MATH 104 - Final Project

Nicolas Fertout

December 2022

Abstract

This project involved the implementation and analysis of the Haar wavelet image compression technique using different sizes and parameters on images of different sizes and sources. I found that the parameters that allowed the best size-reduction while preserving accuracy were the 2×2 Haar matrix, along with an $\varepsilon = 15$. For larger values of ε, the choice of a 4×4 Haar matrix seems to be preferable.

1 Introduction

1.1 Approach

In this report, I will discuss and analyse different usages of the Haar wavelet transform with regard to image compression. The Haar wavelet transform is an efficient image compression technique that uses solely tools from linear algebra. The approach consists in splitting the image (represented as a $m \times n$ matrix) into square matrices of size 2^k, with k being a hyperparameter to be adjusted by the user. Then, each of those square blocks (M, say) get multiplied by the Haar wavelet transformation matrix H on the left and right (to get N, say). This transformation matrix has the interesting property that it leads to sparse matrices, with only a few non-zero entries containing most of the information. Then, one can chose another hyperparameter ε, such that any value below this threshold in N get set to 0 (we obtain $\tilde{N}$, say). We can then easily recover an approximation of the original matrix M by multiplying by the inverse of H. The resulting matrices ($\tilde{M}$, say) make up the compressed image.

1.2 Mathematical Background

The different steps of this algorithm can be summarized as follow:

Data: List of k initial $d \times d$ matrices M_i.
Result: List of k compressed $d \times d$ matrices $\tilde{M}_i$
$i \leftarrow 1$
while $i \leq k$ **do**
  $N_i = H^{-1} M_i H = H^T M_i H$;
  Set $\tilde{N}_i$ s.t. if $(N_i)_{(j,k)} < \varepsilon$, then $(\tilde{N}_i)_{(j,k)} = 0$
  $\tilde{M}_i = H \tilde{N}_i H^{-1}$
  $i \leftarrow i + 1$
end

Algorithm 1: Haar Wavelet Transformation

We will use the 2×2 and 4×4 versions of the Haar wavelet transform, with respective matrices:

$$H_{2\times 2} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$$

$$H_{4\times 4} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{2} & \frac{-1}{2} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & \frac{-1}{2} & 0 & \frac{-1}{\sqrt{2}} \end{pmatrix}$$

Note that, while the Haar wavelet compression can perform lossy compression by varying the ε parameter, it can also perform lossless compression. Indeed, when ε is set to $-\infty$:

$$\tilde{M} = H\tilde{N}H^{-1} = HNH^{-1} = HH^{-1}MHH^{-1} = M$$

2 Datasets

2.1 Data Description

For this project, I decided to use my own dataset: images of different sizes and dimensions from my collection of photos. As opposed to most image dataset found online, this one will not have any pattern, or recurrent objects, shapes or colors. Photos comes from a variety of sources, taken by different cameras, thus leading to a large spectrum of sizes, going from 242 KB to 6.06 MB.

2.2 Data Cleaning

For this project, all photos will be taken in gray scale. Besides, one can notice that in order to apply a $H_{d\times d}$ transform, the dimensions of the input photo need to be multiples of d. I thus adjusted the size of all photos whose height or width is not a multiple of 4 (see helper function `adjust_dimensions`).

3 Results

Images of various sizes and format were compressed using several ε values and Haar matrix size.

Here is a comparison between an original (M), transformed ($\tilde{N}$) and compressed ($\tilde{M}$) photo when using a 4×4 basis, along with corresponding sizes on disk:

Figure 1: My nephew driving a BMW: original gray-scale photo (M). Size = 847 KB

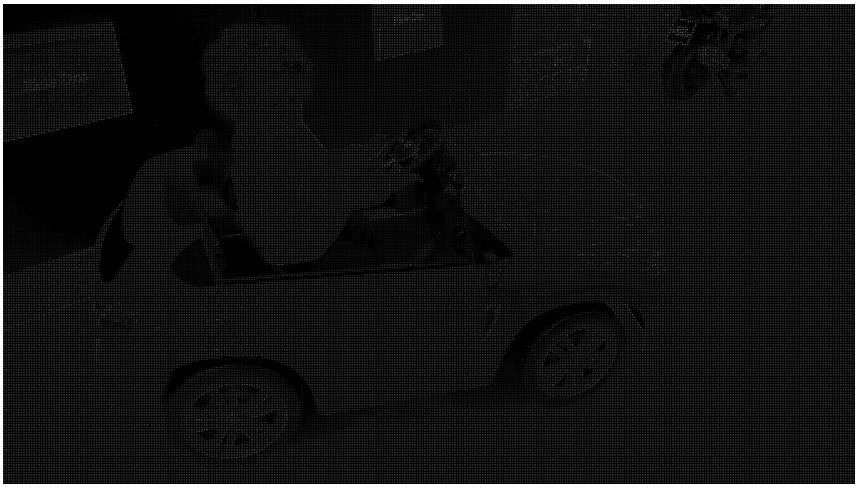

Figure 2: Transformed Photo ($\tilde{N}$). Size = 78.1 KB

Figure 3: Compressed Photo ($\tilde{M}$). Size = 159 KB

We can see in Figure 2 that only the main features were preserved by the transformation, leading to a substantial decrease in size on disk by a factor of 10.8. The recovered picture retains most of the information, while have a decrease in size by a factor of 5.3.

Now, an important tool to analyze the performance of a compression algorithm is the compression ratio r of the number of nonzero entries in N to the number of nonzero entries in $\tilde{N}$. It is directly proportional to the change in size of the image.

Besides, one need to keep track on the loss induced by the compression. This can be done using many different loss function. Here the Root-Mean-Square Error (RMSE) was used. Let m and n be the height and width of the image respectively, and let $I(i,j)$ and $\tilde{I}(i,j)$ be the value of the pixel in position (i,j) of the original image and compressed image respectively.

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{m*n}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j) - \tilde{I}(i,j)]^2}$$

By running compression on the whole dataset, we get the following average values:

Size of Haar wavelet matrix	$\varepsilon = -\infty$	$\varepsilon = 0$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 30$	$\varepsilon = 100$	$\varepsilon = 250$
$H_{2\times2}$	1.00	1.82	2.13	2.49	2.67	2.80	3.09
$H_{4\times4}$	1.00	1.98	2.34	2.74	2.94	2.98	3.01

Table 1: Average Compression Ratio r of the Haar wavelet compression technique for different matrix sizes and ε values.

Size of Haar wavelet matrix	$\varepsilon = -\infty$	$\varepsilon = 0$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 30$	$\varepsilon = 100$	$\varepsilon = 250$
$H_{2\times2}$	15.91	20.22	20.10	20.23	21.55	26.09	53.64
$H_{4\times4}$	2.33	22.13	22.18	22.70	24.13	26.07	32.60

Table 2: Average Root-Mean-Square Error (RMSE) of the Haar wavelet compression technique for different matrix sizes and ε values.

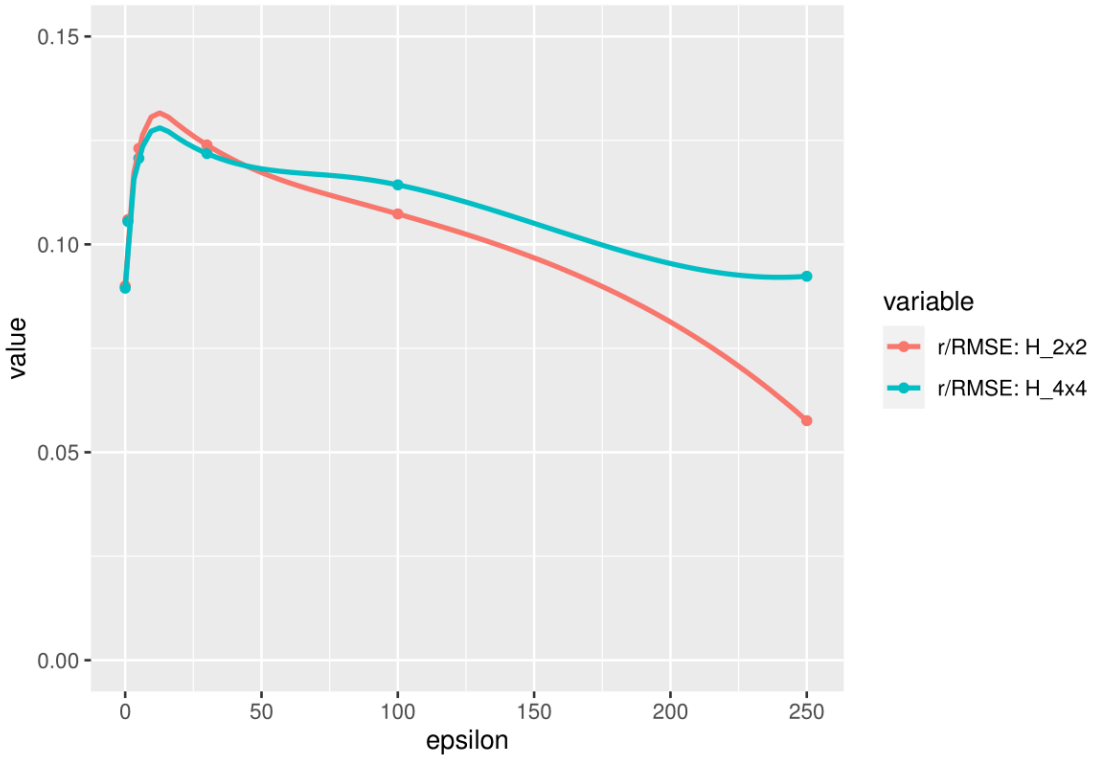

Figure 4: Ratio of r to RMSE for both matrix sizes (curves obtained using spline interpolation).

4 Discussions and Conclusion

Overall, while increasing the value of epsilon increases the average compression ratio, it also reduces a lot the quality. Finding a balance depends on the input and the purpose of the compression task.

However, an important observation is the sharp log-shape of r as ε increases. It is therefore preferable to take smaller values of ε, as the majority of the compression seems to be done. This can further be verified when analyzing the value of RMSE as ε increases. Note that the positive values when taking $\varepsilon = -\infty$ are mainly due to rounding errors (both from all the mathematical operations, but also from the conversion of images by the python library).

Therefore, by plotting the ratio $\frac{r}{RMSE}$, we can get an indicator of the size-reduction to loss ratio. We notice that the best choice of ε seems to be at around 15. Surprisingly, the best size-reduction to loss ratio is achieved at that point using the 2×2 Haar wavelet matrix (even though the 4×4 basis seems to be better for larger values).

Next Steps

As next steps, here are some paths to investigate or analyze further:

- Try other loss functions, such as Mean-Absolute Error, which can be interpretable as the average value by which a pixel is off.
- Identify which KPI (Key Performance Indicator) to use, rather than $\frac{r}{RMSE}$.

- Explore different options based on dataset and compression purpose. Some images with less contrasts might be easier to compress without too much loss, while highly accurate and contrasted photos may require to use a smaller r to prevent the lossy compression to be to significant.

References

[1] Article on different compression performance measures for image compression algorithm.