SIMILAR IMAGES RETRIEVAL IN A LARGE DATABASE

Norman El Fartass
Antonin Condette

Internship report, L3
Supervised par Xavier Bou and Miguel Colom

June 28, 2025

Abstract

IPOL is an image processing journal publishing articles featuring a source code and an online demo for running the algorithm. Over time, a large database of all images tested through the demos has accumulated. Given a new image, one may ask: what are the algorithms the image would be pertinent to try on ? To answer this question, we tried to find similar images in the large data base as each image is associated with a demo. Finding semantically similar images to the original one will then help suggest relevant algorithms. We first extracted embeddings containing the semantic description of the images to compare them easily. As the database is over a million images, we used the clustering algorithm K-Means and reduced the dimension of the embeddings with the PCA. Although results fluctuate between image sets, these conclusive techniques allow us to drastically reduce the computation time for the clustering and the retrieval.

Contents

1 Introduction

In 2009, the IPOL journal was founded at the CMLA (Centre de Mathématiques et de Leurs Applications, presently the Borelli Center). It publishes articles on image processing with the source code, an online demo and an archive containing extensive data and images for each article. Articles can cover topics ranging from facial recognition algorithms to satellite image analysis to medical imaging. All algorithms can be tested online with our own images to see how the code works in action. It provides us with a huge database of over 1 million images related to many different demos.

An interesting question raised is given an image, what are the demos we can assign it to, and which algorithms this image would be pertinent to try on. As each image is assigned to a demo it was tried on, by finding similar images to the one we want to use, it would provide us with one or more relevant demo. One challenge is first to find what makes two images similar. We decided to consider two images to be similar if their semantic descriptions are alike as demos are often based on the content of the images they are linked to.

We now have to find a suitable representation of images taking into account their semantic description to compare them. Traditional methods relied on descriptors such as SIFT, SURF, or ORB, but lacked the ability to capture high-level semantic information, unlike modern models such as CLIP or DINOv2. These methods work by transforming each image into an embedding that captures its semantic description. We can then compare the embedding of a given image with each embedding of all the other images to see which demos we can assign it to. The problem is that, as the database is very large, the computational cost is massive. We must therefore find a way to reduce the computation cost.

The method we will use to solve the problem is the following : we first extract the embeddings of the images containing their semantic description. Then, we reduce the dimension of the embeddings while conserving as much information as possible. Then we will cluster the data to compare each image to categories of concepts reducing drastically the computational time. Finally, we can compare the original image with all the clusters to determine which one is the most relevant and therefore which demos we can link it to.

In this report, we will first discuss in the section 2 the technical methods used to tackle the problem, including the embedding extraction in section 2.1, the representation of data in section 2.2, clustering in section 2.3 and dimensionality reduction in section 2.4. Then we will describe and analyze the results we obtained in section 3 and how to interpret them.

2 Methods

2.1 Embedding extraction

In order to clusterize the images, we need a representation of them in a certain space which is adequate for clustering data. These representations must contain the semantic description of the images, as it is the discriminating factor we are using to compare images. The solution is a Vision-Language deep learning model that encodes images into vectors, called embeddings, on which it is easier to work on.

We chose to use CLIP (Contrastive Language Image Pretraining) developed by OpenAI as it is very general and does not require a predefined set of categories. This allows for zero-shot image representation and comparison.

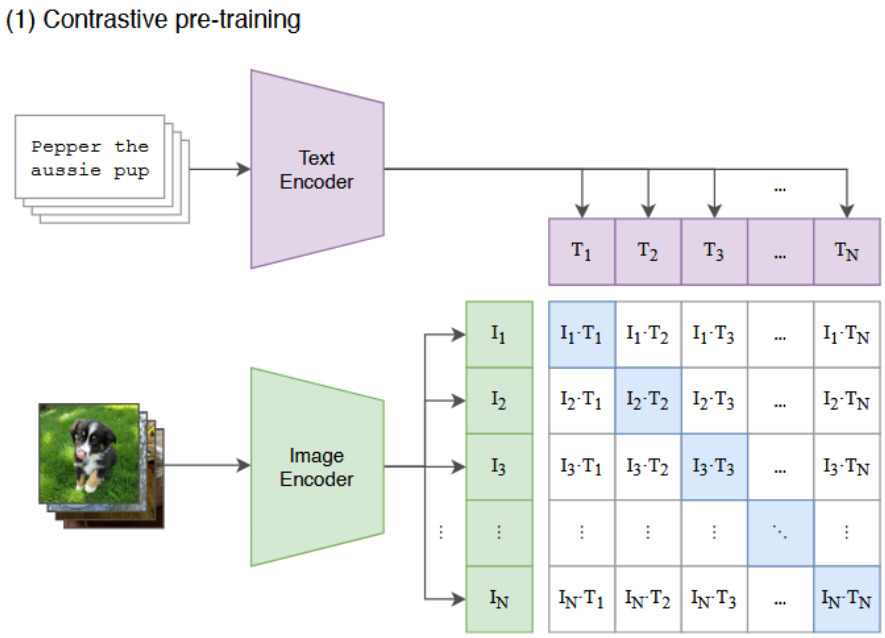


Figure 1: A summary of CLIP approach. Two encoders are used to transform images and text into embeddings. The encoders are trained to match each image to its right corresponding description trough contrastive pre-training. The figure and the concepts are detailed in [7].

CLIP extracts embeddings from the initial images, associating each image with a $\mathbb{R}^n$ vector. It allows us to have a representation of images we can work with while preserving their key semantic characteristics.

2.2 Data Visualization

The extraction of embeddings assigns an $\mathbb{R}^n$ vector to each image. Usually we have $n = 256$ or $n = 512$, and it would be interesting to see the relative positions of these vectors, as the relevant parameter is the distance between the different vectors. The t-distributed stochastic neighbor embedding (t-SNE) algorithm is a dimensionality reduction algorithm that can be used to visualize a dataset: the dimensionality reduction performed by t-SNE tries to preserve the relative distances between the original points. The 2 or 3 dimensional plot is then representative of the global themed organization of the image set. It is a non-invertible transformation of the data only for visualization, as we human can't visualize point-clouds in a space of more than 3 dimensions.

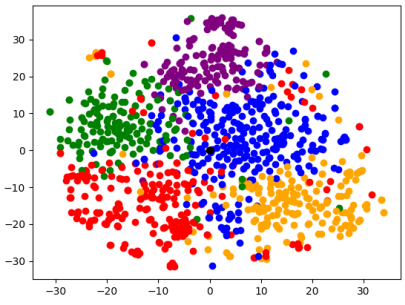


Figure 2: A visualization of a set of IPOL images with t-SNE. The different categories are not clearly separated, and a lot of noise is present.

This algorithm relies on a probabilistic approach ; the high and low dimensional distances between a given point and the other ones are expressed with conditional probabilities, and the Kullback-Leibler divergence between the two probability distributions obtained is minimized. This approach is the same as the SNE algorithm explained in [4], but two main ideas are added : the conditional probabilities are symmetrized, and a Student t-distribution replaces the low dimensional Gaussian distribution used by SNE. These changes provide a much easier cost function to minimize and solve the "crowding problem". It also improves the calculation time and efficiency of the algorithm [6].

More specifically, we define for $i \neq j$:

$$p_{i|j} = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2}{\sum\limits_{k=1,\ k\neq i}^{n} exp(-\|x_i - x_k\|^2/2\sigma_i^2} \tag{1}$$

and $p_{i|i} = 0$. The high dimensional symmetrical joint probabilities are given by:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}. \tag{2}$$

The low dimensional symmetrical probability distribution of t-SNE is set by a Cauchy distribution:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}. \tag{3}$$

The Kullback-Leibleir divergence of these two distributions is:

$$C = \sum_{i,j} p_{ij} log \frac{p_{ij}}{q_{ij}}. \tag{4}$$

whose gradient, calculated in [6], is given by:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}. \tag{5}$$

The cost function C can then be minimized with a gradient descent algorithm as explicited in [6] or [5].

2.3 Clustering

Instead of comparing a given image to every images in the dataset, we want to compare it to the global concepts only. Hence, we want to be able to group images into categories : clustering data consists of partitioning a dataset into interesting subsets. Analysis can then be done on the partitioned data, reducing the calculation time or allowing the identification of a single element with a cluster.

There are many ways in which one can group different elements, and thus many clustering algorithms. State of the art algorithms and their particularities are classified in [9] : the choice of the clustering algorithm must take into account the data set and the problem to solve in order to be relevant.

The construction of the vectors extracted by CLIP is such that the closer they are, the closer is the similarity between the related images [7]. Therefore, a good clustering algorithm for this image recognition problem would be one that groups close images. Thus, K-Means is a relevant algorithm to try.

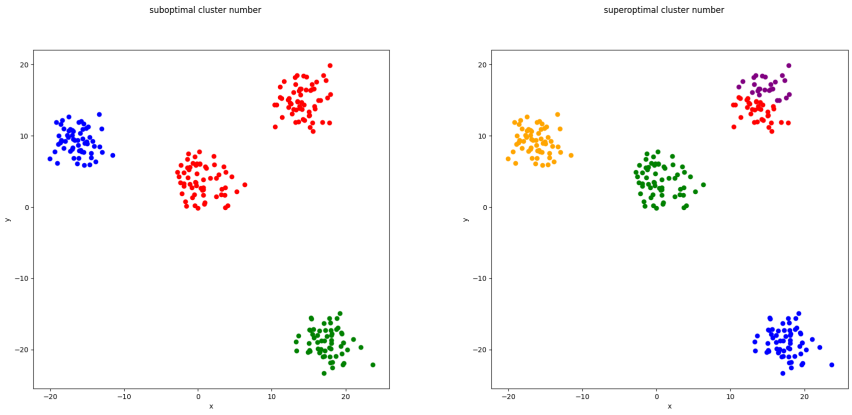
K-Means is a clustering algorithm that minimizes the cluster variance. Each cluster consists of a group of close vectors, and can then be associated with a general image theme. The pseudocode of K-Means algorithm [10] is given in Algorithm 1:

Require: $k \in \mathbb{N}$ and *points* : a set of $\mathbb{R}^d$ vectors
Ensure: a partition of *points* into k parts

```
centroids ⟵ [c_1, c_2, ..., c_k]          ▷ Initialize k centroids from the dataset
▷ Loop until convergence                                                          ◁
while not convergence do
    clusters = [ [ ] for n ∈ {1, ..., k}]                  ▷ Clear previous clusters
    for point in points do          ▷ assign each point to the closest centroid
        cluster_number ⟵ get_closest_centroid_index(centroids, point)
        clusters[cluster_number].append(point)
    new_centroids ⟵ [calculate_centroid(cluster) for cluster in clusters] ▷ Compute the new centroids (usually the mean)
    if new_centroids = centroids then
        return clusters
    else
        centroids ⟵ new_centroids
```

Algorithm 1: Description of K-Means algorithm.

K-Means algorithm requires us to choose k, the number of clusters. The good choice for this integer is given by the elbow method [10]: by plotting inertia (the sum of squared distances to the closest centroid for all observations in the training set) as a function of the number of clusters k, it is possible to find the optimal number. In fact, inertia will decrease sharply with $k \leq k_{opt}$ and decrease very slowly with $k \geq k_{opt}$. Therefore, the graph will form an 'elbow' at k_{opt}. This is explained because when clustering with a too small cluster number, each increment of this number will create a new real cluster, whereas while working above k_{opt}, an increment of the cluster number will only result in the separation of a cluster in half, as illustrated in figure 3.



(a) Sub-obtimal number of clusters (b) Super-optimal

Figure 3: Sub-optimal and over-optimal numbers of clusters for an example dataset. It is clear that going from 3 to 4 clusters decreases a lot inertia, while going from 4 to 5 does not change it a lot.

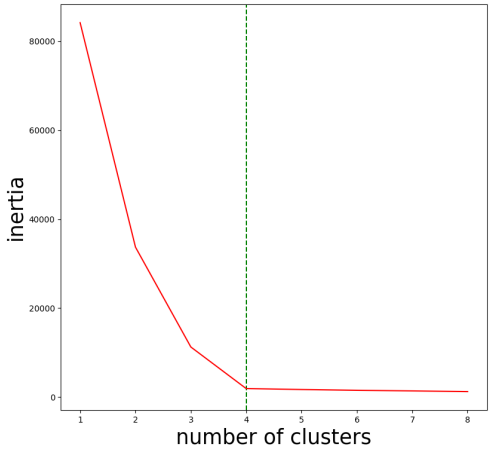


Figure 4: The Elbow Method: it is clear that inertia decreases much less after the fourth cluster. This means that after the fourth cluster, increasing k only results in "cutting a cluster in half"; hence $k_{opt} = 4$.

Here, because of the high number of images we are working with, the data clustering allows us to drastically reduce the calculation time to get similar

images of a given one: once the clustering is done, we are only comparing the input images to each cluster and not to every image in the database.

To test the efficiency of this clustering method, and later to test other concepts, we introduced a controlled dataset of images on which we can actually quantify the accuracy of the clustering and the image-to-cluster comparison. For this purpose, we chose the image sets CIFAR-10 [1] and STL-10 [8], both containing 10 different categories. The interest of these sets is that the categories on which we work on are not overlapping (e.g. a dog and a cat on the same picture). Although it does not perfectly depict reality, working on these controlled datasets will allow us to compare different methods and their efficiency before applying them on a more complex image set, such as IPOL.

image set	number of images	image resolution
CIFAR-10	60,000	32x32
STL-10	13,000	224x224

Table 1: Properties of STL-10 and CIFAR-10 image sets. Having two datasets with different proprieties allow us to have better comparisons.

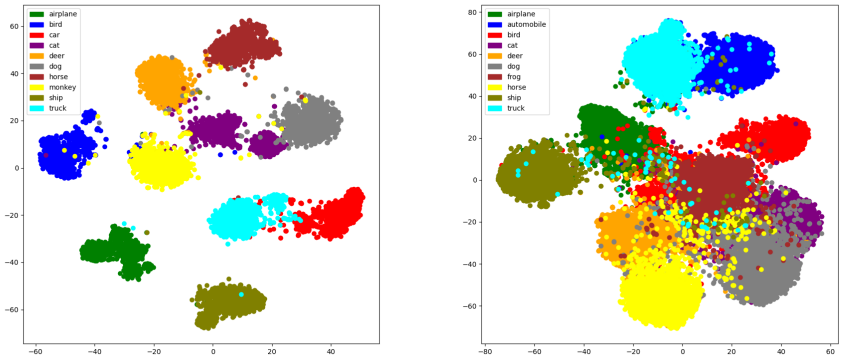
We are in fact clustering about 90% of the images (the training set) for a given dataset, the 10% remaining (the testing set) is used for testing. Concretely, the steps are:

1. Extracting the embeddings from the whole dataset
2. Clustering the extracted vectors from the training set
3. Assigning a category to each cluster
4. Assigning a cluster (thus a category) to each vector of the testing set
5. Computing the confusion matrix and the classification reports

Note that steps 2 and 3 can be repeated, as we do not want two clusters corresponding to the same category : we know that the K-Means algorithm always converges, but its convergence points depends on the initial centroids chosen. Therefore, it is possible to have two clusters with a majority of the same category. Repeating the process until we have all 10 distinct categories allows us to avoid this case.

To offset this same random parameter, steps 1 to 4 are repeated 10 times before computing a confusion matrix which is a quantifying mean detailed after. Hence for each dataset, we tested the equivalent of 10 times the original test set (the same images 10 times, but with different clusters).

The classification reports quantify the accuracy of the tested method by computing and displaying the precision, the recall, the f1-score and the accuracy. Here are the formulas for these values:



(a) STL-10 (b) CIFAR-10

Figure 5: Representation of the image sets with t-SNE. Because of the great number of images on the CIFAR-10 dataset, categories are not as clear as the STL-10 ones.

$$\text{precision} : \frac{\text{True Positive}}{\text{True Positive} \ + \ \text{False Positive}} \tag{6}$$

$$\text{recall} : \frac{\text{True Positive}}{\text{True Positive} \ + \ \text{False Negative}} \tag{7}$$

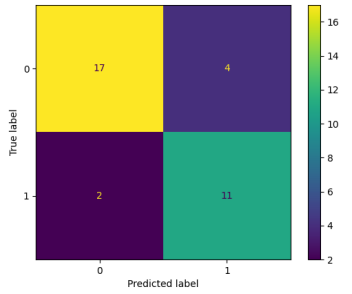
$$\text{f1-score} : \frac{2}{\text{precision}^{-1} + \text{recall}^{-1}} \tag{8}$$

$$\text{accuracy} : \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}} \tag{9}$$

The confusion matrix is a representation of the accuracy of the tested method: it is a double-entry table whose rows represent the real categories and columns the predicted categories. The values of the boxes in the table are the number of images corresponding to the real/predicted couple.

2.4 Dimensionality reduction

The main problem raised by clustering is the calculation time of the algorithm. For instance, the time complexity of K-Means is $O(kndi)$, with k the number of clusters, n the number of data points, i the number of iterations and d the dimension. Therefore, one way to reduce the clustering time is to reduce the number of feature for each embedding: dimensionality reduction is a common method used in data analysis to reduce data from a large dimension space into



(a) Confusion matrix example.

	precision	recall	f1-score	support
0	0.89	0.81	0.85	21
1	0.73	0.85	0.79	13
accuracy			0.82	34
macro avg	0.81	0.83	0.82	34
weighted avg	0.83	0.82	0.83	34

(b) Classification report example.

Figure 6: An example of confusion matrix and its classification report. The confusion matrix shows that most of the predicted label happened to be true as the values in the diagonal are higher.

a smaller one while retaining as much information as possible.

The method used here is the Principal Component Analysis (PCA) reduction, as it is the most usual and common for this problem. We have p-dimensional vectors and the goal is to project them on a q-dimensional subspace with $q < p$.

To keep as much information as possible on the data, we have to find projections that maximize the variance such that distinguishing data and their characteristics is easier. We can also find the subspace such that the distance between the projected vectors and the original ones is the lowest as it may be more relevant intuitively.

Theorem 1. *Minimizing the projection residuals and maximizing the variance are equivalent.*

Proof. Let's first write the data in a matrix $\mathbf{x}$ with each row representing an embedded image and each column a feature of the image. We also make sure that all data are centered, i.e $\frac{1}{n}\sum_{i=1}^{n} x_i = 0$.

Given that the covariance between two points X and Y is $\frac{1}{n}\sum_{i=1}^{n} x_i y_i$, we then have $\frac{1}{n}\mathbf{x^T x} = \mathbf{v}$ where v is the covariance matrix. We are looking for a unitary vector w on which to project our points. The new points will be $y_i = \langle x_i, w\rangle w$.

Minimizing the distance between each points and their projection gives us:

$$\|x_i - \langle x_i, w\rangle w\|^2 = \langle x_i - \langle x_i, w\rangle w, x_i - \langle x_i, w\rangle w\rangle \tag{10}$$
$$= \|x_i\|^2 - 2\langle x_i, \langle x_i, w\rangle w\rangle + \langle x_i, w\rangle^2 \|w\|^2 \tag{11}$$
$$= \|x_i\|^2 - 2\langle x_i, w\rangle^2 + \langle x_i, w\rangle^2 \tag{12}$$
$$= \|x_i\|^2 - \langle x_i, w\rangle^2 \tag{13}$$

with $\|w\| = 1$ as w is unitary. By summing these errors, we then have:

$$\frac{1}{n}\sum_{i=1}^{n} \|x_i\|^2 - \langle w, x_i\rangle^2 = \frac{1}{n}\left(\sum_{i=1}^{n} \|x_i\|^2 - \sum_{i=1}^{n} \langle w, x_i\rangle^2\right) \tag{14}$$

As the first sum does not depend on w, we just have to maximize $\frac{1}{n}\sum_{i=1}^{n}\langle w, x_i\rangle^2$. Defining the variance as:

$$Var[\langle x_i, w\rangle] = \frac{1}{n}\sum_{i=1}^{n}\langle w, x_i\rangle^2 - \left(\frac{1}{n}\sum_{i=1}^{n}\langle x_i, w\rangle\right)^2 \tag{15}$$

We then have:

$$\frac{1}{n}\sum_{i=1}^{n}\langle w, x_i\rangle^2 = Var[\langle x_i, w\rangle] + \left(\frac{1}{n}\sum_{i=1}^{n}\langle x_i, w\rangle\right)^2 \tag{16}$$
$$= Var[\langle x_i, w\rangle] + \left(\langle \frac{1}{n}\sum_{i=1}^{n} x_i, w\rangle\right)^2 \tag{17}$$
$$= Var[\langle x_i, w\rangle] + \left(\langle 0, w\rangle\right)^2 \tag{18}$$
$$= Var[\langle x_i, w\rangle] \tag{19}$$

Minimizing the projection residuals is therefore the same as maximizing the variance. □

Let's focus on maximizing the variance as these two methods are equivalent. In the general case, we are looking for q orthonormal vectors $\{w_1, \ldots, w_q\}$ such that the projection of the points onto the vector space spanned by this base maximizes the variance. These corresponding vectors are in fact eigenvectors of the covariance matrix.

Theorem 2. *The unitary vectors on which to project to maximize the variance are eigenvectors of the covariance matrix* $\mathbf{v}$.

Proof. We have for a single vector w:

$$Var[\langle \mathbf{x}, w \rangle] = \frac{1}{n} \sum_{i=1}^{n} \langle x_i, w \rangle^2 \tag{20}$$

$$= \frac{1}{n} (\mathbf{x}w)^T (\mathbf{x}w) \tag{21}$$

$$= w^T \frac{\mathbf{x}^T \mathbf{x}}{n} w \tag{22}$$

$$= w^T \mathbf{v} w \tag{23}$$

We then have to find w such that it maximizes $w^T \mathbf{v} w$ with the condition $w^T w = 1$. Let's introduce the Lagrange multiplier $L(w, \lambda) := w^T \mathbf{v} w - \lambda(w^T w - 1)$

$$\frac{\partial L}{\partial \lambda} = 0 \Leftrightarrow w^T w - 1 = 0 \tag{24}$$

$$\frac{\partial L}{\partial w} = 0 \Leftrightarrow 2\mathbf{v}w - 2\lambda w = 0 \tag{25}$$

Thus $\mathbf{v}w = \lambda w$ and w is indeed an eigenvector of $\mathbf{v}$. It is easy to see now that the vectors maximizing the variance will be the q first normalized eigenvectors associated with the q highest eigenvalues.

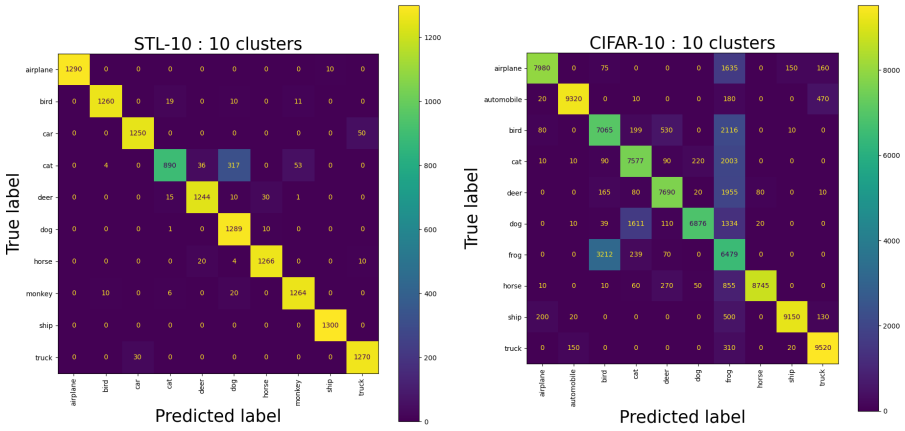
□

While dimensionality reduction significantly improves computation time, it also discards part of the information contained in the CLIP vectors, which may lead to potential inaccuracies in the image retrieval process. We will see how it impacts the performances with the quantified tests.

3 Results and discussion

3.1 Quantifying the initial method

The first thing to test is the quality of the initial method, i.e. the K-Means clustering of the image sets STL-10 and CIFAR-10, as explained in part 2.3; The confusion matrices and the classification reports for these image sets are given in figure 7 and table 4.



(a) STL-10 (b) CIFAR-10

Figure 7: Confusion matrices describing the clustering for the datasets STL-10 and CIFAR-10. Most of images are well predicted as the values in the diagonal are high.

	precision	recall	f1-score	support
airplane	1.00	0.99	1.00	1,300
bird	0.99	0.97	0.98	1,300
car	0.98	0.96	0.97	1,300
cat	0.95	0.65	0.77	1,300
deer	0.96	0.96	0.96	1,300
dog	0.76	0.99	0.86	1,300
horse	0.97	0.97	0.97	1,300
monkey	0.95	0.97	0.96	1,300
ship	0.99	1.00	1.00	1,300
truck	0.95	0.98	0.97	1,300
accuracy			0.94	13,000
macro avg	0.95	0.94	0.94	13,000
weighted avg	0.95	0.94	0.94	13,000

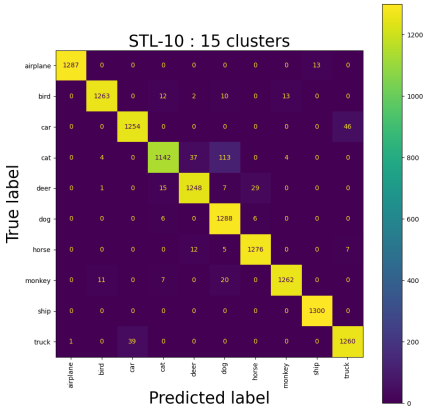
(a) STL-10

	precision	recall	f1-score	support
airplane	0.96	0.80	0.87	10,000
automobile	0.98	0.93	0.96	10,000
bird	0.66	0.71	0.68	10,000
cat	0.78	0.76	0.77	10,000
deer	0.88	0.77	0.82	10,000
dog	0.96	0.69	0.80	10,000
frog	0.37	0.65	0.47	10,000
horse	0.99	0.87	0.93	10,000
ship	0.98	0.92	0.95	10,000
truck	0.93	0.95	0.94	10,000
accuracy			0.80	100,000
macro avg	0.85	0.80	0.82	100,000
weighted avg	0.85	0.80	0.82	100,000

(b) CIFAR-10

Table 2: Classification report of the tests with the K-Means method, K=10; we remark that the results are much better on the STL-10 dataset: the overall accuracy of STL-10 is 0.94, while it is only 0.80 for CIFAR-10

We can already pinpoint that the results are much better on the STL-10 dataset: we observe an accuracy of 0.94 for STL-10 against only 0.80 for the CIFAR-10 dataset. However, the precision of the CIFAR-10 dataset is generally good except for the 'frog' category. We can assume that there is a data particularity on the 'frog' category. In the same way, one very important element is that one fourth of the cat category for the STL-10 dataset is mislabeled as dog, which is translated in the classification report by a low precision number for 'dog' and a low recall number for 'cat'. By examining the figure 5, we realize that the cat category is separated into two groups, one of which is adjacent to the dog cluster. Despite that there are 10 categories, the optimal cluster number is lightly above because of this particularity.



(a) Confusion matrix for the STL-10 dataset, K=15.

	precision	recall	f1-score	support
airplane	1.00	0.99	0.99	1,300
bird	0.99	0.97	0.98	1,300
car	0.97	0.96	0.97	1,300
cat	0.97	0.88	0.92	1,300
deer	0.96	0.96	0.96	1,300
dog	0.89	0.99	0.94	1,300
horse	0.97	0.98	0.98	1,300
monkey	0.99	0.97	0.98	1,300
ship	0.99	1.00	1.00	1,300
truck	0.96	0.97	0.96	1,300
accuracy			0.97	13,000
macro avg	0.97	0.97	0.97	13,000
weighted avg	0.97	0.97	0.97	13,000

(b) Classification report for the STL-10 dataset with the K-Means method, K=15.

Figure 8: After an increase in the number of clusters, results on the STL-10 dataset are highly improved: accuracy went from 0.94 to 0.97, and the cat category f1-score went from 0.77 to 0.92.

Figure 8 gives us the conclusion here, with the examination of the STL-10 dataset, that the optimal cluster number for image recognition is in particular cases a little bit higher than the one given by the elbow method. As we can see in figure 9, the optimal cluster number given by the elbow method is clearly 10, as inertia decreases much less after the tenth cluster. But as we showed before, going a little bit after this number can provide great results.

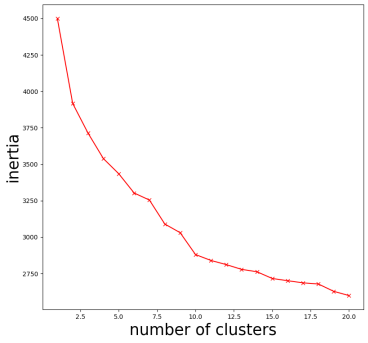


Figure 9: Elbow method on STL-10: here k_{opt} should be 10 as it is where the elbow is located.

	precision	recall	f1-score	support
airplane	0.96	0.80	0.87	10,000
automobile	0.89	0.94	0.91	10,000
bird	0.68	0.70	0.69	10,000
cat	0.78	0.76	0.77	10,000
deer	0.88	0.77	0.82	10,000
dog	0.96	0.69	0.80	10,000
frog	0.38	0.68	0.49	10,000
horse	0.99	0.87	0.93	10,000
ship	0.98	0.92	0.95	10,000
truck	0.93	0.86	0.89	10,000
accuracy			0.80	100,000
macro avg	0.84	0.80	0.81	100,000
weighted avg	0.84	0.80	0.81	100,000

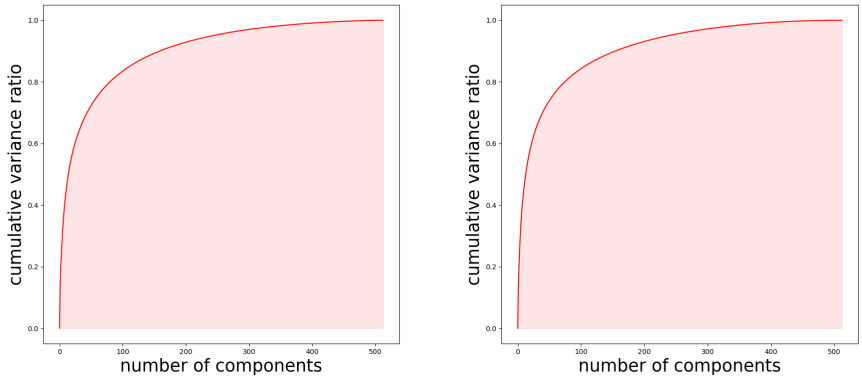
Table 3: Classification report for CIFAR-10 with noise reduction and K-Means with 10 clusters.

Bad results on CIFAR-10 do not come from the fact that it is noisier. Indeed, we can try to remove the noise from the image training set using DBSCAN before clustering it, but the results remain the same as we can see in table 3.

In section 3.2, we will compare the results with these initial ones.

3.2 Going further with PCA

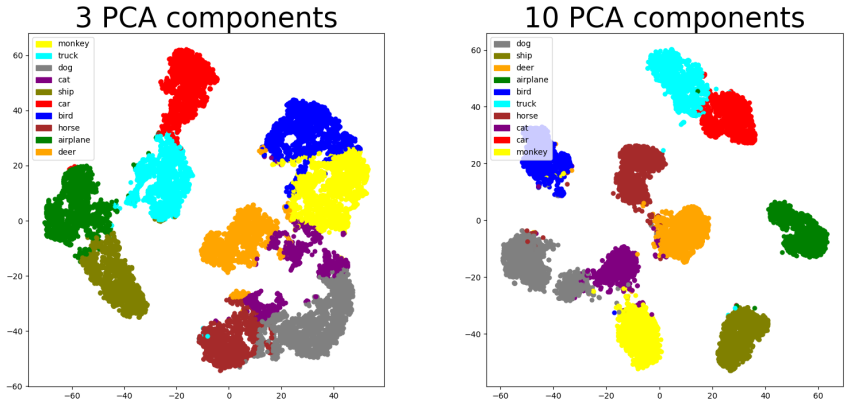
Dimensionality plays an important role when it comes to calculation time for the K-Means algorithm. PCA allows us to reduce this dimensionality, with the cost of information loss. Comparing the efficiency of the already presented image recognition method with the one with PCA will give us a good idea of whether or not the dimensionality reduction is acceptable.



(a) STL-10 (b) CIFAR-10

Figure 10: Evolution of the cumulative variance ratio with the number of principal components. These results are surprising and show that a PCA reduction is indeed promising.

As shown in figure 10, the cumulative variance ratio for both STL-10 and CIFAR-10 image sets is increasing sharply for the first components; this means that a lot of information is contained in these first components. Such data sets is prone to dimensionality reduction, and we can expect good results from these plots.

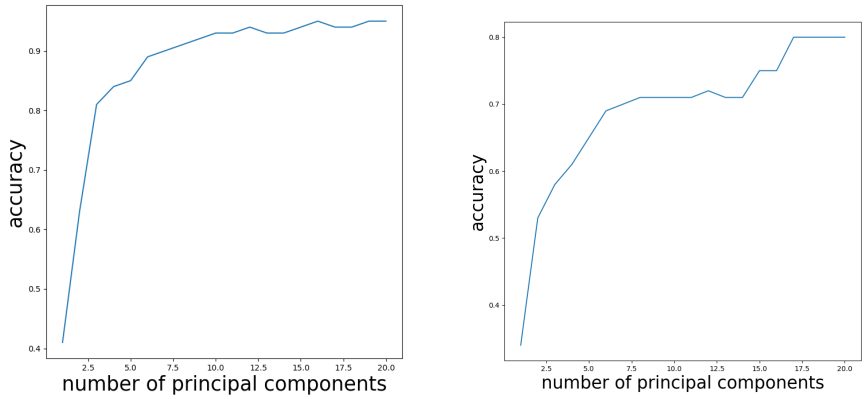


(a) STL-10, 3 principal components. (b) STL-10, 10 principal components.

Figure 11: Visualization of STL-10 with 3 and 10 principal components. Even though the clusters are well defined with only 3 principal components, some categories are closer than they should be (e.g. ship and airplane).

To get a general idea of the accuracy for a given number of principal components, we apply PCA for dimensionality reduction, perform clustering with K-Means, and finally visualize the results using t-SNE. The closer we get from the figure 5, the less information we lose and the better our results.

In fact, most of the performance of our method is achieved using only 10 principal components. We already see from figure 11 that the visualization is really close to the figure 5, but with the plot of the evolution of accuracy with the number of components shown in figure 12 we confirm that accuracy reaches a plateau at 10 components.



(a) STL-10, K=10.

(b) CIFAR-10, K=10.

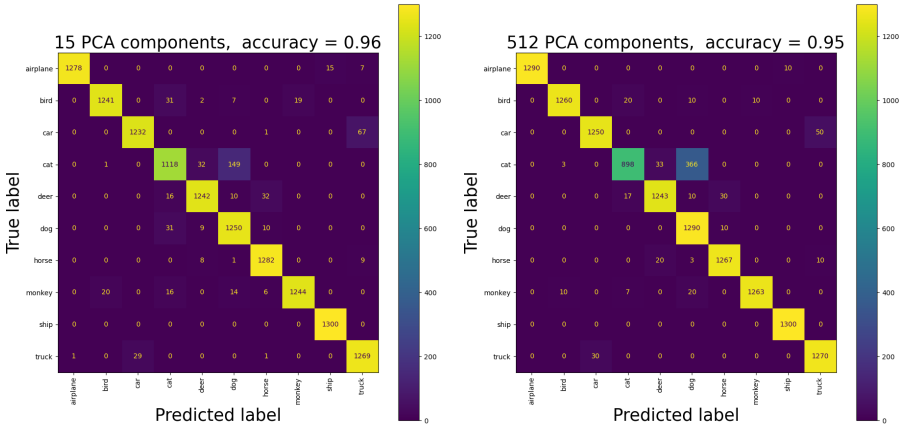
Figure 12: Evolution of the accuracy with the number of principal components kept. The accuracy of STL-10 reaches a plateau very quickly at 10 components, while the CIFAR-10 accuracy reaches it at 20 components (the original accuracy of CIFAR-10 is 0.80).

Because of the dataset STL-10, we explained that forming 15 clusters was much more efficient. For this dataset, we compare the original data – 512 components and 10 clusters – with the reduced dataset where we kept only 15 components and 15 clusters.

These results are very conclusive, since the calculation time of the K-Means algorithm – as diplayed in table 5 – has been divided by 6, and the accuracy – as shown in table 4 – has increased by 0.01 points. Similar results concerning the calculation time are found on the CIFAR-10 image set, but since increasing the number of clusters does not increase the accuracy on this dataset, nothing else can be said about this method here.

The great results found in the dimensionality reduction here come from our requirement for detail in classification; because our categories are wide, the components that classify the fur texture for cats – for example – are irrele-

vant. This enables us to drastically reduce the computation time without losing efficiency.



(a) 15 clusters, 15 components STL-10. (b) 10 clusters, 512 components STL-10.

Figure 13: Confusion matrix for 15 components and 15 clusters, and for 512 components and 10 clusters on STL-10. As explained by table 4, the 'cat' category is better recognized with 15 clusters, even though there are only 15 principal components.

	precision	recall	f1-score	support
airplane	1.00	0.98	0.99	1,300
bird	0.98	0.95	0.97	1,300
car	0.98	0.95	0.96	1,300
cat	0.92	0.86	0.89	1,300
deer	0.96	0.96	0.96	1,300
dog	0.87	0.96	0.92	1,300
horse	0.96	0.99	0.97	1,300
monkey	0.98	0.96	0.97	1,300
ship	0.99	1.00	0.99	1,300
truck	0.94	0.98	0.96	1,300
accuracy			0.96	13,000
macro avg	0.96	0.96	0.96	13,000
weighted avg	0.96	0.96	0.96	13,000

(a) STL-10, 15 principal components, 15 clusters

	precision	recall	f1-score	support
airplane	1.00	0.99	1.00	1,300
bird	0.99	0.97	0.98	1,300
car	0.98	0.96	0.97	1,300
cat	0.95	0.69	0.80	1,300
deer	0.96	0.96	0.96	1,300
dog	0.76	0.99	0.86	1,300
horse	0.97	0.97	0.97	1,300
monkey	0.99	0.97	0.98	1,300
ship	0.99	1.00	1.00	1,300
truck	0.95	0.98	0.97	1,300
accuracy			0.95	13,000
macro avg	0.95	0.95	0.95	13,000
weighted avg	0.95	0.95	0.95	13,000

(b) STL-10, all 512 components, 10 clusters

Table 4: Classification reports of the reduced set and the original one for STL-10. The accuracy of the reduced set with 15 cluster is 0.96, while it is only 0.95 for the original set. Note that the problematic cluster (cat) has excellent scores on the reduced embeddings with more clusters.

3.3 Limitations

The nature of our image retrieval method makes its performances dependent on the dataset : our trials with STL-10 and CIFAR-10 depict this flaw. The

dataset	15 clusters, 15 components STL-10	10 clusters, original STL-10
clustering time (s)	0.023	0.139

Table 5: Clustering time depending on principal components kept. The computational cost is indeed reduced as the clustering time is way higher with the original dataset.

difficulty now lies in the transposition of the results to other image sets. IPOL's image base is composed of big clusters like our data, but is a lot more noisy and features the problem of category covering (pictures with multiples categories, e.g. a picture of a cat and a dog). Hence a denoising of the dataset is to be done before reducing the number of features or clustering.

A more general limitation comes from the orientation of our internship, which focused on semantic image retrieval, starting with the use of CLIP to assign a vector to each image. Similarities can be found in non-semantical content, and especially in low-level features. Indeed, numerous IPOL articles deal with low-level features, such as noise and blue level, or geometrical transformations. But because CLIP classifies the images by their semantic content, a more complete approach would be to combine specific image retrieval methods for low-level features and for geometrical transformation with the method developed in this internship, to cover a wider range of articles and answer more needs.

Figure 14: Two examples of noisy images. A future version of the algorithm will consider noisy images to be alike as they are certainly related to denoising demos.

4 Conclusion

Semantic image retrieval with CLIP and K-Means clustering in order to reduce the comparison time has been conclusive, but the number of clusters has to be chosen with attention to the image set. Plus, some particularities have to be fixed in case of overlapping or non-convex categories (as shown by the example of the less conclusive dataset CIFAR-10). The dimensionality reduction set up by PCA is extremely efficient because of our needs in classification; it allows us to choose the granularity of our retrieval. Because we do not need images that are the exact same, which would limit the number of articles we link to an image, the dimensionality reduction with PCA is a relevant tool to use.

Further work has to focus on adapting the current results on other datasets, in particular the IPOL archive, and dealing with the overlapping of categories. Including other image recognition methods would improve the performances, especially for the handling of non semantic-focused articles. Finally, the general method has to be adapted to take into account new images, corresponding to the new articles published.

Acknowledgement

We would like to thank our supervisors, Miguel Colom and Xavier Bou Hernandez, for their help and patience throughout the internship. Their advices were invaluable, and they were able to convey their passion for the field to us in just a few weeks. We would also like to thank the Mathematics Department at ENS Paris Saclay, whose assistance was also crucial to the smooth running of the internship. Finally, we would like to thank our families for their support, optimism and everything else.

References

[1] A. Krizhevsky. *CIFAR-10 and CIFAR-100 datasets*. https://www.cs.toronto.edu/~kriz/cifar.html.

[2] A. Coates *et al. An Analysis of Single-Layer Networks in Unsupervised Feature Learning*. AISTATS, 2011. https://proceedings.mlr.press/v15/coates11a.html.

[3] T. Hastie *et al. The Elements of Statistical Learning*. Springer, New York, 2009. https://doi.org/10.1007/978-0-387-84858-7.

[4] G. E. Hinton and S. Roweis. *Stochastic Neighbor Embedding*. NeurIPS, 2002. https://papers.nips.cc/paper_files/paper/2002/hash/6150ccc6069bea6b5716254057a194ef-Abstract.html.

[5] S. Jung *et al. A Review of t-SNE*. Image Processing On Line, vol. 14, 2024, pp. 250–270. https://doi.org/10.5201/ipol.2024.528.

[6] L. van der Maaten and G. Hinton. *Visualizing Data using t-SNE*. Journal of Machine Learning Research, vol. 9, 2008, pp. 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html.

[7] A. Radford *et al. Learning Transferable Visual Models From Natural Language Supervision*. arXiv, 2021. https://doi.org/10.48550/ARXIV.2103.00020.

[8] STL-10 dataset. https://cs.stanford.edu/~acoates/stl10/.

[9] H. Yin *et al. A Rapid Review of Clustering Algorithms*. arXiv, 2024. https://doi.org/10.48550/ARXIV.2401.07389.

[10] H. Zhao. *Design and Implementation of an Improved K-Means Clustering Algorithm*, Mobile Inf. Sys., vol. 2022, pp. 1–10, https://doi.org/10.1155/2022/6041484.