Introduction:

In the digital age, the abundance of visually appealing content has led to an ever-growing collection of wallpapers. Over the years, I have manually curated a diverse gallery of the digital art-style images, meticulously selecting each one based on its unique aesthetic appeal. However, as the number of images has grown, managing it also becomes more complex.

One particular challenge that has emerged is the inadvertent downloading of duplicate or near-identical images. Given the vastness of the internet and the sheer volume of visually similar content, it is increasingly difficult to remember every image that has been downloaded. This not only leads to unnecessary redundancy in the dataset but also consumes valuable storage space and manual effort in identifying and removing these duplicates.

To address this issue, I designed the program to identify similar images within my gallery. This tool will streamline the management of the collection by preventing the download of duplicate images, thereby ensuring the uniqueness of each wallpaper. This initiative represents a significant step towards efficient data management, and paving the way for more focused curation efforts in the future.

Data preparation:

So, I have this dataset, which is a unique collection of 1209 art-drawing-style, which each has been carefully curated over the time, reflecting a wide range of artistic expressions and styles.

The original resolution and type of these images vary, as they have been sourced from diverse platforms and mediums. However, as to maintain the consistency and efficient processing, all the images have been converted to a standard resolution of 240x426 pixels and saved in the universally accepted JPEG format.

It’s important to note that the images in this dataset are unlabeled. This presents an exciting opportunity for unsupervised learning tasks, such as clustering or anomaly detection, where the inherent patterns and structures within the data can be explored without preconceived notions.

Model implementation

The implementation of this project involves the use of five different models:

* a custom CNN autoencoder:
* VGG19,
* Resnet50V2,
* InceptionV3,
* And Xception.

Where the last four models are pre-trained models, widely recognized for their performance in image classification tasks.

* CNN

About the custom CNN autoencoder model, it is trained on the features extracted from dataset.

In the Encoder part, the model takes input data and compresses it into a lower-dimensional representation.

The decoder part of the autoencoder takes the compressed representation and reconstructs the original input data. The decoder mirrors the encoder in terms of layer structure.

* Pretrained

The last 4 pretrained models are used for comparing the results to the CNN.

Along the process, the input image is converted into a numpy array and then passed through each of these models. The output is a set of features that capture the essential characteristics of the image.

* Cluster

After the feature extraction process, the next step is to use K-Nearest Neighbor clustering algorithm to group similar images together. In the context of our task, we use k-NN with the number of nearest neighbors is 6 for its ability to find the ‘nearest’ neighbors in the feature space. The ‘nearest’ images are considered the most similar to the input image.

* Evaluating models using metrics:

To evaluate the effectiveness of our model in identifying similar images, we use three key metrics, where:

* Cosine Similarity measures the cosine of the angle between two vectors. In our case, these vectors are the feature vectors of two images. A smaller angle (and thus a larger cosine similarity) indicates more similar images
* Structural Similarity Index (SSIM) to considers changes in texture, brightness, and contrast when determining similarity, making it a more comprehensive measure than simple pixel-based comparisons.
* Histogram Similarity to compares the color histograms of two images. A color histogram represents the distribution of colors in an image. By comparing these distributions, we can get a sense of how similar two images are in terms of color composition.

Thus, capture different aspects of image similarity and thereby obtain a more holistic evaluation of model’s performance.

* Show result image
* Compare results here:

Running time

The Inceptionv3 model has the fastest feature extraction time at 6.34 seconds, which is significantly faster than the other models. This could make it a preferred choice for applications where speed is a critical factor.

The VGG19 model has the fastest prediction time at 0.79 seconds, despite having a relatively high feature extraction time. This suggests that once the features are extracted, VGG19 can make predictions more quickly than the other models.

The ResNet50V2, Inceptionv3, and Xception models all use a feature shape of (8, 14, 2048) or (6, 11, 2048), which is larger than the feature shape used by VGG19. This could potentially result in more detailed feature representations, but at the cost of increased computational complexity.

The CustomCNN model has the highest prediction time at 3.45 seconds, which is considerably slower than the other models. This might make it less suitable for applications that require real-time predictions.

* Compare clustering here:
* Compare metrics:

Based on the provided table, we can conclude the following:

* ResNet50V2 and CustomCNN models have the highest average cosine similarity scores, both approximately 85%, indicating that these models are most effective in terms of cosine similarity when using Euclidean distance.
* The CustomCNN model has the highest average structural similarity score at 37.09%. This suggests that this model is more capable of preserving the structural information of the images.
* The Xception model has the highest average histogram similarity score at 50.92%, indicating that this model is most effective in terms of histogram similarity.
* The VGG19 model has the lowest scores across all three metrics, suggesting that it might be less effective for this particular task compared to the other models.