# Artificial Neural Networks and Deep Learning
# Homework #1

(Fall 2023)

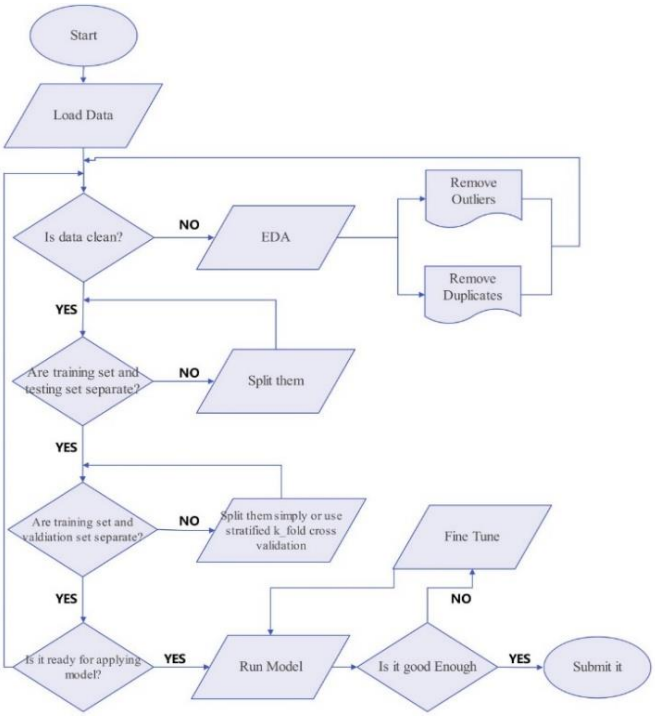

Team: **Perceptron Playoff**

Members: **Mohammad Amiri (10887256)**
**Sara Limooee (100886949)**
**Dorsa Moadeli (10926114)**
**Mohamed Shala (10871548)**

**Politecnico Di Milano**


---

## Introduction




***Project Process***

## EDA on dataset

We initiated the project by uploading a comprehensive umpire array dataset that included both healthy and unhealthy images. Upon analyzing the data distribution, we noted that around 60% of the data pertained to healthy majors, while the remaining 40% represented unhealthy images. This imbalance in our dataset prompted further investigation. In the subsequent phase, we closely examined images to classify them into two major categories: unhealthy, featuring images of leaves, and healthy leaves. We meticulously analyzed image details during this process, identifying potential outliers that could adversely affect our model's performance. We took proactive measures to eliminate these outliers. Leveraging the OpenCV-2 library, we employed a methodology measuring the distance between unwanted images and a reference image. This approach identified approximately 196 images, portraying Sherek and Trollo, as unwanted. Additionally, we implemented a technique to identify and remove 154 duplicate applications by comparing pixel-wise similarities during the Exploratory Data Analysis (EDA) process. After completing these data preprocessing steps, we successfully retained 4850 clean images devoid of outliers and duplicate applications. Moving on

to the EDA section, we partitioned the data into training and test sets, with the test set comprising 100 images and maintaining a 60-40 distribution between unhealthy and healthy majors. Following the data preparation steps, we executed a simple split for the training and validation sections. To further enhance our work, we have plans to implement stratified k-fold cross-validation in the future. This effort resulted in a performance improvement ranging from 5 - 10% compared to our previous performance.

## Data augmentation

In order to prevent overfitting and augment the dataset size, we implemented data augmentation on our dataset before feeding it into the model.fit() function. It's important to note that we applied data augmentation only to the training data after splitting it from the test data, as the intention was to enhance the training phase exclusively. Initially, we introduced a new layer from the Keras library to our model before the actual training layers. This new layer included two methods: "Random Brightness" and "Random Translation" for data augmentation. Subsequently, we incorporated additional data augmentation methods into our models.

## Models

We began the project by setting up a basic neural network, the Simple Convolutional Neural Network (Simple CNN) which achieved an initial accuracy of 65%. Later we added some more layers such as MaxPooling and Dropout and implementing the early stopping technique to avoid overfitting, we were able to increase the accuracy up to 75% locally. As we progressed, we dug deeper into transfer learning, exploring the use of models pre-trained on the ImageNet dataset. The table below illustrates the validation accuracy for each model implemented through these advanced techniques.

| Model | Accuracy |
|---|---|
| Simple CNN | Around 70 |
| Simple CNN + maxpooling | Around 75 |
| Simple CNN + maxpooling + early stopping | Around 76 |
| Simple CNN + maxpooling + early stopping + augmentation | Around 83 |
| Simple CNN + maxpooling + early stopping + augmentation + dropout | Around 84 |
| Transfer Learning (VGG16) | Around 62 |
| Transfer Learning (VGG16) + Fine Tuning | Around 64 |
| Transfer Learning (ResNet50) | Around 62 |
| Transfer Learning (ResNet50) +Fine Tuning | Around 63 |
| Transfer Learning (EfficientB2) | Around 46 |
| Transfer Learning (EfficientB2) + Fine Tuning | Around 46 |

Following the implementation of these techniques, we identified signs of overfitting in our data. In response, we adjusted our data augmentation method. Moreover, we made the strategic decision to persist with three pre-trained models—MobileNetV2, EfficientNetB0, and Xception. The ensuing accuracies achieved by these models are detailed below.

| Model | Accuracy |
|---|---|
| Transfer Learning (MobileNet) | Around 72 |
| Transfer Learning (MobileNet) + Fine Tuning | Around 82 |
| Transfer Learning (EfficientB0) | 71 |
| Transfer Learning (EfficientB0) + Fine Tuning | 81 |
| Transfer Learning (MobileNet) + Fine Tuning + K-fold | 84 |
| Transfer Learning (EfficientB0) + Fine Tuning + K-fold | 85 |

In the initial phase, we achieved an accuracy of 82% on the server. We then switched to ensemble models in the second phase, which resulted in a 75% accuracy on the server with different test data from the first phase.

## Self-Supervised Learning

In our pursuit of enhancing model performance, we incorporated self-supervised learning using SIMCLR into our methodology. By leveraging contrastive learning, SIMCLR enables the model to learn intricate patterns and features inherent in the data, ultimately leading to improved generalization and robustness. Since we had overfitting models. The decision to integrate SIMCLR into our model training pipeline stemmed from its ability to extract meaningful representations without the need for explicit labeling in the training data. This is particularly advantageous in scenarios where labeled data is scarce or expensive to obtain. Through the application of self-supervised learning with SIMCLR, we aimed to empower our model with a more comprehensive understanding of the data, ultimately contributing to enhanced performance and adaptability across various tasks. But, in the end, we were not so lucky to run SIMCLR on our models. We think that we made a kind of mistake in feeding data to the structure.

## Ensembled model

In this part, we tried to train the models we implemented in previous steps with 3 approaches. First, we give the whole train set to the models and evaluate its accuracy using test data. In the second approach, we do KFold or Stratified KFold cross-validation and store each model of each fold with its accuracy score. For the third way, we do a resampling of 2800 data (around 60% of the data train) and train each model with those selected data. As a result, we had 20 models trained in which we would like to select those with superior accuracy from the implementations and combine them through an ensemble approach. Referencing the table below, you will find a comprehensive overview of the tested models, including the accuracy of each model on the complete test data and within class-0 and class-1 categories.

| Model No | Model2 | Model5 | Model8 | Model13 | Model15 | Model17 |
|---|---|---|---|---|---|---|
| Accuracy total | 80% | 81% | 80% | 76% | 81% | 80% |
| Accuracy class 0 | 85% | 95% | 87.50% | 75% | 95% | 95% |
| Accuracy class 1 | 78.33% | 71.67% | 75% | 76.67% | 71.67% | 70% |

Specifically, we applied various models to each test dataset, individually predicting the output of each model. Notice that for performing the prediction of each model, we must first perform the required preprocessing on test data according to each model.

To obtain the final prediction label through the aggregation of all individual model outputs, we performed an argmax operation on the sum of predictions generated by each model. It is worth noting that the return value of the prediction function is a float probability; hence, we applied the argmax operation post the summation process.

In our experimentation, we successfully employed an ensemble of four distinct models, resulting in an accuracy of 79% in local testing and 75% on the server. The ensemble comprised the following models:

- Model1: MobileNetV2
- Model2: EfficientNetB0
- Model3: EfficientNetB0 with KFold Cross Validation-Fold 3
- Model4: MobileNetV2 with KFold Cross Validation-Fold 5

For easy execution, access the clean dataset, derived from the provided original dataset via this link.

## Contributions

In the initial exploratory data analysis (EDA) phase, Dorsa and Mohamed spent three days refining the dataset. They focused on extracting valuable insights, with Mohammad implementing functions, including outlier detection, to enhance data quality. Moving on to the model definition stage, Mohammad and Sara collaborated closely. They experimented with various models, incorporating data augmentation into the structures. Meanwhile, Dorsa and Mohamed worked on enhancing the initial 65% result obtained by simple CNN by adding layers for improved performance. Concurrently, Mohammad and Sara explored diverse models such as ResNet50 and MobileNet. Following the model definition, the team collectively fine-tuned the models, contributing different ideas that significantly improved performance. Mohamed conducted stratified k-fold cross-validation, and Mohammad implemented a self-supervised learning method to further enhance results. In parallel, Dorsa and Sara aimed to exceed expectations. They employed ensemble methods, such as max voting on different model runs, to improve results. In the final stage, each participant contributed to specific sections of the report based on their tasks. Simultaneously, Mohamed took on the task of cleaning and logically sorting the code for clarity.