88/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 428ms/step - accuracy: 0.9931 - loss: 0.0441 - val\_accuracy: 0.9974 - val\_loss: 0.0388

Epoch 89/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 428ms/step - accuracy: 0.9937 - loss: 0.0447 - val\_accuracy: 0.9974 - val\_loss: 0.0383

Epoch 90/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 406ms/step - accuracy: 0.9973 - loss: 0.0420 - val\_accuracy: 0.9974 - val\_loss: 0.0370

Epoch 91/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 410ms/step - accuracy: 0.9927 - loss: 0.0464 - val\_accuracy: 0.9974 - val\_loss: 0.0355

Epoch 92/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 430ms/step - accuracy: 0.9951 - loss: 0.0427 - val\_accuracy: 0.9974 - val\_loss: 0.0386

Epoch 93/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 432ms/step - accuracy: 0.9936 - loss: 0.0466 - val\_accuracy: 0.9974 - val\_loss: 0.0372

Epoch 94/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **21s** 429ms/step - accuracy: 0.9970 - loss: 0.0399 - val\_accuracy: 0.9974 - val\_loss: 0.0376

Epoch 95/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **22s** 456ms/step - accuracy: 0.9890 - loss: 0.0442 - val\_accuracy: 0.9974 - val\_loss: 0.0375

Epoch 96/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **40s** 428ms/step - accuracy: 0.9933 - loss: 0.0428 - val\_accuracy: 0.9974 - val\_loss: 0.0372

Epoch 97/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 407ms/step - accuracy: 0.9935 - loss: 0.0436 - val\_accuracy: 0.9948 - val\_loss: 0.0383

Epoch 98/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 422ms/step - accuracy: 0.9970 - loss: 0.0399 - val\_accuracy: 0.9974 - val\_loss: 0.0362

Epoch 99/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 411ms/step - accuracy: 0.9942 - loss: 0.0390 - val\_accuracy: 0.9974 - val\_loss: 0.0372

Epoch 100/100

**48/48** ━━━━━━━━━━━━━━━━━━━━ **20s** 419ms/step - accuracy: 0.9953 - loss: 0.0396 - val\_accuracy: 0.9948 - val\_loss: 0.0372

Epoch 1/10

**48/48** ━━━━━━━━━━━━━━━━━━━━ **103s** 1s/step - accuracy: 0.9432 - loss: 0.1897 - val\_accuracy: 0.9922 - val\_loss: 0.0480

Epoch 2/10

**48/48** ━━━━━━━━━━━━━━━━━━━━ **32s** 566ms/step - accuracy: 0.9824 - loss: 0.0956 - val\_accuracy: 0.9948 - val\_loss: 0.0461

Epoch 3/10

**48/48** ━━━━━━━━━━━━━━━━━━━━ **27s** 554ms/step - accuracy: 0.9928 - loss: 0.0667 - val\_accuracy: 0.9948 - val\_loss: 0.0401

Epoch 4/10

**48/48** ━━━━━━━━━━━━━━━━━━━━ **27s** 557ms/step - accuracy: 0.9936 - loss: 0.0544 - val\_accuracy: 0.9974 - val\_loss: 0.0349

Epoch 5/10

**48/48** ━━━━━━━━━━━━━━━━━━━━ **28s** 576ms/step - accuracy: 0.9986 - loss: 0.0424 - val\_accuracy: 0.9974 - val\_loss: 0.0335

Epoch 6/10

**48/48** ━━━━━━━━━━━━━━━━━━━━ **27s** 554ms/step - accuracy: 0.9966 - loss: 0.0399 - val\_accuracy: 0.9974 - val\_loss: 0.0329

Epoch 7/10

**48/48** ━━━━━━━━━━━━━━━━━━━━ **28s** 587ms/step - accuracy: 0.9998 - loss: 0.0337 - val\_accuracy: 0.9974 - val\_loss: 0.0317

Epoch 8/10

**48/48** ━━━━━━━━━━━━━━━━━━━━ **27s** 568ms/step - accuracy: 0.9997 - loss: 0.0342 - val\_accuracy: 0.9974 - val\_loss: 0.0304

Epoch 9/10

**48/48** ━━━━━━━━━━━━━━━━━━━━ **28s** 584ms/step - accuracy: 0.9984 - loss: 0.0319 - val\_accuracy: 0.9974 - val\_loss: 0.0287

Epoch 10/10

**48/48** ━━━━━━━━━━━━━━━━━━━━ **27s** 563ms/step - accuracy: 1.0000 - loss: 0.0301 - val\_accuracy: 0.9974 - val\_loss: 0.0282

**12/12** ━━━━━━━━━━━━━━━━━━━━ **7s** 384ms/step

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", len(result))

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/\_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", len(result))



