Logo

Description automatically generated

CSE485 Deep Learning

**Project Documentation**

**Submitted to:**

Prof. Dr. Mahmoud Ibrahim Khalil

Eng. Mahmoud Soheil

**Submitted by:**

Youssef George Fouad 19P9824

Kerollos Wageeh Youssef 19P3468

Mahmoud Mohamed Omar 19P5803

CESS Senior-2

Table of Contents

[1. Problem Description 3](#_Toc154279582)

[2. GETTING STARTED 3](#_Toc154279583)

[2.1. Importing Needed Libraries 3](#_Toc154279584)

[2.2. Importing Leaf Classification Dataset 4](#_Toc154279585)

[3. DATA EXPLORATION 5](#_Toc154279586)

[4. DATA PREPROCESSING 6](#_Toc154279587)

[4.1. Standardization 6](#_Toc154279588)

[4.2. Correlation Analysis 7](#_Toc154279589)

[4.2.1. Margin features analysis 7](#_Toc154279590)

[4.2.2. Shape features analysis 9](#_Toc154279591)

[4.2.3. Texture features analysis 10](#_Toc154279592)

[4.3. Label Encoding 11](#_Toc154279593)

[4.4. MLP Neural Network 12](#_Toc154279594)

[4.4.1. Train-Test split 12](#_Toc154279595)

[4.4.2. Neural network architecture 13](#_Toc154279596)

[4.4.3. Training 14](#_Toc154279597)

[4.4.4. Results 15](#_Toc154279598)

[5. DEEP CONVOLUTIONAL NEURAL NETWORK 16](#_Toc154279599)

[5.1. Import Images and Preprocessing 16](#_Toc154279600)

[5.2. First Architecture and Trial 17](#_Toc154279601)

[5.3. Second Architecture and Trial 18](#_Toc154279602)

[5.4. Transfer Learning with a Pre-Trained ResNet-18 19](#_Toc154279603)

[5.5. Comparing CNN Trials 20](#_Toc154279604)

[6. A HYBRID APPROACH 21](#_Toc154279605)

[7. CONCLUSION 22](#_Toc154279606)

[8. SOURCE CODE and NOTEBOOK 23](#_Toc154279607)

# Problem Description

There are estimated to be nearly half a million species of plant in the world. Classification of species has been historically problematic and often results in duplicate identifications.

The objective of this project is to use binary leaf images, to classify 99 species of plants.

# GETTING STARTED

## Importing Needed Libraries

Before building our deep learning model, we start by importing needed python libraries that help us process the images, build, train, test our model accuracy, and visualize the results.

# deep learning libraries (pytorch)

import torch

import torch.nn as nn

import torch.optim as optim

import torch.nn.functional as F

import albumentations as A

from albumentations.pytorch import ToTensorV2

from torchvision import datasets, transforms, models

# importing dataset

import os

from PIL import Image

import cv2

import pandas as pd

# data processing

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

import numpy as np

# visualization

import matplotlib.pyplot as plt

from tqdm import tqdm

from torch.utils.tensorboard import SummaryWriter

import time

## Importing Leaf Classification Dataset

The dataset consists of 990 **binary** images, labels. In addition, pre-extracted features provide a numerical representation of leaves different properties, which are margins, shapes, texture.

* There are 64 marginal features, 64 shape features, 64 texture features.

The **binary** images are in **JPG** format which are easy to read in shape.

* View 100 sample images of the dataset.

image\_filenames = sorted(os.listdir("Dataset/images/"))

fig = plt.figure(figsize=(10, 10))

gs = gridspec.GridSpec(10, 10, wspace=0.05, hspace=0.05)

for i, image\_filename in enumerate(image\_filenames[:100]):

image\_path = os.path.join(image\_folder, image\_filename)

img = Image.open(image\_path).convert('1') # '1' for black/white

ax = plt.subplot(gs[i])

plt.imshow(img)

A black and white image of different leaves

Description automatically generated

# DATA EXPLORATION

In this section we will explore the given numerical features of “train.csv” file.

* Import data in pandas DataFrame and check its shape and features type.

train\_csv = pd.read\_csv('Dataset/train.csv')

print(f"There are {len(train\_csv.species.unique())} species in the train set.")

print(f"There are {len(train\_csv.id.unique())} images in the train set.")

print(f"There are {train\_csv.shape[1]} columns in the train set.")

print(f"There are {train\_csv.isnull().sum()} missing values in the train set.")

* There are 990 images in the train dataset, which we will work on, divided among 99 classes/ labels/ species.

A black background with white text

Description automatically generated

* Each image is given an ID column, a label (species), 64 margin features, 64 shape features, and 64 texture features. **No missing values at all!**

print(train\_csv.dtypes.value\_counts())

print(f"{train\_csv.select\_dtypes(include=['int64']).columns[0]} is the only int column in the train set.")

print(f"{train\_csv.select\_dtypes(include=['object']).columns[0]} is the only object column in the train set.")

print(f"There are {train\_csv.filter(regex='margin').shape[1]} MARGIN features.")

print(f"There are {train\_csv.filter(regex='shape').shape[1]} SHAPE features.")

print(f"There are {train\_csv.filter(regex='texture').shape[1]} TEXTURE features.")

A screen shot of a computer

Description automatically generated

# DATA PREPROCESSING

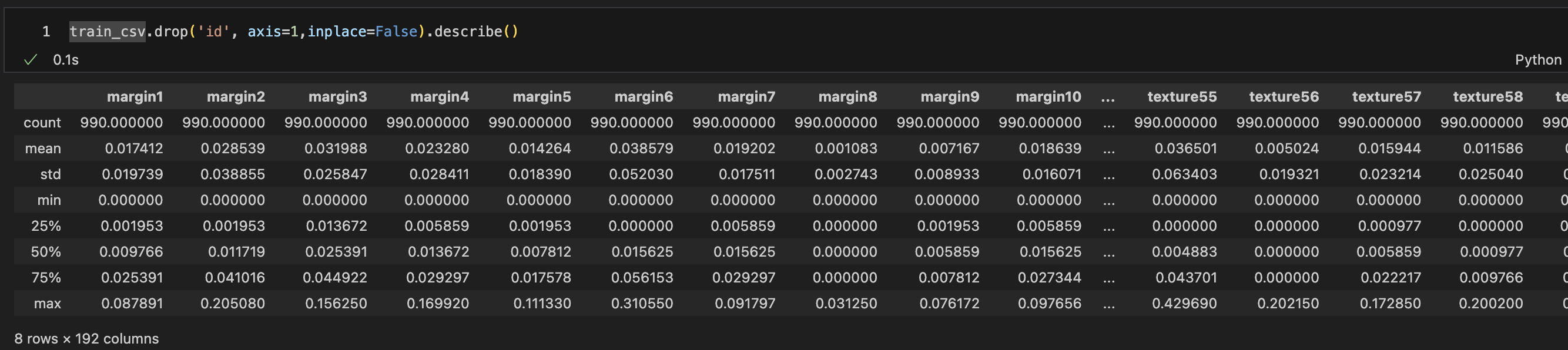
## Standardization

Standardizing numerical features ensures all of them are on a certain scale having a zero mean and standard deviation of one, which helps in better understanding of the relationships between these features in addition to helping us to reach better model training and performance results.

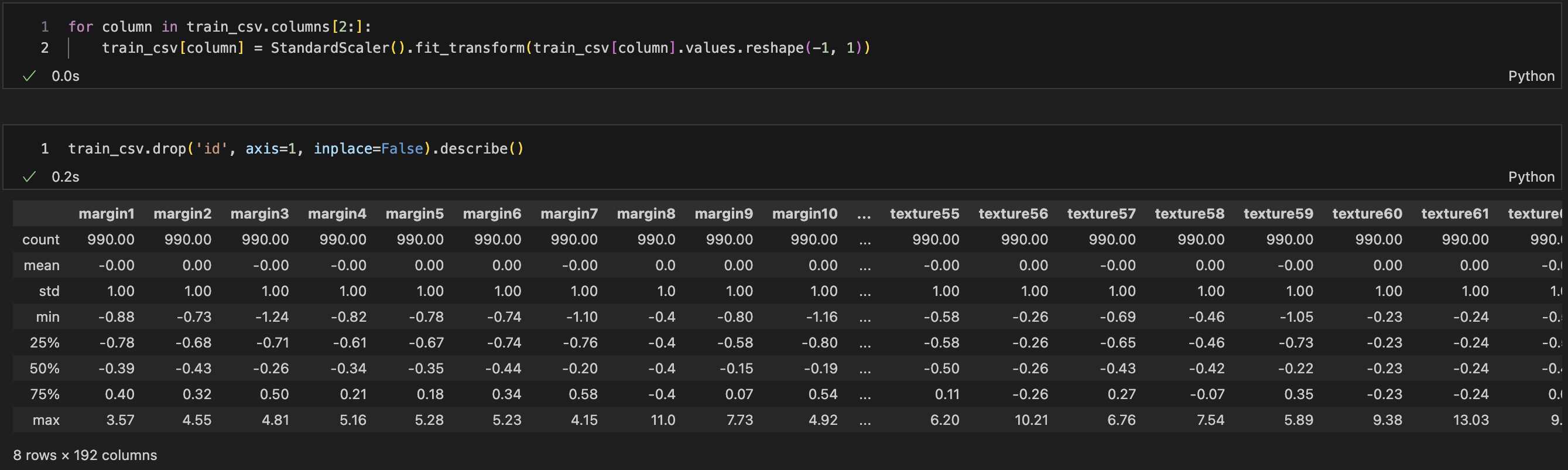
where:

As we can see in the table below, the mean and standard deviation for each feature varies; therefore, standardization helps us flatten the mean and standard deviation to 0 and 1.

**Before standardization:**



**After standardization:**



## Correlation Analysis

Since we have 192 numerical features, we would undergo Pearson correlation analysis between each category of features alone (margin, shape, texture).

We undergo correlation analysis between features of data to quantify relationships between these features and try to find patterns, identify trends, or exclude redundant features.

After we calculate correlation between each feature and the other features, we end up with a (n x n) matrix of numbers ranging from -1 to 1 that resembles how strong each feature correlates with another one.

We will consider only those who are > |0.8|, which are >0.8 or <-0.8, and we will filter them out in a separate chart as shown below in each of the 3 features categories.

### Margin features analysis

fig, ax = plt.subplots(1,2,figsize=(15,5))

margin\_corr = train\_csv.filter(regex='margin').corr()

sns.heatmap(margin\_corr,vmin=-1, vmax=1, center=0, cmap='viridis', mask=np.triu(np.ones\_like(margin\_corr,dtype=bool)), linewidths=0.5, ax=ax[0])

sns.heatmap(margin\_corr,vmin=-1, vmax=1, center=0, cmap='viridis', mask= np.triu(np.ones\_like(margin\_corr, dtype=bool))|(np.abs(margin\_corr) <= 0.8) , linewidths=0.5, ax=ax[1])

plt.title('Correlation between margin features')

plt.show()

print(f"There are {train\_csv.filter(regex='margin').shape[1]} MARGIN features in the train set.")

for i in range(len(margin\_corr)):

for j in range(i+1,len(margin\_corr)):

if i != j and abs(margin\_corr.iloc[i,j]) >= 0.8:

print(f"{margin\_corr.columns[i]} and {margin\_corr.columns[j]} have high correlation = {round(margin\_corr.iloc[i,j],2)}")

The results show how only few of the 64 marginal features are highly correlated, i.e. pass our threshold > |0.8|

A graph of a number of colored dots

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated

### Shape features analysis

fig, ax = plt.subplots(1,2,figsize=(15,5))

shape\_corr = train\_csv.filter(regex='shape').corr()

sns.heatmap(shape\_corr,vmin=-1, vmax=1, center=0, cmap='viridis', mask=np.triu(np.ones\_like(shape\_corr,dtype=bool)), linewidths=0.5, ax=ax[0])

sns.heatmap(shape\_corr,vmin=-1, vmax=1, center=0, cmap='viridis', mask= np.triu(np.ones\_like(shape\_corr, dtype=bool))|(np.abs(shape\_corr) <= 0.8) , linewidths=0.5, ax=ax[1])

plt.title('Correlation between shape features')

plt.show()

print(f"There are {train\_csv.filter(regex='shape').shape[1]} SHAPE features in the train set.")

for i in range(len(shape\_corr)):

for j in range(i+1,len(shape\_corr)):

if i != j and abs(shape\_corr.iloc[i,j]) >= 0.8:

print(f"{shape\_corr.columns[i]} and {shape\_corr.columns[j]} have high correlation = {round(shape\_corr.iloc[i,j],2)}")

The results show how a lot of the 64 shape features are highly correlated with each other, i.e. pass our threshold > |0.8|

A screenshot of a graph

Description automatically generated

### Texture features analysis

fig, ax = plt.subplots(1,2,figsize=(15,5))

shape\_corr = train\_csv.filter(regex='texture').corr()

sns.heatmap(shape\_corr,vmin=-1, vmax=1, center=0, cmap='viridis', mask=np.triu(np.ones\_like(shape\_corr,dtype=bool)), linewidths=0.5, ax=ax[0])

sns.heatmap(shape\_corr,vmin=-1, vmax=1, center=0, cmap='viridis', mask= np.triu(np.ones\_like(shape\_corr, dtype=bool))|(np.abs(shape\_corr) <= 0.8) , linewidths=0.5, ax=ax[1])

plt.title('Correlation between texture features')

plt.show()

print(f"There are {train\_csv.filter(regex='texture').shape[1]} TEXTURE features in the train set.")

for i in range(len(shape\_corr)):

for j in range(i+1,len(shape\_corr)):

if i != j and abs(shape\_corr.iloc[i,j]) >= 0.8:

print(f"{shape\_corr.columns[i]} and {shape\_corr.columns[j]} have high correlation = {round(shape\_corr.iloc[i,j],2)}")

A comparison of a graph

Description automatically generated with medium confidence

A black background with white text

Description automatically generated

The results show how only 1 pair of the 64 texture features are highly correlated, i.e. pass our threshold > |0.8|

## Label Encoding

As we can see in the screenshot below, the label is not encoded but rather a string of the leaf class, which cannot be used in neural networks training process.

A screenshot of a computer

Description automatically generated

Therefore, encoding the species column using a Label Encoder will replace the text values with integer values between 1-99 replacing the 99 different classes of the leaves’ species.

train\_data = pd.DataFrame()

train\_data['label'] = LabelEncoder().fit\_transform(train\_csv['species'])

train\_data = pd.concat([train\_data,train\_csv.drop(['id','species'],axis=1)],axis=1)

print(f"There are {train\_csv.label.nunique()} unique labels in the train set.")

print(f"There are {train\_csv.species.nunique()} unique species in the train set.")

train\_data.head()

A black and white screen with numbers

Description automatically generated

## MLP Neural Network

With the 100 standardized extracted features we reached this stage with, we will try to build a classification model that classifies an input image between the 99 classes of leaves.

### Train-Test split

### Neural network architecture

### Training

### Results

# DEEP CONVOLUTIONAL NEURAL NETWORK

## Import Images and Preprocessing

## First Architecture and Trial

## Second Architecture and Trial

## Transfer Learning with a Pre-Trained ResNet-18

## Comparing CNN Trials

# A HYBRID APPROACH

# CONCLUSION

# SOURCE CODE and NOTEBOOK

You can access the project source code on GitHub. Press [Here.](https://github.com/youssefg7/CNNLeafClassification)