% segmentation and spectrogram

data = load('your data here');

signal = data.your data name;

file\_name='your data';

Fs = 128;

segment\_length = 30;

num\_segments = floor(length(signal) / (segment\_length \* Fs));

window = hamming(segment\_length \* Fs);

noverlap = floor(0.5 \* length(window));

nfft = 2^nextpow2(length(window));

save\_directory = 'your directory here';

for i = 1:num\_segments

start\_index = (i - 1) \* segment\_length \* Fs + 1;

end\_index = min(i \* segment\_length \* Fs, length(signal));

segment = signal(start\_index:end\_index);

[S, F, T] = spectrogram(segment, window, noverlap, nfft, Fs);

h = figure('Visible', 'off');

imagesc(T, F, 10\*log10(abs(S)));

axis xy;

xlabel('Time (s)');

ylabel('Frequency (Hz)');

title(sprintf('Segment %d Spectrogram', i));

colorbar;

file\_name = sprintf('%s\_spectrogram\_%d.png',file\_name,i);

full\_file\_dir = fullfile(save\_directory, file\_name);

saveas(h, full\_file\_dir);

end

close(h);

% train/val/test separation

import os

import random

import shutil

dataset\_dir = "/your directory here"

folders= ["your spectrogram folders here"]

train\_ratio = 0.7

val\_ratio = 0.15

test\_ratio = 0.15

for folder in folders:

    folder\_dir = os.path.join(dataset\_dir, folder)

    for k in ["train", "val", "test"]:

        os.makedirs(os.path.join(folder\_dir, k), exist\_ok=True)

   features = os.listdir(folder\_dir)

    for feature in features:

        feature\_dir = os.path.join(folder\_dir, feature)

        pictures = os.listdir(feature\_dir)

        random.shuffle(pictures)

        train\_number = int(len(pictures) \* train\_ratio)

        val\_number = int(len(pictures) \* val\_ratio)

% ResNet64 Classifier

!pip list | grep -e fastai -e torch -e torchvision

!pip install roboflow

from roboflow import Roboflow

rf = Roboflow(api\_key="YOUR API KEY HERE")

project = rf.workspace("your workspace name here").project("your Project name here")

dataset = project.version(2).download("folder")

import os

labels = os.listdir("Your directory here")

print("No. of labels: {}".format(len(labels)))

print("-----------------")

for label in labels:

    print("{}, {} files".format(label, len(os.listdir("your directory here"+label))))

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

fig, ax = plt.subplots(nrows=2, ncols=2)

fig.tight\_layout()

cnt = 0

for row in ax:

    for col in row:

        image\_name = np.random.choice(os.listdir("your directory here"+ labels[cnt]))

        im = Image.open("your directory here{}/{}".format(labels[cnt],image\_name))

        col.imshow(im)

        col.set\_title(labels[cnt])

        col.axis('off')

        cnt += 1

plt.show()

from fastai.vision import \*

from fastai.metrics import accuracy

import torchvision.transforms as transforms

from torchvision import datasets

from torch.utils.data import DataLoader

path = "your data directory here"

mean = [0.485, 0.456, 0.406]

std = [0.229, 0.224, 0.225]

normalize = transforms.Normalize(mean=mean, std=std)

transform = transforms.Compose([

    transforms.Resize(size),

    transforms.ToTensor(),

    normalize,

])

dataset = datasets.ImageFolder(root=path, transform=transform)

batch\_size = bs

data\_loader = DataLoader(dataset, batch\_size=batch\_size, shuffle=True, num\_workers=4)

for batch in data\_loader:

    images, labels = batch

from fastai.vision.all import \*

data = DataBlock(blocks=(ImageBlock, CategoryBlock),

                 get\_items=get\_image\_files,

                 splitter=RandomSplitter(valid\_pct=0.2, seed=42),

                 get\_y=parent\_label,

                 item\_tfms=Resize(460),

                 batch\_tfms=[\*aug\_transforms(size=224), Normalize.from\_stats(\*imagenet\_stats)])

dls = data.dataloaders(path)

learner = cnn\_learner(dls, models.resnet34, metrics=[accuracy, Precision(average='macro'), Recall(average='macro'), F1Score(average='macro')])

learner.fine\_tune(20)

interp = ClassificationInterpretation.from\_learner(learner)

interp.plot\_confusion\_matrix()

import torch

import torch.nn as nn

from torchsummary import summary

from torchvision.models import resnet50  # ya da diğer ResNet modellerini seçebilirsiniz

class ResNet64(nn.Module):

    def \_\_init\_\_(self):

        super(ResNet64, self).\_\_init\_\_()

        self.resnet = resnet50()

    def forward(self, x):

        return self.resnet(x)

model = ResNet64()

print(model)

input\_channels, height, width = 3, 224, 224

summary(model, input\_size=(input\_channels, height, width))

% YOLOv5 Classifier

!git clone https://github.com/ultralytics/yolov5  # clone

%cd yolov5

%pip install -qr requirements.txt  # install

import torch

import utils

display = utils.notebook\_init()  # checks

from utils.downloads import attempt\_download

p5 = ['n', 's', 'm', 'l', 'x']  # P5 models

cls = [f'{x}-cls' for x in p5]  # classification models

for x in cls:

    attempt\_download(f'weights/yolov5{x}.pt')

#Infer using classify/predict.py

!python classify/predict.py --weights ./weigths/yolov5s-cls.pt --source bananas.jpg

# Ensure we're in the right directory to download our custom dataset

import os

os.makedirs("../datasets/", exist\_ok=True)

%cd ../datasets/

# REPLACE the below with your exported code snippet from above

!pip install roboflow

from roboflow import Roboflow

rf = Roboflow(api\_key="YOUR API KEY")

project = rf.workspace("yolov5-classification").project("banana-ripeness-classification")

dataset = project.version(1).download("folder")

#Save the dataset name to the environment so we can use it in a system call later

dataset\_name = dataset.location.split(os.sep)[-1]

os.environ["DATASET\_NAME"] = dataset\_name

%cd ../yolov5

!python classify/train.py --model yolov5s-cls.pt --data $DATASET\_NAME --epochs 20 --img 128 --pretrained weights/yolov5s-cls.pt

!python classify/val.py --weights runs/train-cls/exp/weights/best.pt --data ../datasets/$DATASET\_NAME

#Get the path of an image from the test or validation set

if os.path.exists(os.path.join(dataset.location, "test")):

  split\_path = os.path.join(dataset.location, "test")

else:

  os.path.join(dataset.location, "valid")

example\_class = os.listdir(split\_path)[0]

example\_image\_name = os.listdir(os.path.join(split\_path, example\_class))[0]

example\_image\_path = os.path.join(split\_path, example\_class, example\_image\_name)

os.environ["TEST\_IMAGE\_PATH"] = example\_image\_path

print(f"Inferring on an example of the class '{example\_class}'")

#Infer

!python classify/predict.py --weights runs/train-cls/exp/weights/best.pt --source $TEST\_IMAGE\_PATH

#Directory infer

os.environ["TEST\_CLASS\_PATH"] = test\_class\_path = os.path.join(\*os.environ["TEST\_IMAGE\_PATH"].split(os.sep)[:-1])

print(f"Infering on all images from the directory {os.environ['TEST\_CLASS\_PATH']}")

!python classify/predict.py --weights runs/train-cls/exp/weights/best.pt --source /$TEST\_CLASS\_PATH/

% YOLOv8 Classifier

import os

HOME = os.getcwd()

print(HOME)

# Pip install method (recommended)

!pip install ultralytics

from IPython import display

display.clear\_output()

import ultralytics

ultralytics.checks()

from ultralytics import YOLO

%cd {HOME}

%cd {HOME}

Image(filename='runs/classify/predict/dog.jpeg', height=600)

!yolo task=classify mode=predict model=yolov8n-cls.pt conf=0.25 source='https://media.roboflow.com/notebooks/examples/dog.jpeg'

from IPython.display import display, Image

model = YOLO(f'{HOME}/yolov8n-cls.pt')

results = model.predict(source='https://media.roboflow.com/notebooks/examples/dog.jpeg', conf=0.25)

!mkdir {HOME}/datasets

%cd {HOME}/datasets

!pip install roboflow --quiet

from roboflow import Roboflow

rf = Roboflow(api\_key="YOUR\_API\_KEY")

project = rf.workspace("roboflow-universe-projects").project("banana-ripeness-classification")

dataset = project.version(4).download("folder")

%cd {HOME}

!yolo task=classify mode=train model=yolov8n-cls.pt data={dataset.location} epochs=20 imgsz=128

!ls -la {HOME}/runs/classify/train/

!cat {HOME}/runs/classify/train/results.csv | head -10

%cd {HOME}

!yolo task=classify mode=val model={HOME}/runs/classify/train/weights/best.pt data={dataset.location}

%cd {HOME}

!yolo task=classify mode=predict model={HOME}/runs/classify/train/weights/best.pt conf=0.25 source={dataset.location}/test/overripe

import glob

from IPython.display import Image, display

for image\_path in glob.glob(f'{HOME}/runs/classify/predict/\*.jpg')[:3]:

      display(Image(filename=image\_path, width=600))

      print("\n")

project.version(dataset.version).deploy(model\_type="yolov8-cls", model\_path=f"{HOME}/runs/classify/train/")