**Department of Computer Engineering**



**Cairo University**

**Faculty of Engineering**

Spring 2024

**Pattern Recognition and Neural Networks‬‏**

**Project Document**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Sec** | **BN** | **Code** |
| Abdulrahman Mohammed Abdulfattah | 1 | 22 | 9210587 |
| Osama Saleh Farag | 1 | 12 | 9210195 |
| Ibrahim Tarek Abdelazeem Amin | 1 | 3 | 9210014 |
| Amr Salahuddin Fouad Yusuf | 1 | 29 | 9210774 |

Table of Contents

[Project Pipeline 2](#_Toc166363880)

[1. Image Loading 2](#_Toc166363881)

[2. Preprocessing Module 2](#_Toc166363882)

[3. Feature Extraction Module 3](#_Toc166363883)

[4. Model Selection 3](#_Toc166363884)

[5. Performance analysis 3](#_Toc166363885)

[Other work 4](#_Toc166363886)

# Project Pipeline

A diagram of a software company

Description automatically generated

Our project pipeline was divided into the following stages:

## Image Loading

Images loading module was responsible for loading the fonts dataset and dividing it into the training set and testing set.

## Preprocessing Module

The image preprocessing module was responsible for the following:

* Noise Removal

We noticed that some images contain S&P noise, so we apply median filter to remove the noise, we also apply some gaussian blur to remove minor noise.

* Image Enhancement

We apply some image enhancement techniques such as making the background always have the same color, and making the text always appear in the same color regardless of the image itself, we also considered fixing the image rotation, but after using SIFT it was pointless.

* Segmentation

We convert the image into binary image to allow simple handling of the image and to be able to extract better features from the image.

## Feature Extraction Module

For selecting our features, we considered the following:

* Letter width
* Font Thickness
* Baseline % of letter height
* Letter % of word width
* Approximated font size
* Curvature of letters
* SIFT (#words count = 360)
* SURF
* HoG of each letter

But after testing we found that SIFT yields the best results.

## Model Selection

We tested the following classifiers:

* 1. SVM

which didn’t reach any high prediction accuracy percentage.

* 1. Adaboost (#weak learners = 100)

Adaboost can learn complex decision boundaries,

making it a good choice for data that isn't easily separated by a straight line. SVMs typically require a non-linear kernel function to handle such data, which can add complexity, also it excels at combining multiple "weak learners" (like decision stumps) into a stronger ensemble classifier. This can outperform a single SVM, especially if the base learners are well-suited to the problem.

Adaboost resulted in a very good results when paired with SIFT feature, we got results up to 99.5% accuracy.

* 1. XGBoost (#weak learners = 100)

Like Adaboost, XGBoost also yielded very good results for test cases we run onto.

We submitted both the Adaboost and XGBoost module.

## Performance analysis

After testing potential classifiers, we got the following performance analysis:

* SVM: 47.8% accuracy
* Adaboost: 99.5% accuracy
* XGBoost: 99.2% accuracy

# Other work

The SIFT features with Adaboost was not our first idea, we also tried to segment each letter (like how OCRs work) and try to match each letter to our train set letters, then we calculate the collective probability that the image is from “X” class, but the following idea failed due to various reasons, but we also submitted the code related to the idea (but it’s not fully implemented)