SET REDUNDANCY AND APPLICATIONS TO IMAGE COMPRESSION *

MINH HAI NGUYEN † AND BJÖRK RASMUS‡

Abstract. Technological developments have increased the availability of data. However, the vast scale of data presents issues both in storage and transformation. Previous studies have discussed various methods for compressing and modifying data; although, there is a lack of comparative studies that analyses which methods are most suitable for image data sets. Our aim is to find a compression method superior to PNG for compressing an image data set: specifically the CIFAR10 and MNIST data sets. We present a comparison of the compression rates for the base-line PNG compression with delta encoding, MMD, MMP, and $k-$nearest-neighbors, all combined with PNG compression, with respect to the CIFAR10 and MNIST data sets; additionally, we will combine both MMP and MMD with KMeans clustering. Our results show that by just applying PNG compression, we attained a compression rate of 1.25 for the CIFAR10 data set and 2.67 for the MNIST data set. In contrast, our best compression combination for CIFAR10 - $k-$nearest-neighbors, delta encoding, and PNG attained a compression rate of 1.99; our best compression combination for MNIST - MMP, delta encoding, and PNG attained a compression rate of 3.21. Further research should explore more compression combinations such as additional set redundancy extraction techniques combined with clustering methods not explored in this paper.

Key words. Image compressipon, set redundancy extraction, KMeans clustering, delta encoding, Min-Max Predictive method, Min-Max Differential method, CIFAR10, MNIST, $k-$nearest neighbors

1. Introduction. Hardware and software development in recent years has generated the possibility of gathering extensive amounts of data. For instance, the creation of data volume is projected to triple between 2020 and 2025 [7]. However, using such amounts of data without molding creates problems: Firstly, the internet transfer of massive amounts of data is too slow to be efficiently used. Secondly, the storage of large amounts of data is costly and unproductive. Our project is a collaboration between Institut national des sciences appliquées (INSA) de Toulouse and Groupe Société pour l'informatique industrielle (SII) with the goal of tackling the inefficacy of large data sets; specially by exploring avenues to compress large data sets - data sets involving images - to more efficiently transfer data sets. The goal of this project is to build a compression method that is superior to the well-known PNG method for compression. Thus, if we can combine multiple processes that decreases a data set more than PNG, the project will be deemed a success. The compression methods will be exclusively lossless methods. A lossless compression method is a method that does not lose information during the compression; the opposite compression technique is known as lossy compression - as PNG compression is a lossless technique we do not find it appropriate to compare it to lossy techniques.

The first section of this report presents the data-transformation-process: a filtering method, set redundancy methods, and an image compression method. The second section will detail the experimental results of the three set redundancy methods to determine which method is superior. The third section discusses the results and the implications of said results. For this project, we use the CIFAR10 image data; the data set is provided by CS Toronto [6] and MNIST data set which can be found at [3].

*4A Research Project. Code and supplementary material are available at our GitHub Repository

†INSA de Toulouse (mhnguyen@insa-toulouse.fr).
‡INSA de Toulouse (rbjork@insa-toulouse.fr).

2. Background. To proceed with this paper, we must first describe the theory used for the experiments. The theory is divided into three sections: The first section presents theory about pre-processing the data. The second section presents the set redundancy extraction methods that we use in the project. The third section presents our modifications of the set redundancy extraction methods and a modified approach to compression.

2.1. Pre-processing the data. Before applying the set redundancy extraction methods to the data, we must first pre-process the data so that the SRE methods can more efficiently extract the data. A measurement of how efficiently the SRE methods can extract the data is given by the entropy of the images: the entropy of the images is a measurement of how similar the images are to each other; a low entropy implies that the images are similar and a high entropy implies the opposite. To reduce the entropy of the images we apply the KMeans clustering algorithm to group images that are similar into clusters. After clustering similar images together, the images within each cluster are naturally more similar to each other and thus more suitable for the SRE methods.

2.1.1. Entropy of images. To better understand the information of an image, we can resort to the concept of entropy first introduced by Claude E. Shannon in 1948 [5]. This entropy is defined by the distribution of an image, which is unknown is this case. For simplicity, let us consider a gray image of L levels of intensity $\Omega = \{l_1 < l_2 < ... < l_L\}$ and of size $H \times W$, then the probability that a pixel is at level l_i is the normalized frequency of l_i, denoted by p_i. Then, the Shannon entropy is approximated by

$$H_S(\Omega) = -\sum_{l_i \in \Omega} p_i \log_2(p_i)$$

Since the compressibility of an image depends on the image's entropy, so our objective is, by using SRE, to reduce the entropy of every image in the data set for better compression. A simple way to reduce the entropy of an image is to reduce the range of its pixel values, this is achieved in our work using SRE 2.2 techniques presented and other data transformations presented at 3.1.

2.1.2. KMeans Clustering. Clustering is a method in data analysis which allows the user to cluster - or to sort - data which is similar by a certain characteristic. This is mainly done for two reasons: Firstly, clustering data presents the user with a systematic filing system; by clustering data by a certain characteristic, the user can apply an operation on each group which will have the same, or at least similar, effect on each data point within said group. Secondly, clustering may reveal outliers within the data set. For this project, we apply k-means clustering with the usual Euclidean measure to produce the sets of similar images.

Mathematically, we want to minimize the following objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \tag{2.1}$$

on the space of the centroid c_j where $x_i^{(j)}$ represents the data point i belonging to centroid j. We can not solve this optimisation problem explicitly, but there are iterative algorithms for solving this.

We used the implementation of [4] which is very efficient enables the parallelization.

Algorithm 2.1 KMeans Clustering

1: Randomly divide the data into k clusters
2: **while** not converged **do**
3: (Allocation update). Choose the new allocation as the closest centroid obtained at previous step
4: (Centroid update). Compute the centroid of the new class, defined by the new allocation
5: **end while**

2.2. Classical Set Redundancy Extraction. In [2], Karadimitriou et al. introduced a method for compressing sets of similar images called *Set Redundancy Compression*. The main idea is to use the similarities of the set to extract the information (extra-image information) and modify the images using the similarities before applying an individual compression. In this article, we developed some methods described in [2] for decoding purposes (because we do not want to lose information) and we proposed our new methods for compressing image sets.

The goal of set redundancy extraction is to group the data set into various subcategories with similar characteristics. The sub-categories aid the compression and affects the entropy of the data. We will modify both set redundancy compression methods, this is mainly to aid with the decoding of the compression. This section presents two methods called the Min-Max Differential method and the Min-Max Predictive method.

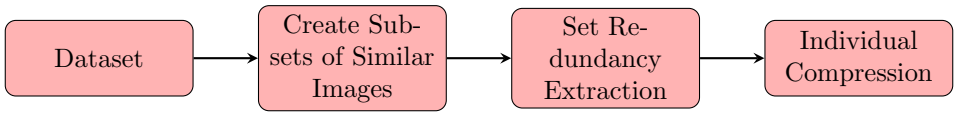

Fig. 1: Classical SRC Scheme adapted to image dataset

In our work, we used KMeans for creating subsets of similar images, MMD and MMP (see Section 2.2.1 and Section 2.2.2) for the SRE step of the scheme and the PNG Compreesion for the last step since we want a lossless compression process. A new scheme is proposed in Section 3.

2.2.1. Min-Max Differential Method. The goal of the Min-Max Differential method (MMD) is to compress the bits needed to present the pixel values in the image. This is accomplished by finding a minimum and maximum pixel value of each pixel within a data set containing images. Each individual pixel can then be represented by its difference to either the minimum or maximum pixel value instead of its actual pixel value. The encoding algorithm can be written as

This algorithm uses half of available statistics for each pixel (min or max) and the previous pixel as the distance prediction. We modify this feature which makes the algorithm different from the original algorithm used in [2] for the decoding purpose. This works for the continuous images but sometimes we see the oscillation effect if the image is not continuous. The decoding phase follows the same process in reversed order.

In terms of computational cost, these encoders and decoders are implemented efficiently by flattening the images onto one-dimensional arrays and `numpy` provides

Algorithm 2.2 Min-Max Differential Encoder

For a set of images, calculate the `min` image and the `max` image
for each image I in the set **do**
  Replace the first pixel by the distance to the `min`: $I_1 = I_1 - \texttt{min}_1$
  for each pixel i in image I **do**
    if $I_{i-1} - \texttt{min}_{i-1} < \texttt{max}_{i-1} - I_{i-1}$ **then**
      Replace the I_i pixel by $I_i - \texttt{min}_i$
    else
      Replace the I_i pixel by $\texttt{max}_i - I_i$
    end if
  end for
end for

Algorithm 2.3 Min-Max Differential Decoder

For a set of images, and the `min` image and the `max` image
for each image I in the set **do**
  Replace the first pixel by the corrected value: $I_1 = I_1 + \texttt{min}_1$
  for each pixel i in image I **do**
    if $I_{i-1} - \texttt{min}_{i-1} < \texttt{max}_{i-1} - I_{i-1}$ **then**
      Replace the I_i pixel by $I_i + \texttt{min}_i$
    else
      Replace the I_i pixel by $\texttt{max}_i - I_i$
    end if
  end for
end for

very efficient functions to complete this process. We implement this process using `numpy` without any explicit loop.

2.2.2. Min-Max Predictive Method. In fact, we can consider that MMD (2.2.1) a predictive method, we predict the distance of the pixel p by the distance of the previous pixel. However, this method does not allow us to use all of available statistical features. The Min-Max Predictive method (MMP) uses both the minimum and maximum image. For each pixel p of image I, we denote

$$L_p = \lfloor m.\frac{I_p - \texttt{min}_p}{\texttt{max}_p - \texttt{min}_p} \rfloor \tag{2.2}$$

the level of that pixel, and where m is the number of possible levels that we can consider as a hyper parameter of our method. Hence, the predicted value is computed as

$$\hat{I_p} = \texttt{min}_p + \frac{L_p}{m}(\texttt{max}_p - \texttt{min}_p) \tag{2.3}$$

then we store the distance $I_p - \hat{I_p}$.

As the pixels in the images are continuous, they do not have drastically different values from their neighbours, this process of modifying the pixel values to its respective level saves storage space [2]. The process can be seen in Algorithm 2.4.

However, as the levels are continuous, they have similar values. Instead of saving the actual level, we will save storage by instead storing the difference between the levels

Algorithm 2.4 Min-Max Predictive Method

1: For a set of images, calculate the `min` image and the `max` image
2: Calculate the value at pixel i
3: **for** each image I in the set **do**
4: Calculate the level image using eq. 2.2.
5: Update the pixel level using eq. 2.3
6: **end for**

(starting from the first level which will remain at its original level). This concept is similar to delta encoding which we present in Section 3.1.

In [1], Ait-Aoudia et al. used the SRE method for medical images (CT images), but in a data set like CIFAR10, the similarity between images are not as good as in CT images. We adapted the SRC method to a general data set, as described in Figure 1, which produced supplementary informations. The evolution of supplementary images produced by the scheme 1 for CIFAR10 data set is given in Figure 2

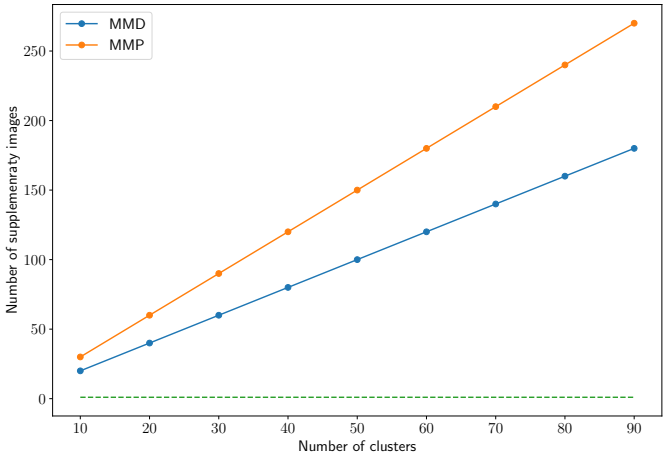

Fig. 2: Supplementary information produced by scheme 1

3. A modified approach. We found that the SRE methods described in Section 2.2 produce supplementary images (e.g. the `min` and the `max` image). This yields two issues. Firstly, the additional images increases the size of the data set. Secondly, the additional images yield greater clustering complexity as the new images must too be grouped within clusters.

In this section, we develop two methods for reducing the information in an image (using the set similarities or not) without producing any supplementary information. Our methods are well adapted for real-world data sets and are implemented very efficiently in Python.

In this scheme, the entropy reduction step is done using SRE or delta encoding (see Section 3.1) or even both with different orders. We showed that this scheme work better than the original scheme 1.

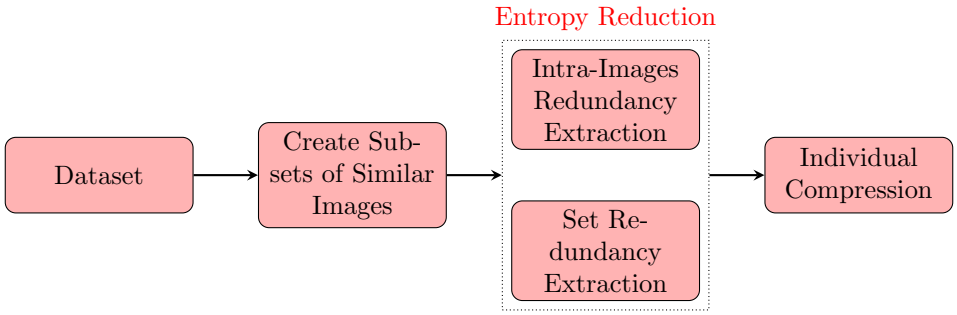

Fig. 3: A New Compression Scheme

3.1. Delta encoding. Delta encoding is a well-known and widely used concept in mathematics. Delta encoding compresses information by only storing the difference between two data points. This is done by using the first data point as a mother node: it is stored as its real value. The subsequent data point is stored as the difference between its real value and the mother node; the following node is thus stored as the difference between its real value and the second real value.

We extend this ideal into 2D which is adapted to images; the delta method in 2D is largely inspired by the concept of the *gradient* (see Wikipedia). In fact, the gradient shows how much the pixel values change instead of the pixel values itself. Since the natural images are usually continuous, it is obvious that the gradient fluctuates less than the pixel values.

We adapt the concept of the gradient to be able to do the decoding phase and our Delta Encoder is in fact the horizontal gradient calculated by channel while keeping the first line of image unchanged. An illustration of this delta encoding using the famous Lenna image is given in Figure 4.

(a) Lenna (b) Delta Encoding

Fig. 4: Illustration of Delta Encoding applying to Lenna

In the CIFAR10 data set, delta encoding works very well, as illustrated in Figure 5, we reduced significantly the information in the images.

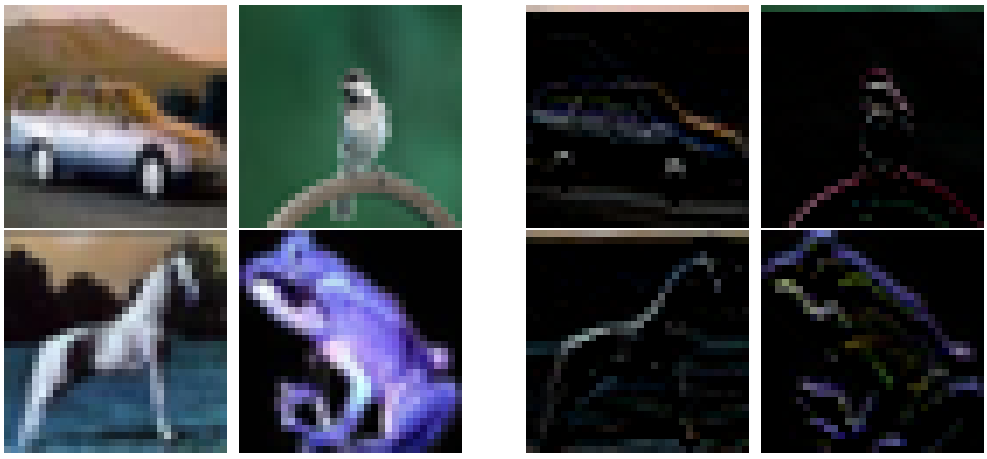

Fig. 5: Illustration of Delta Encoding applying to CIFAR10

3.2. $K-$**nearest-neighbors.** The original $k-$nearest-neighbors algorithm is used for a supervised classification problem: the label of a new data point is predicted using a majority vote of its k-nearest neighbors (the neighbors are determined by a pre-defined distance - for this project we use the Euclidean distance). From the experiments of compression models using SRE (presented in Section 2.2) with clusters from KMeans, we observed that the quality of the images in a cluster was not good enough in real-world data set such as CIFAR10, i.e the images are not similar enough since the data set was very large (60000 images). Moreover, we propose a method that applies the set to extract the extra-image similarities without producing any other images (which is a disadvantage of SRC). An illustration of how similar the neighbor of an image is to the original image is given in Figure 6a and Figure 6b.

We adapt the $k-$nearest-neighbors algorithm to our compression problem as described in Algorithm 3.1. An example of how Algorithm 3.1 works can be found in Figure 6c: we dramatically reduce the information in the encoding image and therefore reduce its size.

Algorithm 3.1 $K-$nearest-neighbors adapted to compression problem

1: **for** each image I in the set **do**
2: Calculate the nearest image in the set using $k-$nearest neighbors algorithms (here $k = 1$), denoted by I_{nb}
3: Replace I by $I - I_{\text{nb}}$ and save the neighbor index for decoding phase.
4: **end for**

4. Experimental results. To assess the compression rates we apply each compression method to both the CIFAR10 and MNIST data sets. Below is the result of base-line PNG usage; the base-line PNG result serves as the basis of comparison for our compression techniques. Following the PNG results will be the results of applying the 4 processes with PNG: (1) the delta method, (2) the delta method and MMD, (3) the delta method and MMP, and (4) $k-$nearest-neighbors and delta encoding.

(a) A random image in CIFAR (b) Its nearest neighbor in CIFAR (c) Its encoding by differentiating

Fig. 6: Illustration of $k-$nearest neighbors

4.1. PNG. To compare our compression methods we must first determine the baseline, i.e., what results our techniques are to be measured against. For this, we apply PNG to the CIFAR10 and MNIST data sets. Applying PNG to the CIFAR10 data set yields a compression rate of 1.25 and applying PNG to the MNIST data set yields a compression rate of 2.67.

4.2. Delta encoding. To see the effects of the delta encoding method we experimented with the combination of the delta method and the MMP with the CIFAR10 data set. The result is shown in Figure 7. The images are sorted in size to create a better plot; this leads to the problem that image 1 might not be the same image in both MMP and MMP with delta as either method may have been superior in compressing a certain image. However, the plot is used to show the average improvement of combining MMP with delta and should thus not be taken literally: the plot is a proof of concept rather than an analytical comparison of the methods.

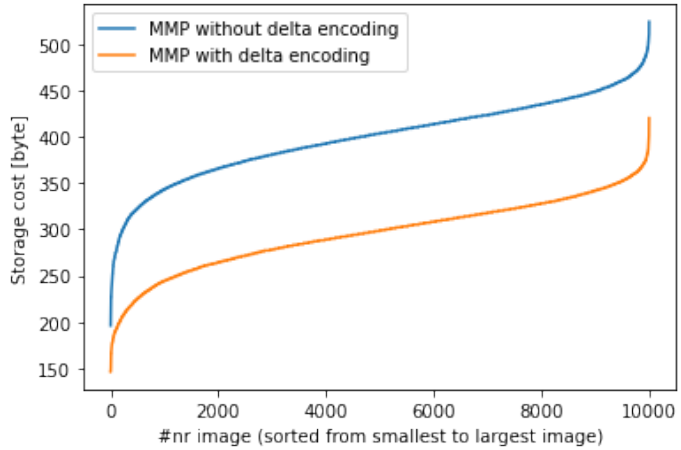

Fig. 7: Comparison of the storage cost for MMP with and without delta encoding

On average, the delta-modification to the MMP decreases the storage cost of each image by 26%: the delta method works very well with both MMP and PNG.

Applying delta encoding before the PNG method yields a compression rate of 1.96 for the CIFAR10 data set; delta encoding and PNG yields a 2.46 compression rate on the MNIST data set.

4.3. Delta encoding and MMD. In Figure 8 we note the results from applying MMD and delta encoding to the CIFAR10 data set.

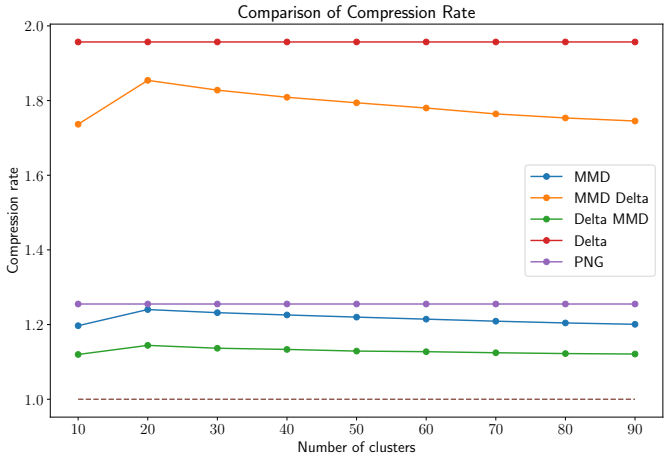

Fig. 8: Experimental results Min-Max Differential and Delta applying to CIFAR10

Solely using the delta encoding method is the superior choice; the addition of the min- and max-images does not justify the total compression rate. However, for the cases where we combine the MMD and delta encoding, we can tell that it is better to apply MMD before delta encoding. It is obvious why the compression works better when applying the delta method after applying MMD: MMD extracts data which is similar, i.e, images whose pixels are similar. As such, the delta method will find images with more continuous values (the gradient fluctuates less than the pixels) than if the delta method was applied before the MMD and delta encoding can therefore better compress the data. The peak compression is reached at 20 clusters which yields a compression rate of 1.85.

Figure 9 presents the the different MMD/delta combinations on the MNIST data set.

By combining MMD and delta encoding we cannot achieve greater compression than solely using PNG. We can note that the compression rates differ between the two data sets. For the CIFAR10 data set, delta encoding performs best and for the MNIST data set, excluding PNG compression, MMD and delta encoding performs best. We also note that applying MMD before delta

4.4. Delta encoding and MMP. Figure 10 displays the results of applying MMP and delta encoding.

Similar to the MMD method, it performs worse than delta encoding. Additionally, it also perform better when applying the set redundancy extraction method before the delta encoding. The peak compression rate for the MMP/delta combination is reached at 20 clusters and compresses the data set by 1.51. Interestingly, MMP without delta encoding increases the size of the data set; most likely due to bad compression as well

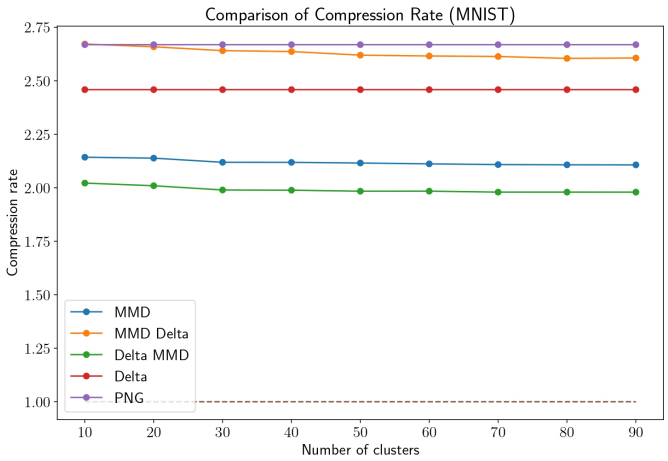

Fig. 9: Experimental results Min-Max Predictive and Delta applied to CIFAR10

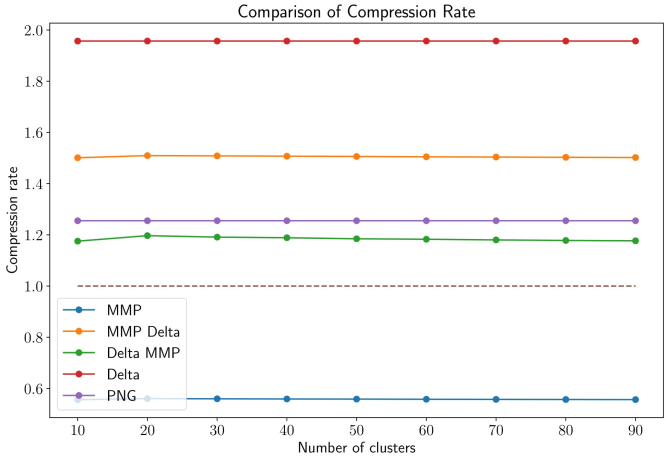

Fig. 10: Experimental results Min-Max Predictive and Delta applied to MNIST

as increasing the data set by adding min-, and max-images. By recalling Figure 2 we realized that the MMP method adopts more supplementary images than the MMD method; this could explain why the MMP actually increases the size of the data set after compression while the MMD does not.

Figure 11 presents MMP applied to the MNIST data set.
MMP combined with delta encoding performs very well. It is significantly greater in its compression rate than PNG. The peak compression rate, 3.21, is reached by using 30 clusters. MMP does not increase the size of the data set, as it did with respect to the CIFAR10 data set, when applied to the MNIST data set.

4.5. $K-$**nearest-neighbors.** The results from $k-$nearest-neighbors can be viewed in Figure 12.

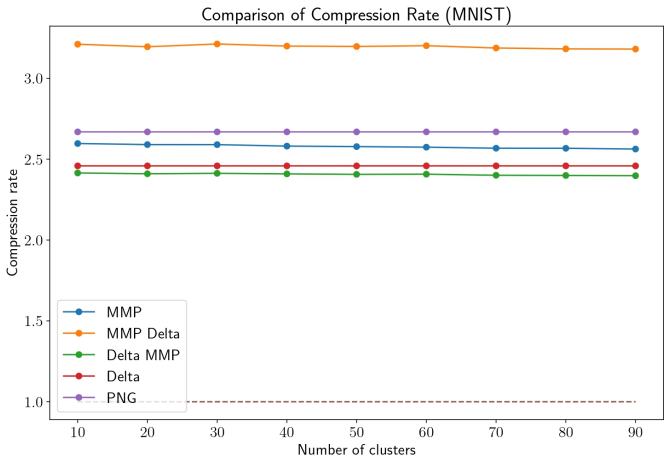

Fig. 11: Experimental results of MMP and delta encoding applied to MNIST

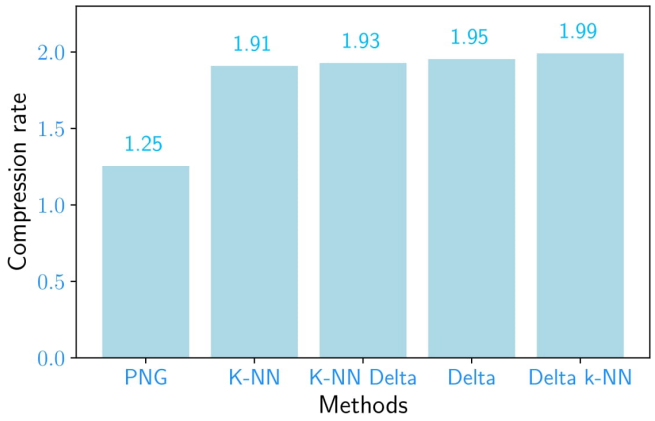

Fig. 12: Experimental results of combining $k-$nearest-neighbors with delta encoding applied to CIFAR10

All combinations of applying $k-$nearest-neighbors and delta encoding to the data set perform well. We used the nearest neighbor to reduce the information in every image of the data set.

Figure 13 presents k-nearest-neighbors applied to the MNIST data set.
We note that, as well as applied to the CIFAR10 data set, the combination of delta and k-nearest-neighbors compresses the MNIST data set best. The rest of the combinations does not perform better than PNG. We proposed different methods for entropy reduction step in the scheme Figure 3, and show here in the Figure 14 an example of how these methods modify an image, using the MNIST data set.

5. Conclusions. A summary of the peak compression rate applied to the CIFAR10 and MNIST data sets are presented in Table 1 and 2 respectively.

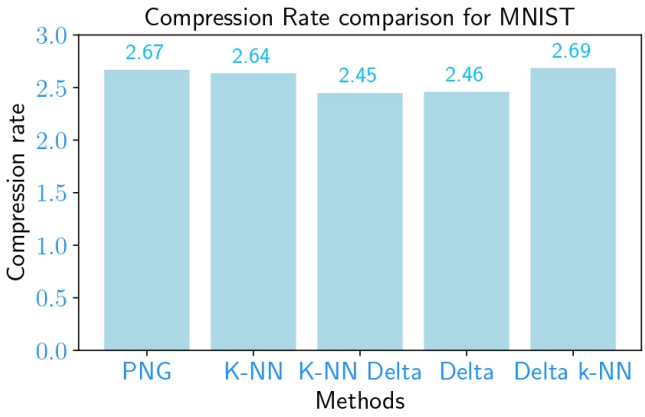

Fig. 13: Experimental results of combining $k-$nearest-neighbors with delta encoding applied to CIFAR10

(a) Original (b) Delta (c) MMD (d) MMP (e) $k-$nn

(f) Delta $k-$nn (g) $k-$nn Delta (h) Delta MMD (i) MMD Delta (j) Delta MMP (k) MMP Delta

Fig. 14: Illustration of how an image within MNIST is modified by different methods

CIFAR10 - Compression technique	Compression rate
$K-$nearest-neighbors + Delta + PNG	1.99
Delta + PNG	1.96
$K-$nearest-neighbors + PNG	1.91
MMD + Delta + PNG	1.85
MMP + Delta + PNG	1.51
PNG	1.25

Table 1: CIFAR10 compression results

MNIST - Compression technique	Compression rate
MMP + Delta + PNG	3.21
MMD + Delta + PNG	2.67
PNG	2.67
$K-$nearest-neighbor + PNG	2.64
Delta + PNG	2.46
$K-$nearest-neighbor + Delta + PNG	2.45

Table 2: MNIST compression results

All the peak compression rates of our methods perform better than PNG applied to the CIFAR10 data set. Applied to the MNIST data set the performance is not as good. Only MMP, delta encoding, and PNG performs better than PNG; MMD, delta encoding, and PNG performs equally to solely using PNG. Thus, we have, for both data sets, discovered compression techniques superior to the PNG method. It is thus clear that the aim of this paper has been achieved. A key result from this paper is that combining multiple compression techniques is a valid approach for compressing data sets. Our best performing method for compressing the CIFAR10 data set combines three techniques: PNG, delta encoding, and the $k-$nearest-neighbor method. It would be interesting to further explore additional combinations of compression techniques to reach even better compression rates.

Acknowledgments. We would like to thank Kaveena Persand for continuous support throughout the project. Her guidance has been crucial in both planning and executing the project.

REFERENCES

[1] S. Ait-Aoudia and A. Gabis, *A comparison of set redundancy compression techniques*, EURASIP Journal on Advances in Signal Processing, (2006), https://doi.org/10.1155/ASP/2006/92734.

[2] K. Karadimitriou and J. M. Tyler, *Set redundancy, the enhanced model, and methods for compressing sets of similar images*, PhD Dissertation at Louisiana State University, 26 (1997), pp. 47–52.

[3] Y. LeCun, *The mnist database of handwritten digits*, http://yann.lecun.com/exdb/mnist/ (accessed 2022-05-10).

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cour-

napeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[5] C. E. Shannon, *A mathematical theory of communication*, The Bell System Technical Journal, 27 (1948), pp. 379–423, https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

[6] C. Toronto, *The cifar-10 dataset*, https://www.cs.toronto.edu/~kriz/cifar.html (accessed 2022-05-10).

[7] A. von See, *Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025*, 2021, https://www.statista.com/statistics/871513/worldwide-data-created/ (accessed 2022-02-10).