Arabic Font Recognition

Authors

Ahmed Hany AbdelWahab AbdelGawad

Ahmed Mohamed Ahmed Abdeaal

Marwan Mohamed Abbas

Mohamed Ahmed Mohamed Abdelmoniem

Submitted to:

Eng. Essam Wissam

Eng. Mohamed ElSayed

Contents:

1 Project Pipeline

The project's pipeline is a multi-step process designed for image classification tasks, specifically focusing on extracting features from images and using a Support Vector Machine (SVM) model for classification. Let's break down the pipeline into its key components and explain their roles in the overall process.

1.1 Image Preprocessing

The first step involves preprocessing the input images to enhance their quality and make them suitable for feature extraction. This includes color fixing, binarization, orientation detection, and region cropping. Color fixing ensures that the text is darker than the background, crucial for accurate feature extraction. Binarization converts the image into a binary format, simplifying the data for subsequent processing. Orientation detection identifies the angles of text orientations, aiding in standardizing image orientation. Region cropping isolates the text region for focused feature extraction.

1.2 Feature Extraction

Once the images are preprocessed, the next step is feature extraction. This step is crucial for capturing meaningful information from the images that can be used for classification. The feature extraction process includes extracting SIFT descriptors, BoVW (Bag of Visual Words) features, Gabor features, and Laws texture energy measures. Each type of feature captures different aspects of the image, such as texture, shape, and spatial information, providing a comprehensive representation of the image content.

1.3 Feature Scaling and Dimensionality Reduction

After feature extraction, the features are scaled using Min-Max scaling to ensure uniformity across different feature scales. Additionally, highly correlated features are identified and dropped to reduce redundancy and improve model performance. These preprocessing steps prepare the features for input into the classification model.

1.4 Classification with SVM Model

The final stage of the pipeline involves using an SVM model for image classification. SVM is a supervised learning algorithm capable of handling both linear and non-linear classification tasks. The model is trained on the preprocessed and scaled features to learn patterns and make predictions on new images. The SVM model used in this pipeline is trained on labeled data to classify images into predefined categories.

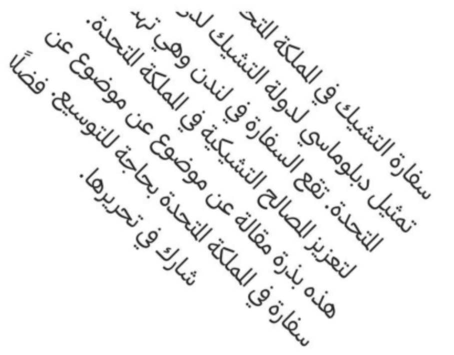
Overall, the pipeline encompasses image preprocessing, feature extraction, feature scaling, dimensionality reduction, and classification using an SVM model. Each step contributes to transforming raw image data into a format that the classification model can understand, leading to accurate and reliable image classification results suitable for various applications like object recognition, scene understanding, and more.

2 Preprocessing Module

The preprocessing steps outlined in the provided code are designed to enhance the quality of images containing text for better readability and processing. The process begins with fixing the color contrast to ensure that the text appears darker than the background, which is crucial for accurate text detection algorithms. This step involves inverting the image colors if the average color of the border pixels suggests a light background, making the text more distinguishable.

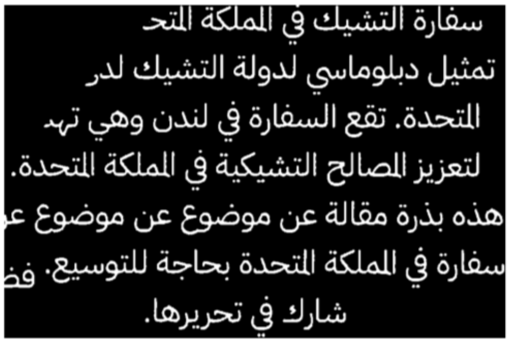
Following color fixing, the image undergoes binarization through median blurring and Otsu's thresholding techniques. These operations help to reduce noise and create a clear binary image where text regions are emphasized against the background. Additionally, applying morphological closing further refines the image by connecting text regions and making contours more detectable, setting the stage for accurate orientation detection.

The next crucial step is detecting text orientations using minimum area rectangles around contours. This involves computing the orientation of each rectangle and adjusting angles for a consistent representation. Visualizing these rectangles overlaid on the image aids in understanding the orientation distribution, which is further analyzed through a histogram of angles. This histogram provides insights into the dominant text orientation, crucial for subsequent image rotation.



سفارة التشيك في المملكة المتحـ
تمثيل دبلوماسي لدولة التشيك لد
المتحدة. تقع السفارة في لندن وهي تهـ
لتعزيز المصالح التشيكية في المملكة المتحدة.
هذه بذرة مقالة عن موضوع عن موضوع عن
سفارة في المملكة المتحدة بحاجة للتوسيع. فضلًا
شارك في تحريرها.

Figure 1: Unprocessed Image



سفارة التشيك في المملكة المتحـ
تمثيل دبلوماسي لدولة التشيك لدر
المتحدة. تقع السفارة في لندن وهي تهـ
لتعزيز المصالح التشيكية في المملكة المتحدة.
هذه بذرة مقالة عن موضوع عن موضوع عـ
سفارة في المملكة المتحدة بحاجة للتوسيع. فـ
شارك في تحريرها.

Figure 2: Processed Image

The final stages of preprocessing involve finding the best rotation angle based on the histogram analysis and applying the rotation to the image. This step aligns the text in a standardized orientation for easier processing downstream. Additionally, the process identifies and crops the region containing the largest bounding rectangle, further refining the image to focus solely on the text area, which is vital for text extraction and recognition tasks.

Overall, this comprehensive preprocessing pipeline ensures that the text in images is optimally prepared for subsequent text detection, extraction, and recognition algorithms, ultimately improving the accuracy and efficiency of text-based image processing workflows.

3 Feature Extraction and Selection

Having preprocessed the images, we now move on to extracting useful features from them. We have not one but three different feature extractors that attempt to capture different aspects of the images. Starting with **Gabor filters**, which is long known for its use in these contexts, we then move on to **SIFT Features** that are described using **BoVW**(Bag of Words) and the cherry on top is the **Law's Texture Energy Measures**.

3.1 Gabor Filters

We first start by building the Gabor filters using different orientations, frequencies and sigmas. The Gabor filter is defined as follows:

$$g_{\gamma,\lambda,\theta,\phi}(x,y) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\phi\right) \tag{1}$$

$$x' = x\cos\theta + y\sin\theta \tag{2}$$

$$y' = -x\sin\theta + y\cos\theta \tag{3}$$

Next, we convolve the built Gabor filters to obtain the Gabor responses. This produces feature maps that are then used to calculate the **energy** of the responses.

$$r_{\gamma,\lambda,\theta,\phi}(x,y) = \int\int I(\varepsilon,\eta)g(x-\varepsilon, y-\eta)d\varepsilon d\eta \tag{4}$$

$$E_{\gamma,\lambda,\theta}(x,y) = (g^2_{\gamma,\lambda,\theta,0}(x,y) + g^2_{\gamma,\lambda,\theta,-\frac{\pi}{2}}(x,y))^{1/2} \tag{5}$$

To obtain the final feature vector, we first derive a **max image** across the feature maps and proceed to calculate the μ and σ of the feature maps and the max image. Finally, we concatenate the μ and σ values to obtain a feature vector of size $2 \times$ **orientations** $\times$ **frequencies**.

3.2 SIFT Features

The sift features describe an area around the keypoints which are usually corners of interest as such these would usually occur around letters and since a descriptor describes and area around a keypoint it will basically describe the letter, the only issue is the descriptor number is different, taking a mean or average is meaningless for descriptors.

However what is useful is counting the similar descriptors since every font would produce specific descriptors to letters, so to produce a good measurement of similarity is using kmeans to cluster these centroids, and based on the number of descriptors lying in each cluster we decide what class we are.

3.3 Law's Texture Energy Measures

In this study, we explored the use of Law's Texture Energy Measures for image feature extraction. This method involves applying a set of small convolution masks, known as Law's masks, to the image. Each mask is designed to respond to a specific type of texture in the image, such as level spots, edges, spots, waves, or ripples.

The masks are created by taking the outer product of two vectors, which capture certain textural properties. Each Law's mask is then convolved with the image, calculating a weighted sum of the pixels under the mask at each position. The weights are given by the values in the mask.

The output of the convolution is squared and then averaged over a larger window to calculate the texture energy. This gives a measure of how much of the texture corresponding to each mask is present in the image. The texture energy measures for all masks are combined to create a feature vector for each pixel or region in the image.

$$C(x,y) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} I(x-i, y-j)M(i,j) \tag{6}$$

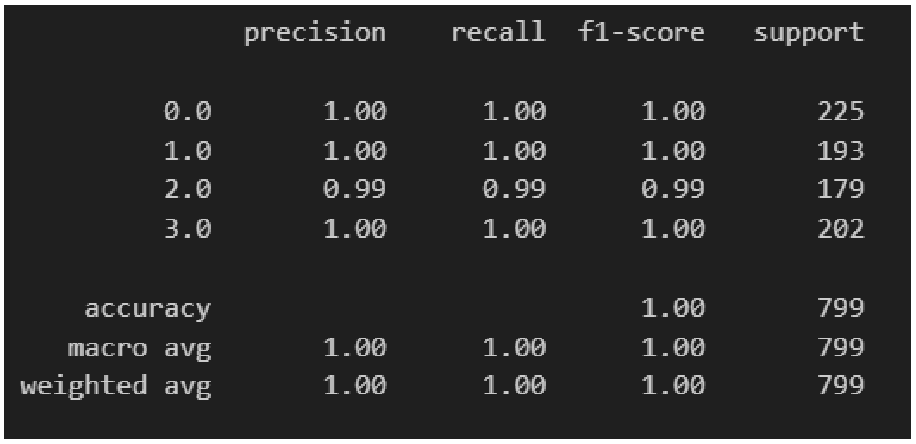
$$E = \frac{1}{N} \sum_{x,y} C(x,y)^2 \tag{7}$$

This method is particularly effective for textures that are defined by local intensity variations, and it's robust to changes in illumination and contrast. However, it's less effective for textures that are defined by specific spatial relationships between pixels, and it can be computationally intensive due to the need to convolve the image with multiple masks.

$$F = [E1, E2, ..., En] \tag{8}$$

4 Model Selection and Training Module

4.1 Logistic Regression

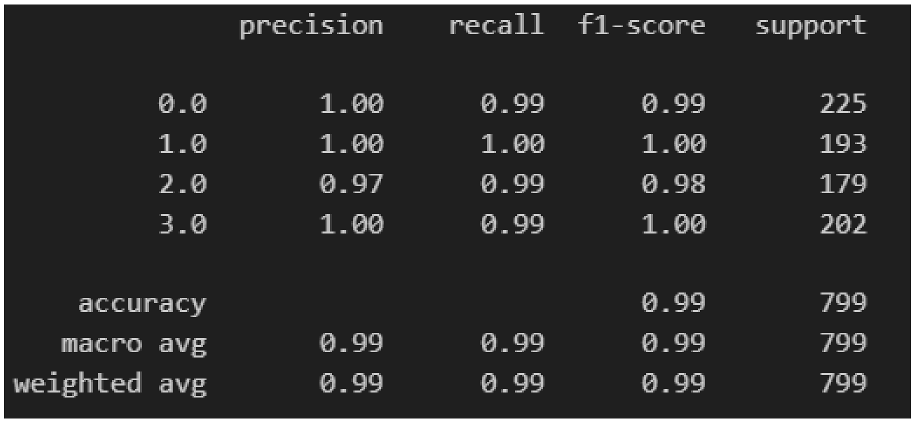


```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       225
         1.0       1.00      1.00      1.00       193
         2.0       0.99      0.99      0.99       179
         3.0       1.00      1.00      1.00       202

    accuracy                           1.00       799
   macro avg       1.00      1.00      1.00       799
weighted avg       1.00      1.00      1.00       799
```

4.2 Linear Discriminant Analysis

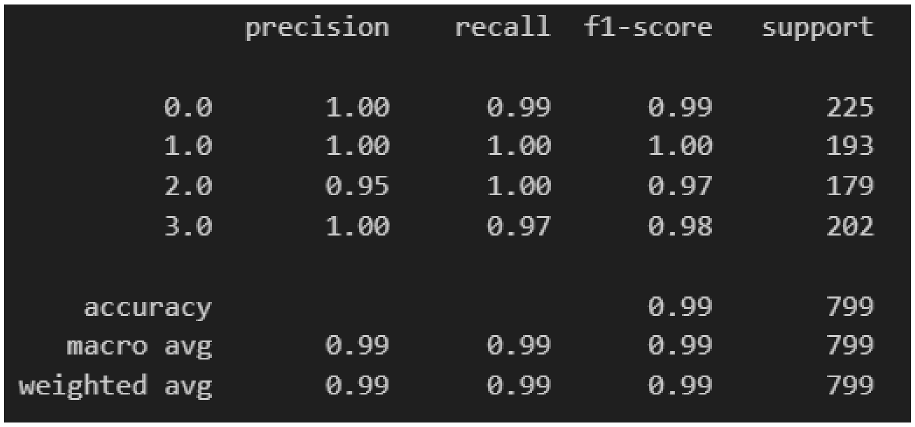


```
              precision    recall  f1-score   support

         0.0       1.00      0.99      0.99       225
         1.0       1.00      1.00      1.00       193
         2.0       0.97      0.99      0.98       179
         3.0       1.00      0.99      1.00       202

    accuracy                           0.99       799
   macro avg       0.99      0.99      0.99       799
weighted avg       0.99      0.99      0.99       799
```

4.3 Quadratic Discriminant Analysis

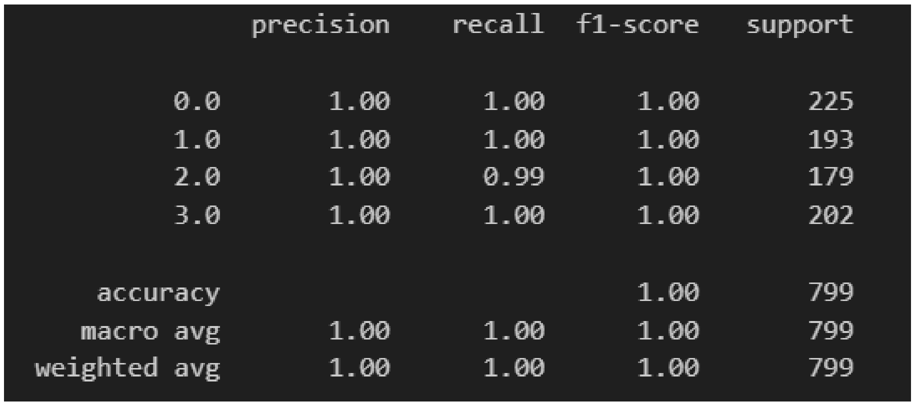


```
              precision    recall  f1-score   support

         0.0       1.00      0.99      0.99       225
         1.0       1.00      1.00      1.00       193
         2.0       0.95      1.00      0.97       179
         3.0       1.00      0.97      0.98       202

    accuracy                           0.99       799
   macro avg       0.99      0.99      0.99       799
weighted avg       0.99      0.99      0.99       799
```

4.4 Support Vector Machine

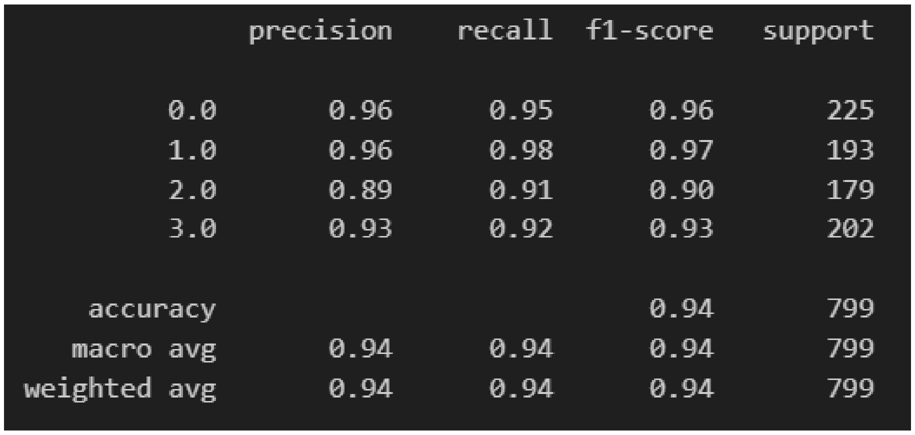


```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       225
         1.0       1.00      1.00      1.00       193
         2.0       1.00      0.99      1.00       179
         3.0       1.00      1.00      1.00       202

    accuracy                           1.00       799
   macro avg       1.00      1.00      1.00       799
weighted avg       1.00      1.00      1.00       799
```

4.5 Decision Tree Classifier

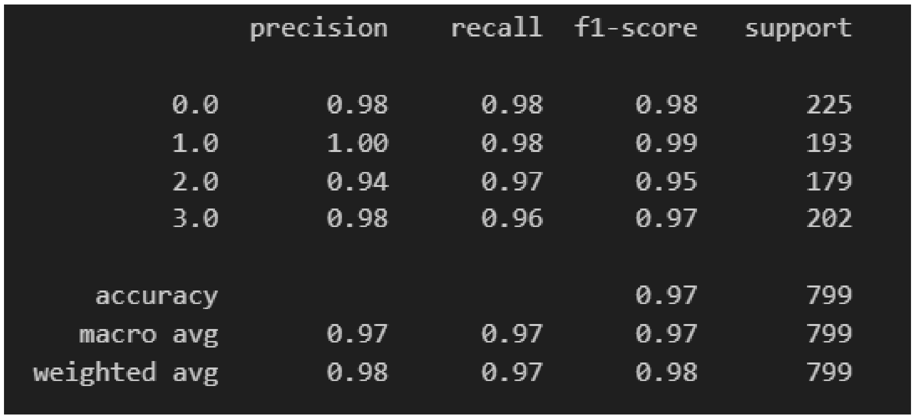


```
              precision    recall  f1-score   support

         0.0       0.96      0.95      0.96       225
         1.0       0.96      0.98      0.97       193
         2.0       0.89      0.91      0.90       179
         3.0       0.93      0.92      0.93       202

    accuracy                           0.94       799
   macro avg       0.94      0.94      0.94       799
weighted avg       0.94      0.94      0.94       799
```

4.6 XGBoost Classifier

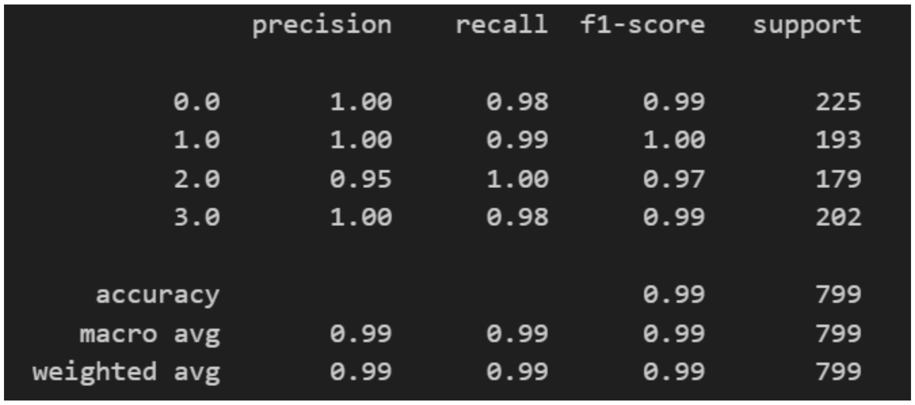


```
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98       225
         1.0       1.00      0.98      0.99       193
         2.0       0.94      0.97      0.95       179
         3.0       0.98      0.96      0.97       202

    accuracy                           0.97       799
   macro avg       0.97      0.97      0.97       799
weighted avg       0.98      0.97      0.98       799
```

4.7 Random Forest Classifier

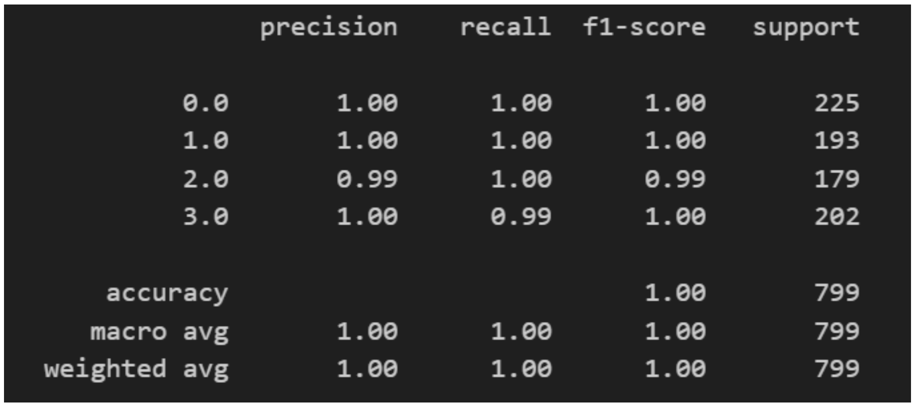


```
              precision    recall  f1-score   support

         0.0       1.00      0.98      0.99       225
         1.0       1.00      0.99      1.00       193
         2.0       0.95      1.00      0.97       179
         3.0       1.00      0.98      0.99       202

    accuracy                           0.99       799
   macro avg       0.99      0.99      0.99       799
weighted avg       0.99      0.99      0.99       799
```

4.8 MLP Classifier



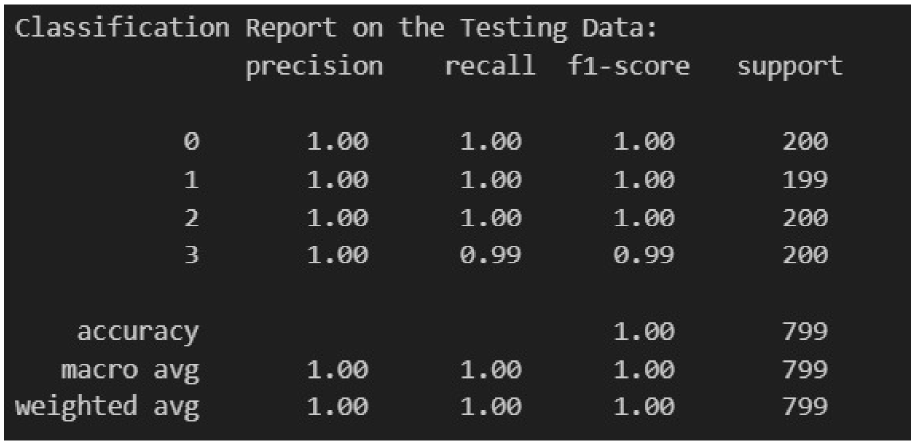
```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       225
         1.0       1.00      1.00      1.00       193
         2.0       0.99      1.00      0.99       179
         3.0       1.00      0.99      1.00       202

    accuracy                           1.00       799
   macro avg       1.00      1.00      1.00       799
weighted avg       1.00      1.00      1.00       799
```

5 Performance Analysis Module

To put our model to the test we set aside 200 images from each class and they were tested at the very end to determine the accuracy of the model in each class respictively. This gave us insights into which class/classes actually affects our performance.



```
Classification Report on the Testing Data:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       200
           1       1.00      1.00      1.00       199
           2       1.00      1.00      1.00       200
           3       1.00      0.99      0.99       200

    accuracy                           1.00       799
   macro avg       1.00      1.00      1.00       799
weighted avg       1.00      1.00      1.00       799
```

As evident from the recall in the third row, the only class which had errors was class 3 **Scheherazade New** and thus was the only one to not have 100% accuracy.

6 Enhancements and Future Work

Although the model is already performing well, there are still several areas to improve upon for maximizing the model's performance. Hence, we will explore some possible enhancements and ideas.

6.1 Preprocessing Enhancements

The preprocessing module can be improved by applying something that we found in a research paper that it would detect the text in an image using image processing techniques and then repeat that text in the empty portion of the image, thus making it have more data and hence more obvious results.

Moreover, repeating the text can result in more defined clusters and hence better results.

6.2 Finding Optimum Hyperparameters

Due to the long and time-consuming task of training the kmeans classifier we were unable to fully determine the best **k** to use. Consequently, we thought of utilizing the **elbow method** to determine the optimal value and achieve so by repeated training on different values of k.

6.3 Generating More Data

A cherry on top to our training phase was to generate more data using **GANs** or **Diffusion Models** to boost our training and validation sets. This boosting results in a larger collection of features that can allow for higher performance and enables us to explore even more complex models.

6.4 Explore More Complex Models

Generating or aquiring more data can enable us to test more models, such as CNNs, and learn more complex filters.