UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

Organize Wallpaper Images Based on Similarity

Machine Learning

Lecturer: Dr. DOAN Nh-at Quang

Team members:

Pham Gia Phuc – M23.ICT.010
Nguyen Dang Minh – M23.ICT.008

April 10, 2024

Contents

1. INTRODUCTION

1.1. Context

In the digital age, the sheer volume of captivating visual content has given rise to an expansive accumulation of wallpapers. This collection, which is personally and painstakingly gathered over the years, represents a big selection of digital art-fashion imagery, each has been chosen with great care, making sure that it stands out for its exceptional aesthetic qualities.

However, with the growing size and complexity of the wallpaper collection, dealing with and organizing every image has become increasingly challenging wherein newly downloaded wallpapers might be duplicate or near-identical to the existing ones. This not only leads to unnecessary redundancy but also consumes extra valuable storage space and manual attempts to keep the individuality of each image.

1.2. Objective

To address this issue, we have designed a machine learning program specifically to check and examine the dataset of wallpapers, using the unsupervised learning techniques to detect and quantify the similarity between images, taking into account the identification of potential duplicates. Thereby, the system will initiatively represent a significant step towards efficient data management, and paving the way for greater focused efforts in the future.

2. Materials And Methods

2.1. Materials

The custom Wallpaper dataset is a completely unique collection of 1209 multi-style images, most of which are tributes to the craft of digital drawing. They cowl a huge variety of creative expressions and patterns, which have been carefully curated and assembled over a long time period. The collection sourced from a wide range of platforms and mediums, as a result, there are many extraordinary unique types and resolutions, including Joint Photographic Experts Group (JPEG), Portable Network Graphics (PNG) and JPEG File Interchange Format (JFIF).

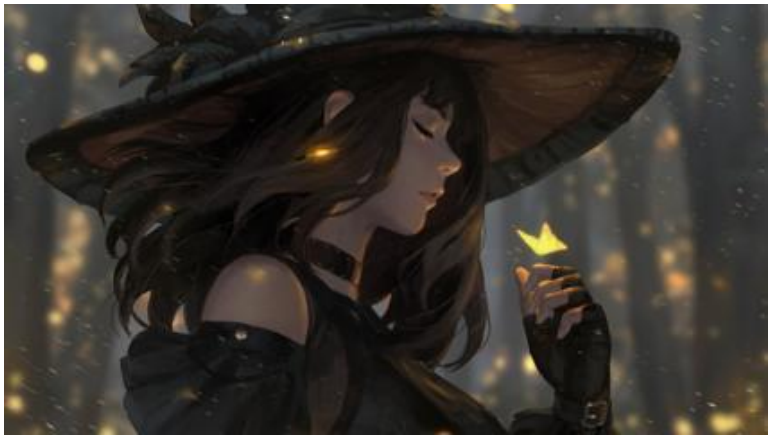

Figure 1. An image inside dataset

In order to offer a steady standard that helps consistency and speed in processing, every image has been transformed into the widely used JPEG format with resolution of 240x426 pixels, (16:9) which ensures ease of use and compatibility with a whole lot of applications and hardware.

Also, it is important to emphasize that this dataset's images are unlabeled.

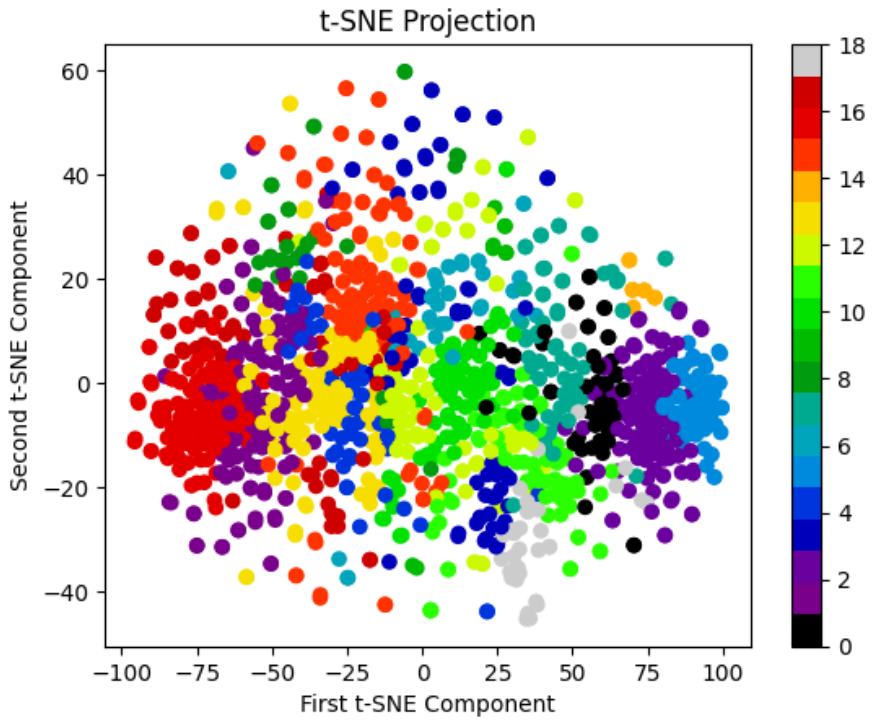

Figure 2. Visualize the scattering image data using k-Means algorithm

2.2. Methods

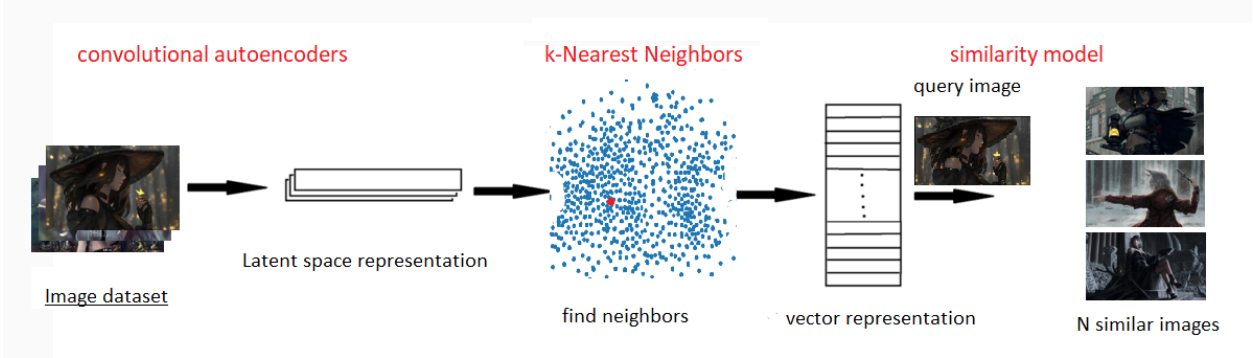

Figure 3. Experiment for finding similar image

Our experiment process consists of four parts. Firstly, we initiate the process by taking the filename of the image and converting it into an array representation using suitable image processing libraries. This involves reading the image file and converting it into an array format compatible with the trained Convolutional Neural Network (CNN) model.

Secondly, we utilize the array representation of the image to extract features through the trained CNN model. Each image is processed through the CNN, resulting in a set of high-level features that capture its distinctive characteristics.

Subsequent to the feature extraction phase, the K-Nearest Neighbor (k-NN) clustering algorithm comes into play, with the objective is to organize comparable images into clusters. Within the scope of this project, the k-NN algorithm is configured with six nearest neighbors, a setting chosen for its proficiency in identifying the closest counterparts in the feature space. Images deemed 'nearest' are thus considered most akin to the input image, facilitating the grouping of like images.

Finally, we visualize the results by plotting the N similar images alongside the query image. This visualization provides a comparative view, allowing users to observe the similarities and differences between the query image and its nearest neighbors.

2.2.1. Model implementation

There are five different models used in the project's implementation for feature extraction of the image data. This suite consists of four well-known pre-trained models in imagenet dataset: VGG19, ResNet50V2, InceptionV3 and Xception, in addition to our custom Convolutional Neural Network (CNN) Autoencoders.

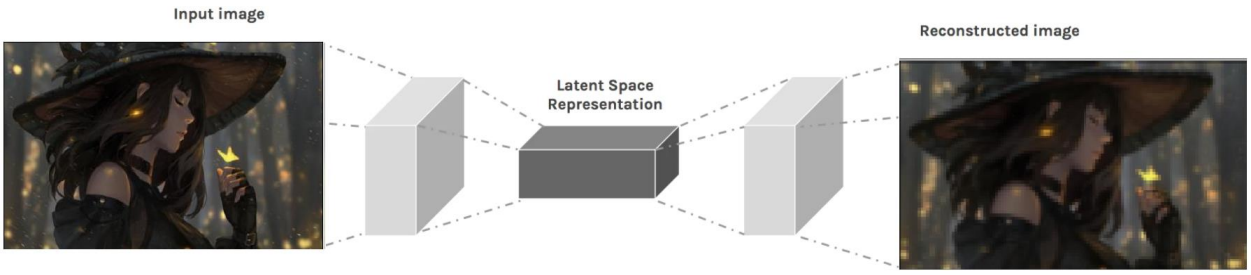

Figure 4. Convolutional AutoEncoders process

Focusing on our custom CNN Autoencoder, it undergoes training utilizing features derived from the dataset. The encoder segment of this model is responsible for condensing the input data into a more compact, lower-dimensional form, where conversely, the decoder segment works to reconstruct the input data from this compressed form, effectively mirroring the encoder's architecture in reverse. In this experiment, we only use the encoder segment for getting the feature set of our dataset.

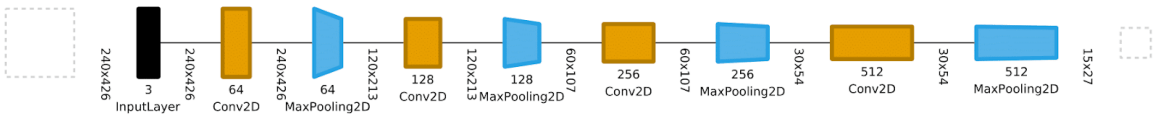

Figure 5. Convolutional AutoEncoders Structure

On the other hand, the four pre-trained models perform a comparative role by comparing the outcomes with the custom CNN autoencoder. The input pictures are initially converted into

numpy arrays all through the processing stage. These arrays are then sequentially fed through each model, resulting in a collection of features that encapsulate the core attributes of the images.

2.3. Evaluating

To thoroughly evaluate the efficacy of our model in discerning similar images, we employ a set of three metrics, each designed to capture different dimensions of similarity between images.

To start with, Cosine Similarity, which calculates the cosine of the angle between two vectors in a multi-dimensional space, which represent the feature vectors extracted from the images. A higher cosine similarity score, which corresponds to a smaller angle between the vectors, signifies a greater similarity between the images in question. This metric is particularly useful for figuring out how well the visual content of two images aligns.

$$\cos\theta = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

The Structural Similarity Index (SSIM) is then included in our evaluation procedure to evaluate texture, brightness, and contrast variations when determining how similar two images are to one another. Unlike simpler metrics that may only compare pixel values, SSIM provides a comprehensive analysis by evaluating the visual impact of these attributes on perceived image similarity, making it a more comprehensive measure than simple pixel-based comparisons.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Lastly, Histogram Similarity, which shows the distribution and frequency of colors inside each image, plays a crucial role in the assessment framework. By analyzing these histograms, it is able to determine the degree to which two photographs have a comparable color scheme, thereby offering a numerical assessment of similarity based on color. This is particularly important in cases where color distribution plays a key role in the overall composition and aesthetic of the images.

For evaluation purposes, we selected a subset of 120 images from the dataset to form our test set. Each image from this test set is used as a query input to retrieve similar images from the dataset using our retrieval system. Upon querying, we obtain 11 similar images for each query image based on their feature similarities. To ensure accurate evaluation, we exclude the first retrieved image from each query set, as it corresponds to the input image itself. Subsequently, we calculate the average of three above metrics across all query images. This evaluation process allows us to assess the effectiveness and performance of our image retrieval system comprehensively.

3. RESULTS

3.1. Query result

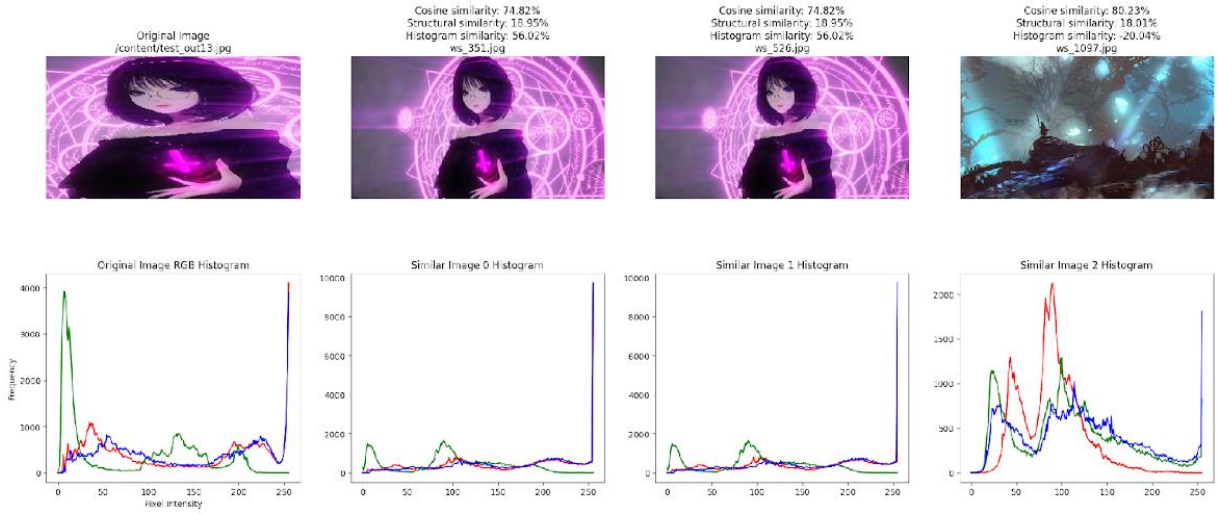

Figure 6. Visual representation of the input image alongside its three most similar images from the dataset and its corresponding histogram using our Convolutional Autoencoders

The input image is a different version of some image inside the dataset, where it has different dimensions and scale. In spite of those differences, our models have been able to recognize the original photograph by identifying its excessive-level features. When compared to the input image, the dataset image named *ws_351.jpg*, shows a rather excessive cosine similarity (74. 74.82%) and histogram similarity (56.02%). One interesting thing to note is our investigation also turned up a duplicate in the dataset. Although lacking the context, the input's histogram distribution closely matches that of the one inside the dataset, indicating similarity in color composition. However, the third image shows a pretty high cosine similarity despite notable variances in things portrayed, which possible explanations for this resemblance include similar color patterns and drawing styles. Nevertheless, it is vital to consider that since the query image is a creative portrayal, excessive similarity ratings could not continually appropriately constitute similarities between actual-world items.

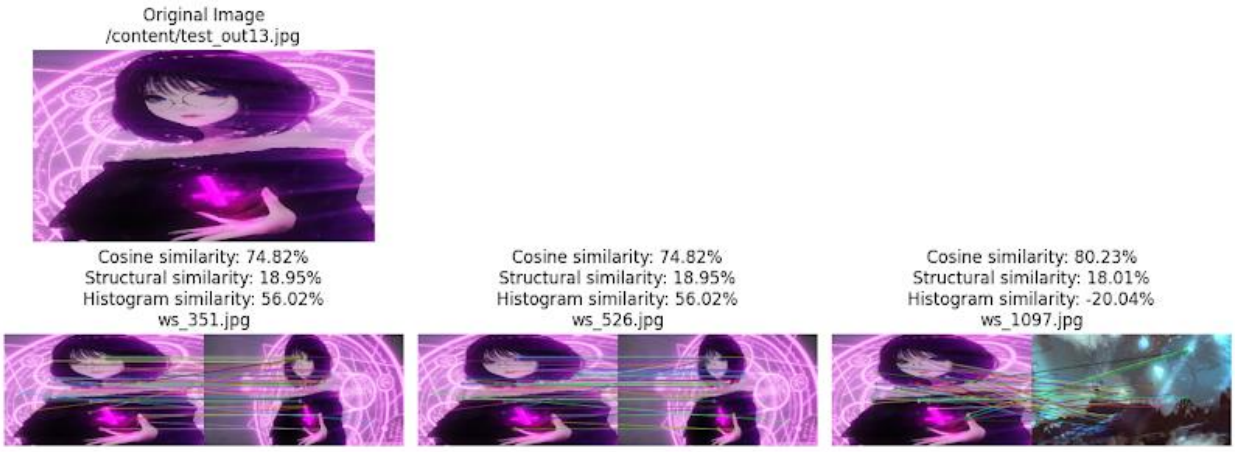

Figure 7. Visual representation of the similar features detected in the input image and the resultant images retrieved from the dataset. Parallel lines in the highlight denote a higher degree of similarity between two images, visually illustrating the extent of shared features between them

3.2. Evaluation of the feature extraction

3.2.1. Running time

Model (using Euclidian Distance)	Feature Shape	Feature Extraction time	Prediction time
VGG19	(7, 13, 512)	21.31	0.79
ResNet50V2	(8, 14, 2048)	11.09	2.54
Inceptionv3	(6, 11, 2048)	6.25	1.74
Xception	(8, 14, 2048)	21.35	2.75
CustomCNN	(15, 27, 512)	11.29	1.45

Table 1. Models' running time

It is obvious that speed is important when assessing the feature extraction models' performance measures. With a balance between accuracy and efficiency, Inceptionv3 is the fastest model evaluated in terms of each feature extraction and prediction times. Interestingly, despite the fact that our feature extraction-optimized Convolutional Autoencoder takes a bit longer to extract capabilities, it plays competitively, especially in prediction time.

While established models like VGG19, ResNet50V2, and Xception offer robust feature representations, and their computational speed may impact real-time applications as well, our CustomCNN model showcases promising potential as a viable alternative, providing comparable results to popular CNN architectures while exhibiting enhanced efficiency in prediction tasks.

3.2.2. Similarity metrics

Model (using Euclidian Distance)	Average Cosine Similarity (%)	Average Structural Similarity (%)	Average Histogram Similarity (%)
VGG19	63.64	26.01	37.37
ResNet50V2	84.99	34.53	48.71
Inceptionv3	81.71	32.18	48.59
Xception	80.89	31.38	50.92
CustomCNN	76.06	24.13	29.48

Table 2. Models' metric evaluation

The analysis of similarity metrics reveals ResNet50V2 as the standout performer among the evaluated models utilizing Euclidean distance. Notably, it achieves the highest average cosine, structural, and histogram similarities, while InceptionV3 and Xception carefully follow, demonstrating competitive performance throughout similarity measures. Although our Convolutional Autoencoders yield lower results throughout all metrics as compared to different models, they nonetheless gain satisfactory outcomes.

4. CONCLUSION

In this project, we have crafted a machine learning program tailored specifically for examining a dataset of wallpapers, leveraging unsupervised learning techniques. We have aimed to come across and quantify photo similarity inside the dataset. Additionally, we have developed a custom CNN model for feature extraction, which we have compared with popular architectures. Our findings reveal that, whilst CustomCNN may have barely lower similarity scores, its outstanding efficiency advantages, which include faster feature extraction and prediction times, are vital concerns for real-world applications. It is well worth noting that our dataset accommodates many multi-style images, primarily tributes to digital drawing, thus excessive similarity rankings won't continually correlate with real-world item similarities. Moving forward, we accept as true that there may be capability to enhance our results with a more diverse dataset and refined evaluation metrics.

5. REFERENCES

- https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm
- https://medium.com/better-programming/a-guide-on-the-encoder-decoder-model-and-the-attention-mechanism-401c836e2cdb
- https://medium.com/analytics-vidhya/image-similarity-model-6b89a22e2f1a
- https://viscom.net2vis.uni-ulm.de/47AjRKsIAGbo8UAjUZqSLIDwEawIfTkICSjoneFPHGWP0oSWqP