March, 2023

IA325

Algorithmic Information & Artificial Intelligence

Micro-study

teaching.dessalles.fr/FCI

Students: David RAKOTO, Quentin TARDIF

Image classification based on the Normalized Compression Distance

1 Abstract

As a method to approximate the Normalized Information Distance (NID), the Normalized Compression Distance (NCD) can be seen as a universal distance, that could be used in classification tasks. Although the NCD is commonly associated with textual data, this project investigates its potential use for the classification of images.

2 Problem

This project aims to address the challenge of accurately classifying images without relying on traditional methods. If the use of compression and NCD presents a potential solution to this problem, there is a need to determine the optimal parameters for its use and to evaluate its performance in different contexts.

3 Method

3.1 Normalized Compression Distance

The NCD, widely used in the algorithmic information theory to approximate the Normalized Information Distances, was expressed by Rudi Cilibrasi and Paul Vitanyi in 2003[1] as :

$$NCD(x,y) = \frac{Z(x,y) - \min Z(x), Z(y)}{\max Z(x), Z(y)}, \tag{1}$$

where Z designates the length of the compressor output.

While this measure can be used for classification tasks, it is important to note that it depends largely on the compression algorithm used and the way in which the joint compression Z(x, y) is calculated.

With this in mind, our proposal is to assess the effectiveness of image classification by varying the parameters used in the calculation of the NCD.

3.2 Compression algorithms

Thus, we evaluated the accuracy of our classification using several compression algorithms: gzip[2], zlib[3], lzma[4], png[5], jpeg[6] and gif[7]. If the first two are quite similar, lzma has a better compression ratio but is also more time-consuming. The first three algorithms are general compression algorithms that can be used on a lot of different types of objects, whereas png, jpeg and gif are better-adapted to images. In particular, it may also be interesting to study the robustness of these compressors.

If our readers are interested in this subject, we strongly invite them to also consult the article *Face recognition through Kolmogorov complexity*[8] by Jordan Van Eetveldt where the author uses video codecs and the CK1 metric[9] to achieve better results and to improve robustness.

3.3 Calculate the joint compression

Concerning the calculation of the joint compression, we have chosen to focus on three methods that we have simply named side-by-side, line-by-line and column-by-column.

As a *naive* way to compute the joint compression, the side-by-side approach simply consists in putting the two images horizontally or vertically one after the other.

On the other hand, line-by-line and column-by-column approaches create a 2NxN image (respectively Nx2N) from our two images of size NxN. More specifically, the first line of the new image will be the first line (resp. column) of the first image, followed by the first line (resp. column) of the second image, then the second line (resp. column) of the first image, and so on, until the last line (resp. column) of each image is reached.

The following image is intended to illustrate the methods previously explained to the reader.

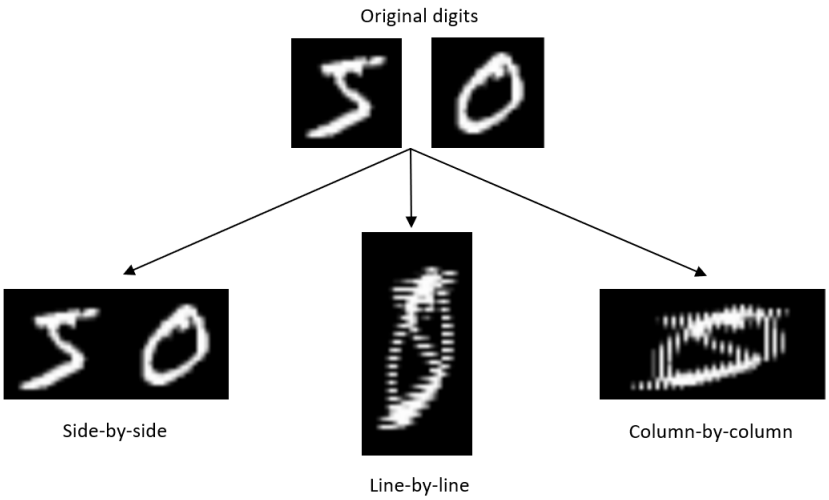


Figure 1 - Illustration of the different approaches for the joint compression

It is to be noted that these two last methods make it possible to preserve a certain form of the digit when this one is compressed with itself or with another representation of itself.

Figure 2 - Column-by-column approach with the same digit

3.4 Evaluate and compare the distances

Before calculating the accuracy of the classification of our images with a k-nearest neighbors classifier[10], we used different other metrics to evaluate the quality of clusters made with our NCDs.

The intra-cluster distance (WSS) and the inter-cluster distance (BSS) are a good starting point that we have chosen to evaluate the aggregation and the separation of our data. To be minimized (respectively maximized), the WSS (resp. BSS) allows to measure the similarity of elements within the same cluster (resp. the dissimilarity between the different clusters).

These distances are calculated using the following formulas:

$$WSS = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, m_i)^2 \tag{2}$$

$$BSS = \sum_{i=1}^{k} n_i . d(m_i, m)^2 \tag{3}$$

where C_i designates a cluster, m_i its centroid, n_i its cardinal and m the data set's centroid.

Often used as a quality measure for the partition of a data set, we also looked at a simplified version of the Calinski-Harabasz index[11] (also known as the Variance Ratio Criterion) as the ratio of the BSS over the WSS. The higher this score is, the better the performance is.

In addition, we have also used the average NCD of our elements depending on the parameters used as a way to predict whether or not the distance will be useful. Indeed, as we are looking for the normality of our compressors ($C(x, x) \approx C(x)$), we want as much as possible to have $NCD(x, x) \approx 0$. If this condition is not met, poor performance is expected.

Finally, we used the distances to perform image classification with a k-nearest neighbors classifier on the MNIST dataset. We then compared the accuracies of the models with the different distances on the test set.

4 Results

All the code associated with this study is available on our Github repository.

4.1 Generic compressors

As explained in the method section, we tried to visualize the effect of the joint compression choice and the compression algorithm on the distances between and within the clusters.

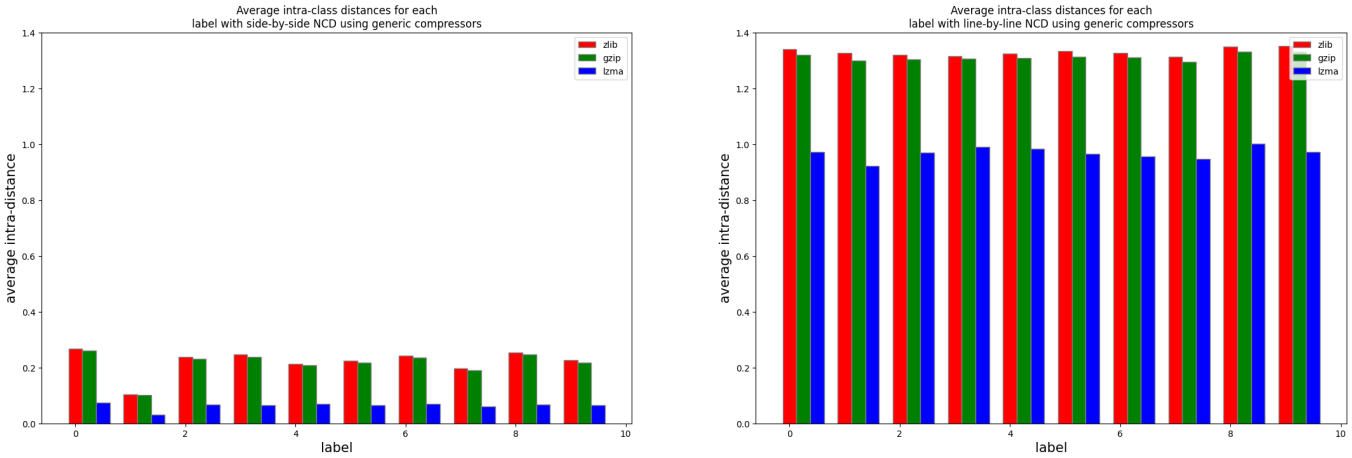


Figure 3 - Intra-cluster distance visualization for generic compressors

We can globally see here that, independently of the joint compression method used, lzma is the compression algorithm producing the smallest intra-cluster distance (*i.e.* the clusters are denser with lzma) while gzip and zlib are rather similar (gzip being slightly better).

We notice however that the line-by-line method tends to increase globally all distances. This implies that the clusters are more scattered with this method.

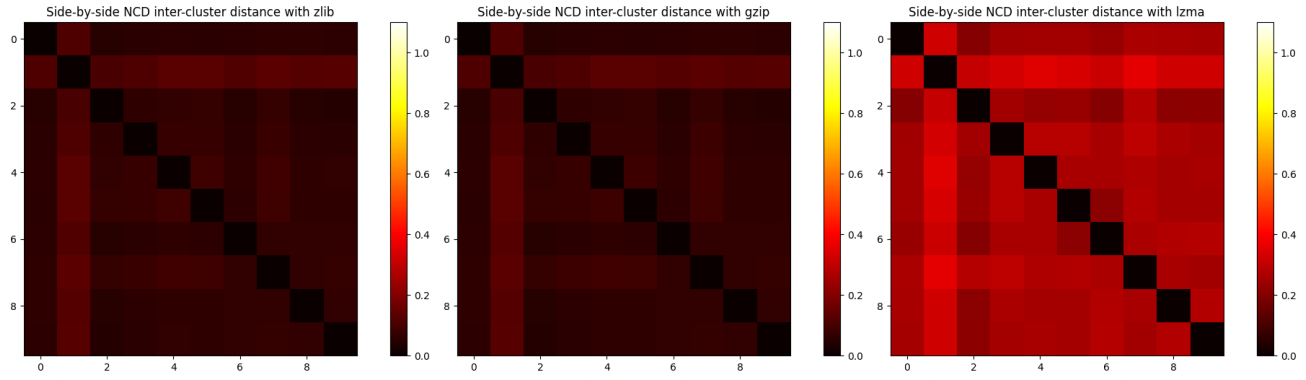


Figure 4 - Inter-cluster distance visualization for generic compressors (side-by-side)

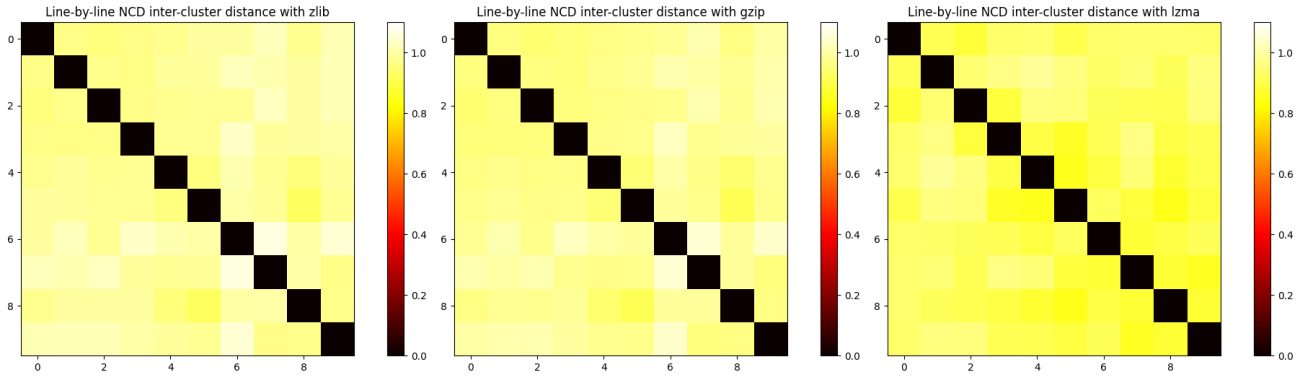


Figure 5 - Inter-cluster distance visualization for generic compressors (line-by-line)

When we look this time at the inter-cluster distance to evaluate if these clusters are well separated, we notice again an advantage for lzma with the side-by-side method since it is globally clearer (implying larger distances).

We can anecdotally note that with the side-by-side concatenation, and for all compression algorithms, the cluster associated to the label '1' is denser, and is well separated from the other clusters. The use of the line-by-line concatenation cancels this effect, and for all compression algorithms, all clusters seem equally dense and equally separated.

For the line-by-line method all distances are increased again. Our clusters are therefore less dense but they are more distinctly separated which tends to counterbalance the intra-cluster distance mentioned above and that is why we need an index that take into account these two distances to better evaluate our models.

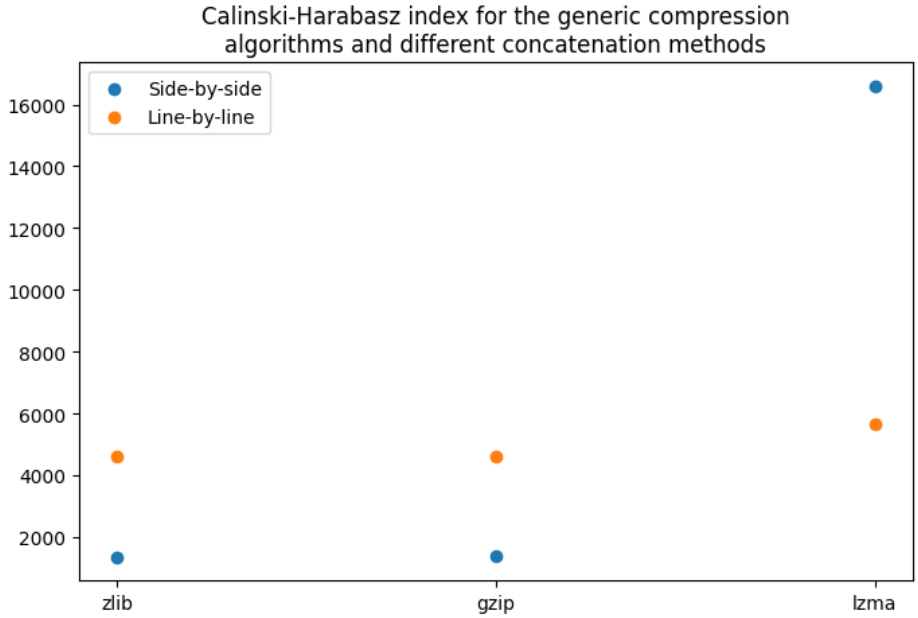
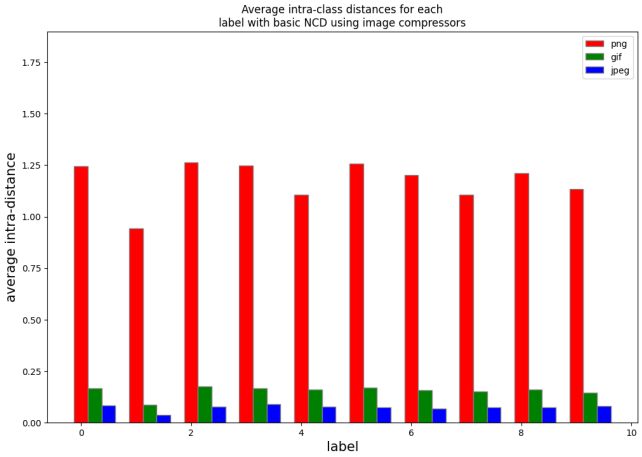


Figure 6 - Visualization of the Calinski-Harabasz index for generic compressors

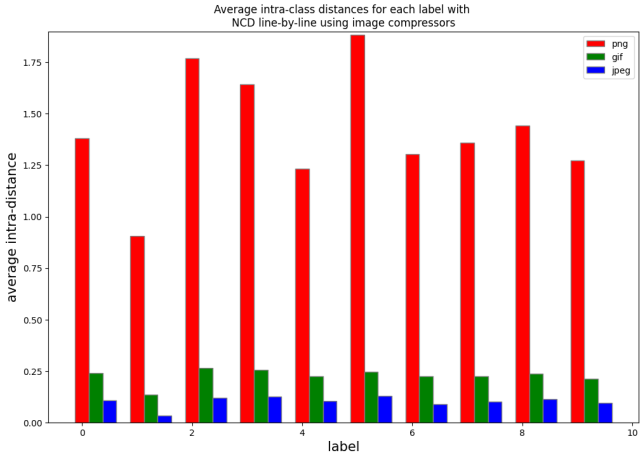
With the Calinski-Harabasz indicator, we can assume that the lzma compression algorithm associated with a side-by-side joint compression computation method should give the best results in terms of accuracy. The use of the line-by-line method tends to increase the predictably poor performance of gzip and zlib but surprisingly lowers those of lzma.

4.2 Image compressors

Here are the values we obtained with the image compressors mentioned above.



Side-by-side



Line-by-line

Figure 7 - Intra-cluster distance visualization for image compressors

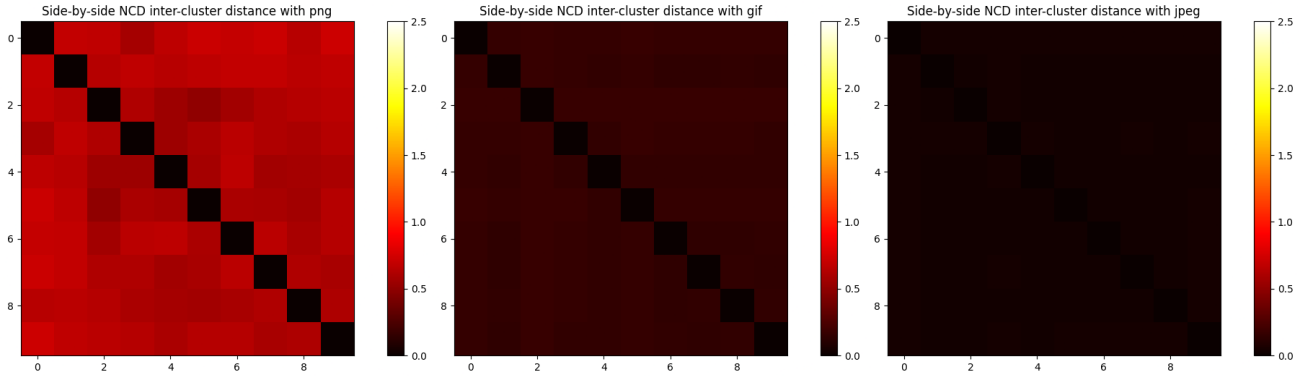


Figure 8 - Inter-cluster distance visualization for image compressors (side-by-side)

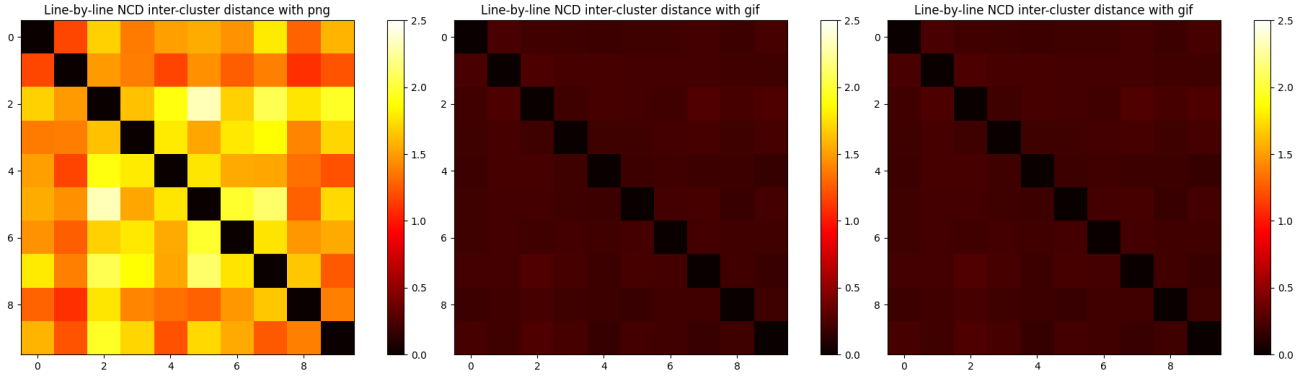


Figure 9 - Inter-cluster distance visualization for image compressors (line-by-line)

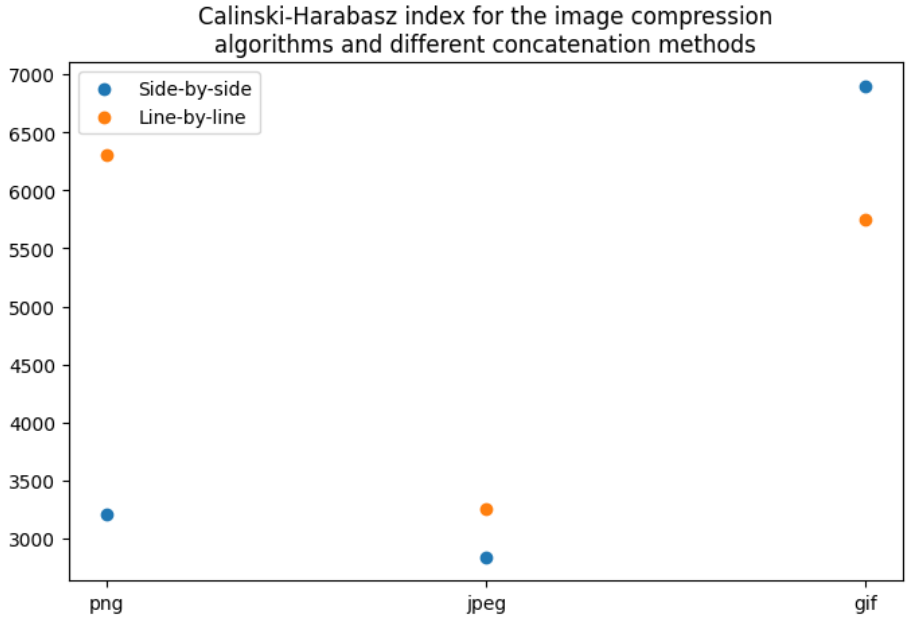


Figure 10 - Visualization of the Calinski-Harabasz index for image compressors

Briefly, we can observe that the PNG compressor creates more scattered but distant clusters, whereas the JPEG compressor creates very dense but also very close clusters compared to the two other methods.

4.3 Distance normality

During our project, we figured that a good way to see if NCD was efficient was to compute the mean NCD for objects with themselves. The results associated with these calculations are summarized in the following table.

Compression algorithm	**Joint compression method**	**NCD(x,x)**
gzip	side by side	0.97
zlib	side by side	0.99
lzma	side by side	0.39
png	side by side	0.09
jpeg	side by side	0.27
gif	side by side	0.12
gzip	line by line	0.08
zlib	line by line	0.09
lzma	line by line	0.04
png	line by line	0.34
jpeg	line by line	0.13
gif	line by line	0.11

4.4 Performance expectation

If the Calinsky-Harabasz index and the distance normality are not perfect indicators for the quality of our distances, it is a good start to have a global vision of our methods' performance.

According to the following graph, using the line by line concatenation results on average in better clustering quality than with the basic side-by-side concatenation. The Calinski-Harabasz index for the side-by-side lzma-based NCD almost seems like an outlier, whereas all other indices look pretty close. We have not been able to explain this difference.

According to the Calinski-Harabasz index, we could expect the best performance using the NCD defined with the line-by-line concatenation with the png compressor, or with the side-by-side compressor with lzma or gif.

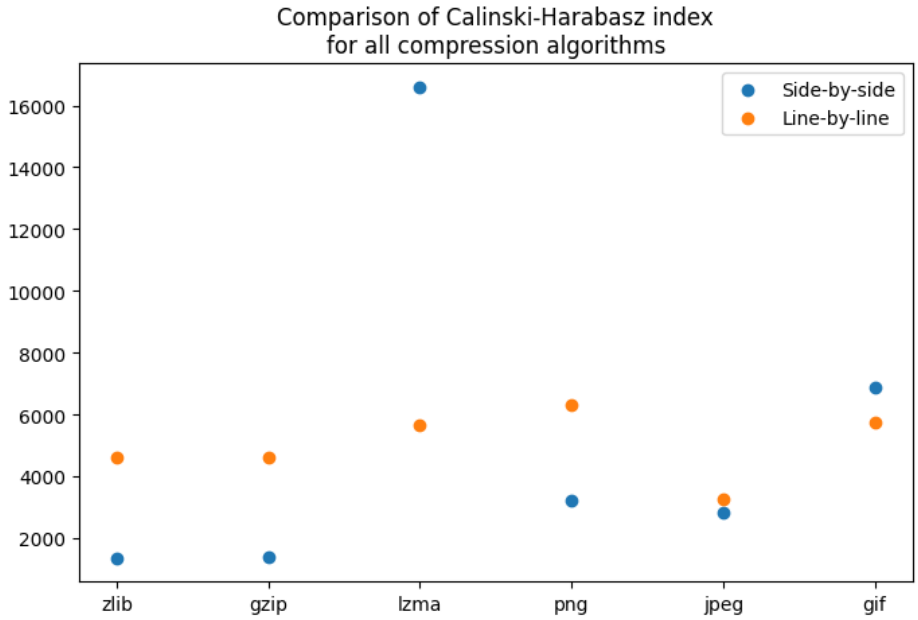


Figure 11 - Visualization of the Calinski-Harabasz index for all compressors

4.5 Classification accuracy

If these indicators help us to estimate the performance of our methods, they do not tell us everything and it is necessary to confront them with the results of accuracy that we can obtain.

Compression algorithm	Joint compression method	Neighbors	Accuracy (in %)
jpeg	side by side	1	92
jpeg	line by line	1	87
gif	side by side	1	85
gif	line by line	1	84
lzma	side by side	1	84
lzma	line by line	1	81
lzma	column by column	1	80
png	side by side	1	76
zlib	line by line	3	52
gzip	line by line	3	47
png	line by line	1	44
zlib	side by side	3	25
gzip	side by side	3	24

If the column-by-column method for the joint compression is mentioned only once in this table, it generally leads to identical results to those obtained with the line-by-line method.

When we compare the results of this table with the different values of the normality of our distances, we realize that it is possible to determine in advance which methods are less appropriate. However, below a certain threshold, it is difficult to say that one distance will be better than another.

The same comment can be made for the Calinsky-Harabasz index. If it can be used in a general context, it works better to compare two methods of joint compression with the same compression algorithm since it is not always a good witness for the global accuracy. In particular, while this index suggests that the JPEG compressor gives the worst performance, it is actually the one that leads to the best classification.

More specifically, we indeed obtained a decent accuracy with lzma on the classification task (81% for the line-by-line and 84% for the side-by-side), as with GIF (84% for the line-by-line and 85% for the side-by-side).

But, much more surprisingly, the performance of the NCD using PNG is quite disappointing, and the distance using JPEG outperforms all the others for the two types of concatenations (87% with the line-by-line and 92% with the side-by-side).

Compared to the different performance obtained with neural networks on this data set, we are rather satisfied with the results obtained with lzma and JPEG.

5 Discussion

The main disadvantage of using the normalized compression distance as a metric for image classification is that it can be computationally expensive and time-consuming. Since the NCD involves compressing three images in total for a single distance, the computation time increase significantly as the size of the data set is important. This can then make it challenging to use the NCD for real-time image classification applications or when dealing with large data sets.

Furthermore, the computation time can also be affected by the choice of compression algorithm and other parameters used in the calculation of the NCD. Therefore, while NCD could be useful tool for image classification, its time-consuming nature is a major challenge that needs to be carefully considered and addressed in order to fully realize its benefits.

In this study, we have chosen to focus mainly on 1-nearest neighbor algorithms. If this method produces limited performance on some data sets, it is useful here as local variations in the data are important on MNIST with these NCD. Our 1-NN algorithms essentially select the single closest point in the training data set as the basis for making predictions, rather than relying on a larger number of neighbors that may not be as relevant. In addition, tests with more neighbors gave us worse performance.

In theory, it would be necessary to use a method such as the elbow method, to find the best number of neighbors to consider for the predictions with the KNN classifier, but the computation time being too important, we could not implement it.

The issue of robustness to noises has not been tackled here but the associated code has been implemented.

It is also worth noting that the images used are small (28x28), so the different compressors have surely been able to deal with almost all the contents of each image. However, as some compression algorithms have a limited scope, it is necessary to continue this study in a more varied framework with more complex data sets or with larger images.

References

[1] Clustering by compression - R. Cilibrasi, P. Vitanyi

[2] gzip - GNU operating system

[3] zlib library

[4] lzma

[5] PNG

[6] JPEG

[7] GIF

[8] Face recognition through Kolmogorov complexity - Jordan Van Eetveldt

[9] A Compression Based Distance Measure for Texture - B. Campana and E. Keogh

[10] K-nearest neighbors classifier, Scikit-learn

[11] The Calinski-Harabasz index