Digital Signal Processing: Image Compression via Transform Coding

Ronan McNally *(rm4064)*
Columbia University in the City of New York

I. Introduction

A vital application of digital signal processing in today's increasingly data-driven world is the ability to faithfully represent a signal without needed to access or maintain the all the data the signal has to offer. This is paramount to the transmission and storage of large volumes of data across various platforms efficiently. One such manifestation of this application is image compression. Here, the image is represented as a two-dimensional signal (dimensions of width and height of the image), though one could consider it a three dimensional signal if one were to include the red, green, and blue color channels as another dimension. There are a variety of transforms which can be employed to represent the image in a reduced form while still being able to maintain perceptual quality upon recovering the signal. The primary exploration in this paper is that of the discrete Fourier transform (DFT) with a tangential exploration of the learned transform via orthobasis learning.

II. Results

A. Implementation

The exploration of image compression via DFT began by making a transform coding pipeline. In this pipeline, I begin by uploading an image. This image is to be divided into a number of patches varied based on user input into the function. Instead of specifying the patch variable in terms of pixel width or height, my patch variable reflected the number of rows and columns the image would be divided into. I resized all my images to be in the shape of a square whose side lengths are divisible by powers of two for ease of computation. Hence, common patch values would be 4, 8, 16, and 32, reflecting 15 patches, 64 patches, etc. Each of these patches would be further divided by the color channel. This is because I chose to make use of the MATLAB commands *fft2()* and *ifft2()* for my DFT and inverse DFT applications. This MATLAB function only works on two dimensional signals (grey-scale only), and so I had to divide up my image patches along their color channels. At this point, I would apply the DFT to each patch component. Using the MATLAB commands *abs()* enabled me to then compare the DFT patch to the input threshold value. By comparing the patch matrix to the threshold value, I could use the resulting matrix of ones and zeros whose locations correspond to their meeting of the Boolean condition applied to then filter out the DFT values below the specified threshold by doing cell-by-cell multiplication. From here, each color channel of each patch could be inverted. Then, using two for loops I could assign the DFT patches and inverted patches to the transform coefficients matrix and compressed image values matrix, respectively. Once applying *uint()* and *abs()* to this compressed image matrix, I could then use *imshow()* to display the resulting compressed image. These variables, alongside calculations for peak signal to noise ratio, root mean square error, and fraction of non-zeroes are what enabled my further analysis.

B. Quantitative Evaluation

The quantitative evaluation of the performance of my image compression code was performed on four separate images in order to reflect how different image characteristics can affect the performance and resulting image quality of a transform even when it is performed with the same number of patches and threshold. As discussed during office hours with Professor Wright, the Fourier transform is more effective on smoother, periodic samples. Samples with piece-wise periodicity may not respond as well. Additionally, given that the JPEG files being used in this analysis may already have experienced image compression (as JPEG is a form of image compression itself), larger image files may show more improvement compared to smaller image files. For this reason, I chose an image of a cartoon sunset, a real sunset, an image of The Portrait of Adele Bloch-Bauer by Gustav Klimt and the Mona Lisa by Leonardo da Vinci as different images with varies degrees of complexity, contrast, color, and periodicity. I used the following equations to calculate peak signal to noise ratio (PSNR), root mean square error (RMSE), and fraction of non-zeroes (FNZ) respectively.

$$PSNR = 20log_{10}(M/RMSE)$$

$$RMSE = (\frac{1}{\text{whc}} \sum |I - \hat{I}|^2)^{\frac{1}{2}}$$

$$FNZ = \frac{\text{number of non-zero Xcoefficients after filtering}}{\text{total number of Xcoefficients}}$$

Using the RMSE equation to find PNSR, I then used tau values of 500, 1000, 2000, 4000, 8000, and 160000 with a patch value of 8 (i.e. 64 patches) to develop PSNR vs FNZ plots for each image I used. This figure can be seen on the next page. Both the Mona Lisa and the Portrait of Adele have greater x- and y-axis limits given their greater sizes. It would appear though that the images which are easier to compress are the sunset photos followed by the Mona Lisa and then the Portrait of Adele. The first sunset photo is a bit muted in color and has more gradual changes apart from a component of the image where waves are breaking. The cartoon sunset, while it does have sharp changes that would make it more difficult for the Fourier transform, has large regions of gradual color change to compensate. Finally, the Mona Lisa's generally more muted and less chaotic color scheme lend it to not having poor compressibility. The Portrait of Adele does poorly here as I believe the portraits complexity yields difficult to compress behavior in the signal.

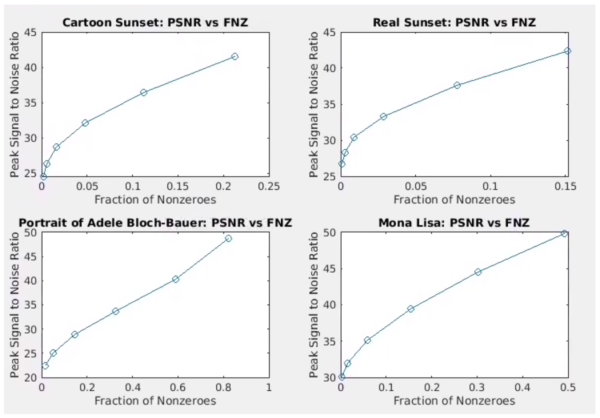

Fig. 1. PSNR vs FNZ plots for each image.

C. Visual Evaluation

Running compressions with patch value 4 (i.e. 16 squares) and taus of 2000, 4000, 8000, and 16000 from top to bottom, I generated the visual results of compressing two of my images. These results can be seen in the figure to the right. Upon inspecting the images, it would appear sharp areas of contrast such as the white text on a colored back seen in the sunset image at the bottom are most sensitive to quantization errors. Additionally, the edges of the patches appear to be perceived as sharp changes to, as it is shown that visually different features of a patch occurring on opposite ends of the patch will appear on the patch's opposite side. The DFT's operation at the edges of patches likely causes border opposite to each other to mix together.

D. Role of Transformation

To see the role transformation can have on the compressed image, in this part of exploring the compression of my images, I replaced the Fourier transform and its inverse with that of the identity transformation. Applying this change to my pipeline to my image of the Portrait of Adele Bloch-Bauer yields the PSNR vs FNZ curve seen in Figure 3. One notices upon inspecting this curve that the ratio has worsened as the PSNR has decreased more than the corresponding drop in FNZ.

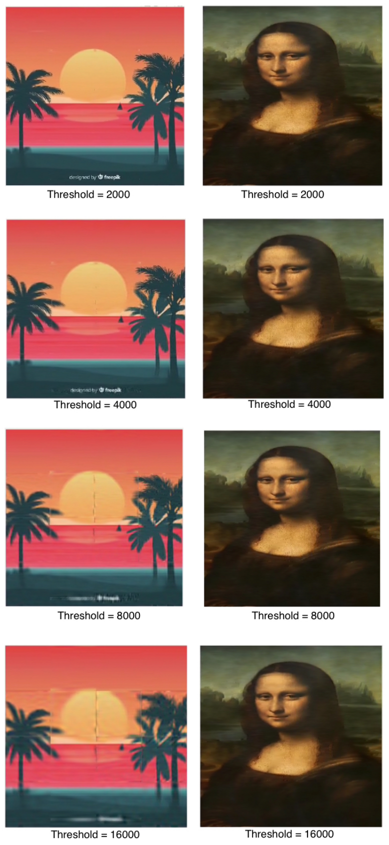

Fig. 2. Change in Visual Quality with respect to an increasing threshold applied to two images.

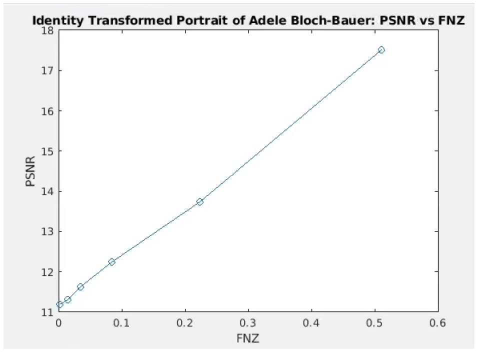

Fig. 3. Resulting PSNR vs FNZ curve when compressing the Portrait of Adele Bloche-Bauer image by identity transformation rather than Fourier.

E. Varying Patch Size

Returning to the Fourier transformation method, I then explored what the impact of varying the number of patches used to divide up the images would have on the PSNR vs FNZ curves. Applied this to a single one of my images, The Portrait of Adele Bloche-Bauer at a single threshold value. What the single plot does is track where the respective ratio between PSNR and FNZ is for each patch size. This plot can be seen in Figure 4 on the next page. The plot indicates the an increase in number of patches has positive returns in the beginning, but experiences diminishing returns after patch size 8 until no longer being worthwhile. With this in mind, I would choose a patch size of 8.However, this will vary from image to image. The function of these patches is to help better catch variation in the signal across the width and height of an image. Images varying in complexity (and where that complexity is located) are going to differ in terms of the number of patches they need.

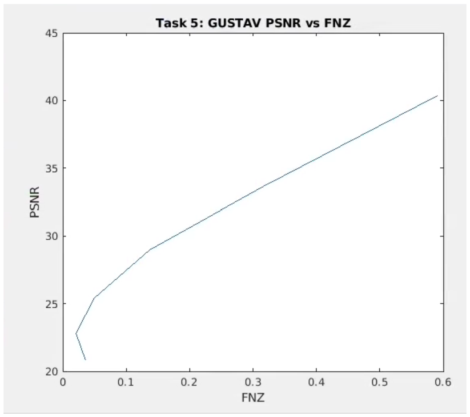

Fig. 4. Curve tracking the ratio between PSNR and FNZ for The Portrait of Adele Bloche-Bauer image. From bottom left to top right along the curve, each corner point reflects a size of 2, 4, 8, 16, and 32.

F. Extension - Transform Learning

In this final section, I incorporated the provided code on Courseworks which performs orthobasis learning into my own pipeline in order to see how its might impact performance. Of particular interest to me was how it would respond to areas of high contrast as the learning model could dedicate more focus towards areas of high contrast while diverting attention away from areas of little contrast. I would expect this to improve it's performance on some of my images. Initial testing with its performance on compression does not give a definite answer. It's clear the function pays close attention to the white text in the sunset image I'm using as it's an area of contrast, but it appears to perform overall worse when reconstructing the image. This comparison can be seen between the original and the learned transform via Figures 5 and 6 respectively. This intuition was confirmed when I compared the PSNR:FNZ of the previous transform with that of the learned transform. Beforehand, under the conditions of it being the cartoon sunset image with a patch size of 32 and threshold of 4000, the ratio was around 26.2645:0.0049. Now with the learned transform, the ratio is instead 14.8825:9.9817. Thus, it would appear the trade-off from the learned transform is worse.

III. Limitations and Potential Future Work

The scope of this project was limited to idealized conditions and the exploration of these image compression techniques

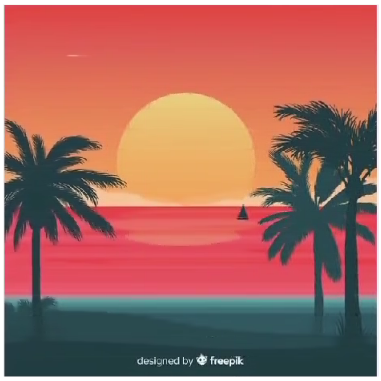

Fig. 5. DFT compressed performance. Text is still fine under the same conditions.

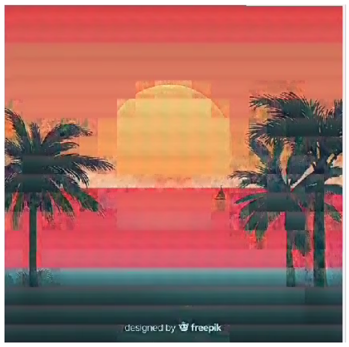

Fig. 6. Learned transform compressed performance. It's clear the text was prioritized, but the sacrifice in other regions of the image appear great.

made assumptions in its analysis. To begin, each image is a JPEG found online. While the inferred visual information of each JPEG still bears relevance to how the Fourier transform responds to it, it is unknown how these online images have already been compressed so as to be stored and transmitted online. Without being aware of what these conditions may be, there is a possibility that variation in precompression across my images may have impacted the results. This limitation informs the potential future work one could have on this project. The conventional JPEG used the discrete cosine transform to perform its compression. Future work could entail exploring what variables have an impact on the direct cosine transform's fidelity and how those factors can relate to those, like patch size, explored here.