Neural Networks Project

NAME	SEC	BN	CODE
Mustafa Tarek Salah	2	21	9211178
Mohammed Yasser	2	13	9211066
Ahmed Bassem Elkady	1	7	9210048
Daniel Nabil Atallah	1	16	9210386

Table of Contents

Project Pipeline

The project consists of two phases which are:

- **The training phase** is done in a few steps:
  - Preprocessing the input data which is the fonts-dataset and saving the outputs in a new folder.
  - Read the folder of the preprocessed data.
  - Train the preprocessed data through some classifiers.
  - Print the accuracy of each model.

- **The testing phase** is done in a few steps:
  - Preprocessing the data to be tested and saving the outputs in a new folder.
  - Read the folder of the preprocessed test data.
  - Predict the preprocessed data through the best classifier.
  - Save the outputs of prediction for each image and its time to be predicted.

Preprocessing Module

This module contains functions for preprocessing images, particularly for tasks like text detection and extraction which are:

1. Binarize Image (binarizeImage): This function converts the input image to grayscale, calculates the mean pixel value, determines if it's a light or dark image, then applies thresholding to create a binary image.
2. Rotate Image (rotateImage): This function rotates the input image by a specified angle. It's primarily used to correct the orientation of images.
3. Hough Transform (houghTransform): This function applies the Hough transform to detect lines in the input image. It's used to correct skewness caused by the rotation of the text.
4. Correct Text Orientation (correctTextOrientation): This function detects and corrects the orientation of text in the image. It uses Tesseract OCR (pytesseract) to analyze the orientation of text and rotates the image accordingly.
5. Filter Image (filterImage): This function applies a median filter to the image to reduce noise.
6. Extract Lines (extractLines): This function extracts lines of text from the preprocessed image. It uses contour detection to identify text regions and then extracts those regions.

Preprocessing Module

We have two main functions that make the pipeline of preprocessing which are:

- Dataset Preprocessing (datasetPreprocess): This function preprocesses a dataset of images by iterating through each image, applying the preprocessing steps mentioned above in the same order, and saving the preprocessed images into corresponding folders in an output directory.
- Image Preprocess (imagePreprocess): This function preprocesses a single image by applying the same preprocessing steps as in the dataset preprocessing.

Feature Extraction Module

This module provides functions for computing Local Phase Quantization (LPQ) features from input images which are:

1. LPQ Feature Extraction (lpq): This function calculates the Local Phase Quantization (LPQ) features from an input image. LPQ is a method used in image processing for feature extraction. It quantifies the local phase information of an image. It returns a vector of size 255 containing LPQ descriptors for the input image.
2. getFeaturesList: This function computes LPQ features for an array of images. It iterates over each image in the input array, computes LPQ features using the `lpq` function, and appends the result to a list.

We also tried to get SIFT features and make PCA to extract features but this attempt gives us lower accuracy so we remove it.

Training Module

This module contains two functions to apply SVM (Support Vector Machine) training and prediction which are:

1. Train Function (svmTrain(data_set, Y)):
    a. - This function is for training an SVM model.
    b. - It splits the dataset into training and validation sets using `train_test_split`.
    c. - It extracts features from the training set using the `getFeaturesList` function.
    d. - It initializes an SVM classifier pipeline with scaling and fitting it on the training data.
    e. - It saves the trained model using `pickle`.
    f. - It evaluates the model's accuracy on both the training and validation sets and prints the results.
2. Predict Function (svmPredict(data_set)):
    a. This function is for making predictions using the trained SVM model.
    b. It extracts features from the input dataset using getFeaturesList.
    c. It loads the trained SVM model from the saved file using `pickle`.
    d. It predicts the labels of the input dataset using the loaded model and returns it.

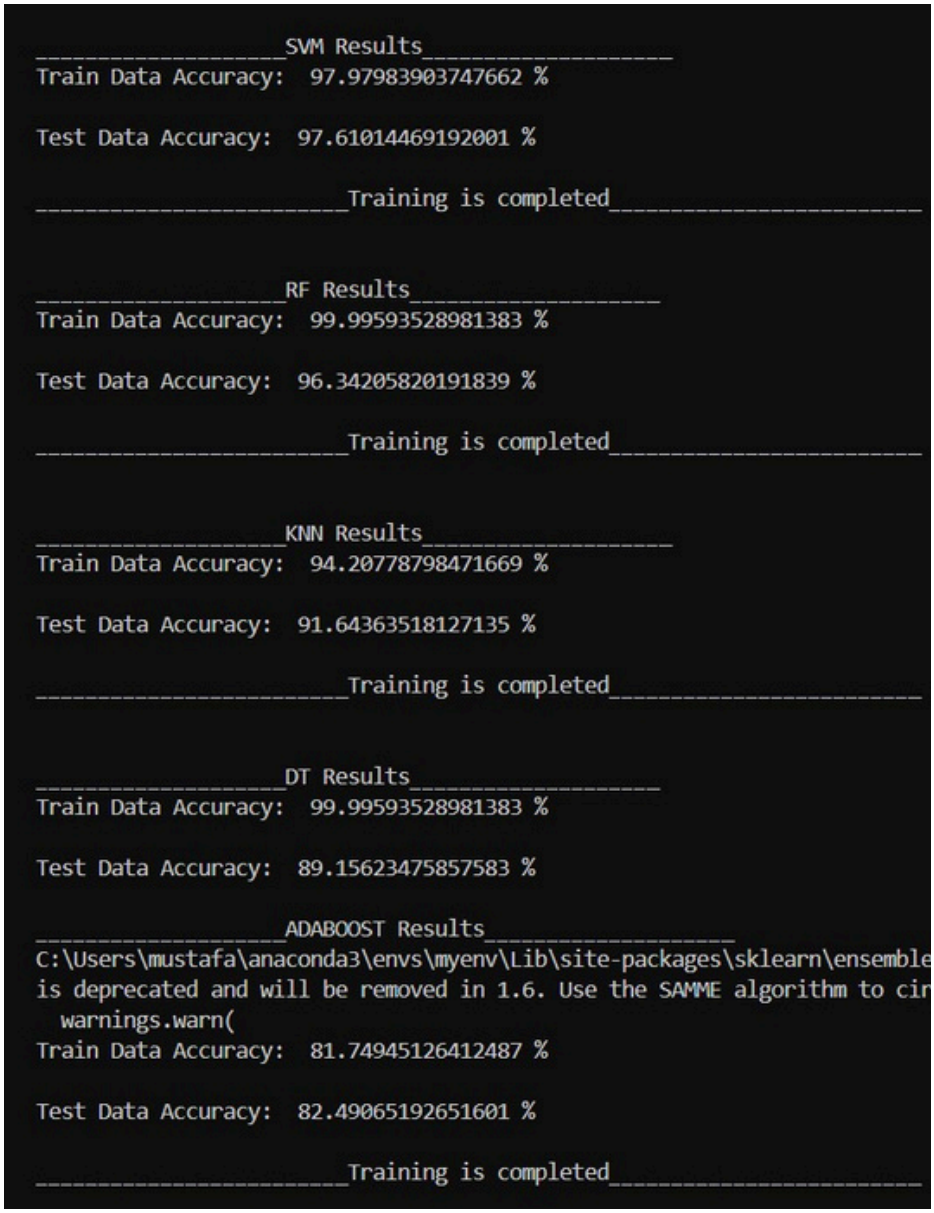
Performance Analysis

We made 5 classifiers which are :

- SVM
- Random Forest
- KNN
- Decision Tree
- Ada Boost

which gives me the following accuracies after running on the dataset using 20% validation.

So SVM gives me the highest accuracy which is **97.6%** so we chose it as our classifier.



Enhancements and Future work

There are more enhancements and future work for this project we could do like:

1. Feature Engineering and Selection:
    a. Experiment with different feature extraction techniques beyond LPQ, such as Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), or Convolutional Neural Networks (CNNs).
    b. Explore advanced feature selection methods to identify the most discriminative features for classification, such as Recursive Feature Elimination (RFE), Principal Component Analysis (PCA), or Feature Importance ranking.
2. Model Improvement:
    a. Investigate other classification algorithms beyond SVM, like Gradient Boosting Machines, or deep learning models such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs). These models might capture complex relationships in the data more effectively.
    b. Tune hyperparameters of SVM and other classifiers using techniques like grid search or Bayesian optimization to find the optimal settings for your data.
3. Data Augmentation and Expansion:
    a. Augment our dataset by introducing variations in the existing samples, such as rotations, translations, scaling, or adding noise. This can help improve the robustness of our model to variations in font styles and conditions.
    b. Consider collecting more diverse data, including samples from additional Arabic fonts or variations in font styles to make our model more generalizable.
4. Ensemble Methods:
    a. Implement ensemble learning techniques such as bagging, boosting, or stacking to combine multiple classifiers trained on different subsets of data or with different algorithms. Ensemble methods often lead to improved performance and robustness compared to individual classifiers.
5. Deployment and Integration:
    a. Develop a more user-friendly interface or application to deploy our classifier, allowing users to input images of Arabic text and obtain predictions of the font type.

Workload Distribution

Preprocessing	**Mustafa Tarek & Daniel**
Feature Extraction	**Mohammed yasser & Kady**
Models Training and Prediction & Analysis	**Mustafa Tarek & Mohammed Yasser & Daniel**
Model Deployment & Post Request	**Kady**