# Learning Wavelet Filters from Fully Connected Networks for Image Compression

Justin Tsun Ko
Keble College
University of Oxford

A report submitted in partial fulfillment of the requirements for the degree of
*Master in Engineering*
Trinity 2021

**Acknowledgements**

I would like to thank my supervisor, Prof. Armour for his guidance and motivation. I would also like to thank my family and friends for their endless support throughout the project. Last but not least, I would like to thank my dog QQ.

**Abstract**

This report considers the possibility of applying machine learning methods to image compression. Fully Connected Networks were used to output wavelet coefficients, which can be used in the discrete wavelet transform stage of the JPEG 2000 image compression scheme. New wavelet filters which can achieve non-trivial latent decomposition and near-perfect reconstruction have been discovered. This project has demonstrated that it is possible to differentiate parameters such as reconstruction and compression performance with respect to individual wavelet filter coefficients.

# Contents

# Section 1

# Introduction

Compression is important. Images, videos and songs make up the bulk of all data storage and transmission use [1]. Without compression, storage and transmission of said data would quickly become costly and infeasible. Fortunately, most image, audio and video data sets exhibit some form of statistical redundancy, which can be exploited by compression algorithms.

For images, the most widely adopted compression scheme is JPEG [2], which is discrete cosine transform [3] based. JPEG 2000 [4], a discrete wavelet transform [5] based, often overlooked younger brother to JPEG, can achieve 20-200% more compression (in terms of bits per pixel) than JPEG, while maintaining the same image quality [6].

This begs the question: If JPEG 2000 (JP2) is superior to JPEG (JPG), Why hasn't JP2's adoption surpassed JPG? One of the main arguments is: JP2 requires more memory and computing resources than JPG [7], posing as an impracticality for consumer-grade machines at the time of its introduction in 2000.

However, as we are equipped with much more capable modern processors and memory devices now [8], it might be high time to revisit JP2 and the discrete wavelet transform (DWT).

This project aims to strengthen the case for JP2's (and by extension the DWT's) revival by combining it with recent developments in the machine learning field. This is attempted by using a neural network to predict wavelet coefficients for use in the DWT stage within the JP2 pipeline. The goal is to predict wavelet coefficients that outperform the stock Cohen–Daubechies–Feauveau (CDF) 9/7 wavelet (A wavelet specifically designed for lossy compression in JP2 [9]) in bits per pixel performance, while having acceptable reconstruction, measured in terms of Structural Similarity Index Measure (SSIM) [10].

The code written for this project has been uploaded to a GitHub repository for reference [11].

# Section 2

# Literature Review and Prerequisites

This section compares modern image compression schemes and describes the pipeline within the JP2 standard. This is supplemented with brief introductions to the theory behind JP2, namely the DWT, and the machine learning concepts utilised in this project, such as artificial neural networks. Equipped with relevant prerequisite information, we then take a dive into the literature to explore further developments in JP2, wavelet theory, and image compression in a machine learning context.

## 2.1 Prerequisites

### 2.1.1 Modern Image Compression Schemes

Older image compression schemes such as the Graphics Interchange Format [12] (GIF) and Tag Image File Format (TIFF) [13] tend to rely on entropy encoding algorithms such as Lempel–Ziv–Welch (LZW) [14] or Run Length Encoding (RLE) [15] to achieve compression. Modern image compression incarnations such as JPG and JP2 pair transform techniques such as discrete cosine transform (DCT) and DWT with specialised entropy encoding paradigms such as Embedded Block Coding with Optimised Truncation (EBCOT) [16] to achieve superior compression. Recently, WebP [17] has added block prediction for even better compression. Table 1 summarises the results.

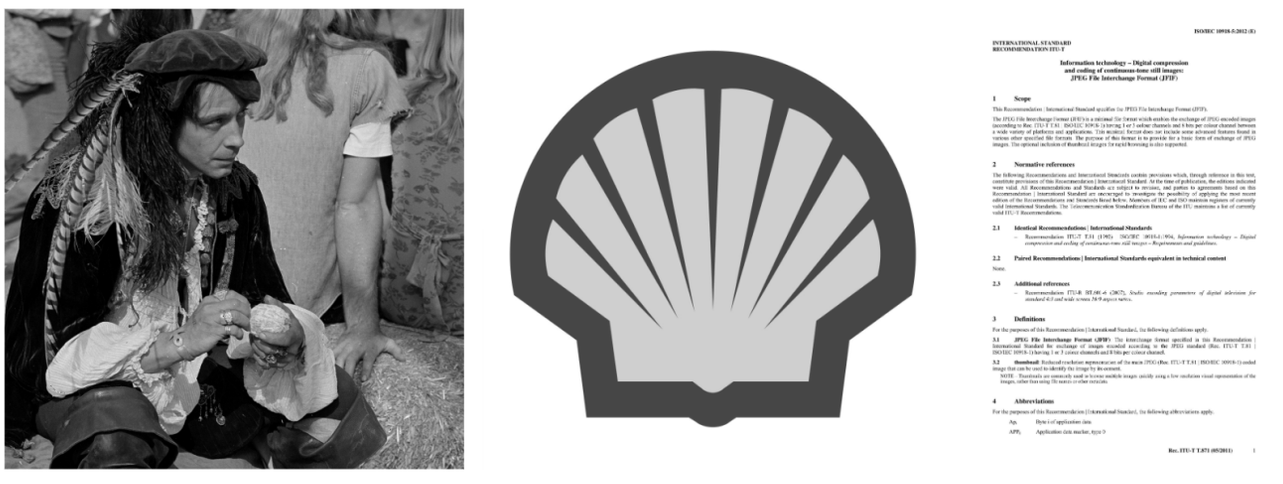
| Compression Scheme | Release Year | Transform Method | Encoding Scheme |
|---|---|---|---|
| WebP | 2010 | DCT | Arithmetic |
| JP2 | 2000 | DWT | EBCOT |
| PNG [18] | 1996 | - | DEFLATE |
| JPEG | 1992 | DCT | RLE |
| GIF | 1987 | - | LZW |
| TIFF | 1986 | - | RLE |

Table 1: Comparing transform and encoding scheme adoption across different image compression schemes.

It should be noted that different compression schemes satisfy different needs; compression performance is only one of these needs. For example, TIFF was developed for true lossless compression, while JPG was developed for lossy compression without substantial decay in quality. GIF has a 256 colour pallet limit per arbitrary tile [12], which drastically increases compression performance; ideal for graphics (such as logos), but fail to recreate colour gradients for natural images (such as human faces).

### 2.1.2 Compression Scheme Comparison

In the literature, compression performance is consistently mentioned without mentioning what kind of image the compression was applied to. In this brief experiment, image types proposed by Taubman in his definitive JP2 text [19] are used, including: Natural, graphical and text images. Examples of each are shown in Figure 1. Ten images from each type will be compressed using aforementioned compression schemes and the resulting average bits per pixel will be compared. Thumbnails for the dataset are shown in Appendix A.



(a) (b) (c)

Figure 1: Examples of different image types. (a), (b) and (c) are examples of natural, graphic and text images respectively.

Attention was taken to ensure images came from uncompressed sources. For natural images, the USC-SIPI database [20] was used. For graphics, TIFF images were created using Scalable Vector Graphics (SVG) files. Images with an alpha channel colour space, which define transparency, were not used. For text, TIFF files were generated directly from word documents. All images were converted to grayscale, to prevent colour transforms such as YCbCr to affect pure compression performance. All compression apart from JPG was done losslessly, to allow for fair comparison (lossless compression is not available due to the nature of the DCT). However, it should be noted that the term 'lossless' only signifies losslessness within the bit depth allowed in the compression scheme. As the bit depth

bit depth signifies how much colour is available at each pixel.

utilised in each implementation is not clear, the following analysis should only be treated as a guide. The program XnConvert [21] was used for compression, and the results are shown in Table 2.

| | TIFF | JPEG* | JP2 | WebP | GIF | PNG |
|---|---|---|---|---|---|---|
| Natural | 7.8 | 6.0 | 4.9 | 4.6 | 7.2 | 4.8 |
| Graphics | 3 | 0.22 | 0.59 | 0.24 | 0.22 | 0.28 |
| Text | 0.57 | 0.60 | 1.1 | 0.24 | 0.54 | 0.38 |

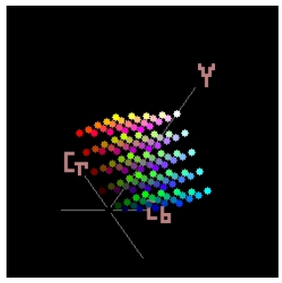



Table 2: Table comparing the bits per pixel measurement of different schemes on different image types. Lower is better. *For JPEG, only lossy compression could be carried out.

It is evident that JP2 performs well in natural images. However, its performance on graphical and text images is lackluster, clearly falling behind JPEG. Reasons for this will be further explored in the following section, where we take a deeper dive into the JP2 pipeline.

Dynamic range is the ratio between the largest and smallest values that a certain quantity can assume.

### 2.1.3 JPEG 2000 Standard

The JP2 standard was developed to succeed JPG. It has been designed for better compression performance, region of interest coding [22], multiple resolution representation / progressive transmission, and high dynamic range support. Its pipeline is shown in Figure 2. It should be noted that for coloured images, a RCbCr transform will take place (shown as 'color transform' in Figure 2). However, as only grayscale images will be considered, this component in the pipeline can be ignored in the scope of this report.

YCbCr

not relavant

how it affects

Compression

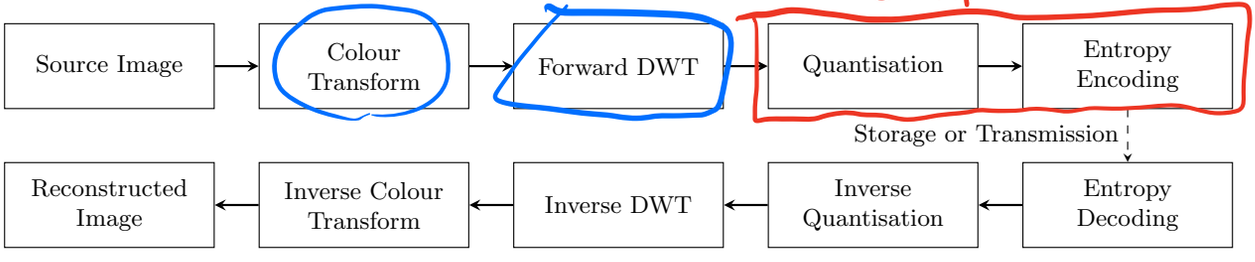



Figure 2: JPEG 2000 Pipeline. This figure is a reconstruction from Taubman's *JPEG 2000 Image Compression Fundamentals, Standards and Practice* [19].

One interpretation of the JP2 system is: The forward DWT warps the image into a sparse latent space; the latent space representation is then quantised and fed into a specialised entropy encoder, to achieve compression.

Standard wavelet.

In practice, the forward DWT is performed using the CDF 9/7 wavelet (For lossy compression) or the LeGall-Tabatabai (LGT) 5/3 wavelet (Coupled with integer to integer mapping for lossless compression [23]). They will be referred to as CDF and LGT for the rest of this report. The lackluster



Want to reduce entropy.

performance of JP2 on graphical and text images in the previous section could be explained by the inability of the LGT wavelet to facilitate an adequately sparse latent space representation.

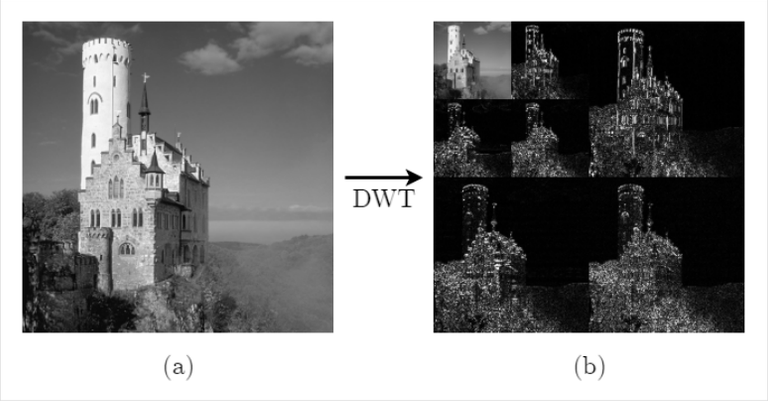



Figure 3: 2D Wavelet Transform. The original image is shown in (a), while its transformed representation is shown in (b), in the form of Mallet's Depiction [24].

It should be noted that the latent space representation in Figure 3b has exactly the same number of pixels as the original image; therefore no compression has happened during the transform. Compression is really carried out by quantisation and encoding.

The encoding stage is then carried out by the MQ Encoder [25], which follows the EBCOT paradigm. EBCOT is successor to previous encoding algorithms such as EZW and SPIHT [12]. Put simply, the MQ encoder exploits the sparsity and statistical redundancy between transform coefficients in different levels of decomposition, illustrated in Figure 4.

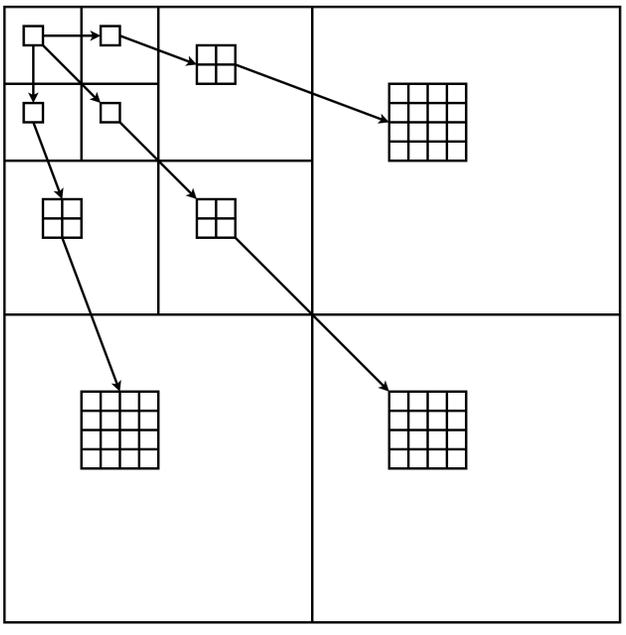


Figure 4: Diagram showing entropic preferences of the EZW scheme [6].

After encoding, the original image can now be fully represented by a bit stream. Storage and transmission has become much easier. To access the image again, we simply perform the inverse of the aforementioned operations.

### 2.1.4 Continuous Wavelet Transform

The Continuous Wavelet Transform (CWT) (2.3) was developed to combat poor temporal representation of the Fourier Tranform (FT) (2.1) and Short-time Fourier Transform (STFT) (2.2).

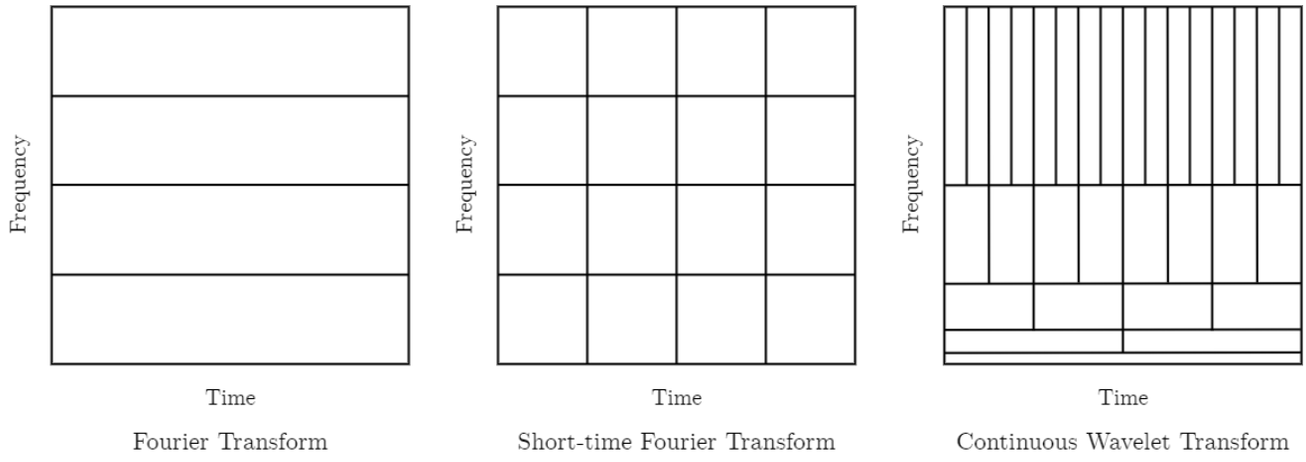



Figure 5: Heisenberg planes for the Fourier Transform, Short-time Fourier Transform and Continuous Wavelet Transform.

$$F(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-i\omega t}\, dt \tag{2.1}$$

$$F(\tau, \omega) = \int_{-\infty}^{+\infty} f(t) w(t-\tau) e^{-i\omega t}\, dt \tag{2.2}$$

$$F(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{+\infty} f(t) \psi^*(\frac{t-\tau}{s})\, dt \tag{2.3}$$

It is obvious from the Heisenberg plane of the Fourier Transform (2.1) in Figure 5, that the Fourier Transform offers no temporal resolution. This manifests in rectangular boxes with infinite width on the Heisenberg plane.

The STFT attempts to tackle this by introducing a window function $w$ and time shift parameter $\tau$. This leads to rectangular boxes with finite height and width on the Heisenberg plane. However, the STFT has its limitations. This is outlined by the observation that high frequency components appear in short bursts, meaning it would be more helpful if rectangles on the top of the Heisenberg plane were narrower. Low frequency components, on the other hand, take longer to happen; having wider rectangles on the bottom of the Heisenberg plane would allow for better detection of these signals.

The wavelet transform addresses this limitation by introducing an additional scaling parameter $s$. The scaling coefficient allows different time resolution at different frequencies, resulting in an optimal Heisenberg plane shown in Figure 5.

There are many types of wavelets for many applications, such as numerical analysis [26], seismic signal analysis [27] and of course, image compression, to name a few. However, to apply the CWT

transform to real life problems, the theory must be manifested in discrete space for computer systems.

### 2.1.5 Discrete Wavelet Transform

The CWT (2.3) can be modified to become the DWT (2.4) by replacing the integral with a summation. It has also been derived that choosing the scale and shift based on dyadic scales, or powers of 2 yields critically sampled wavelet coefficients [28]. This represents the most compact wavelet representation of a signal without losing information.

$$F(\tau, s) = \frac{1}{\sqrt{|s|}} \Sigma_{m=0}^{p} f[t_m] \psi[\frac{t_m - \tau}{s}] \tag{2.4}$$

We have hitherto only considered the 1D DWT. Luckily, translation to 2D is simple. We simply perform filter bank operations [29] on the rows of an image to obtain a semi-transform, then repeat for the columns of the semi-transform. We are presented with $cA$, $cD$, $cH$, $cV$ coefficients, instead of obtaining just $cA$ and $cD$ coefficients in the 1D transform.

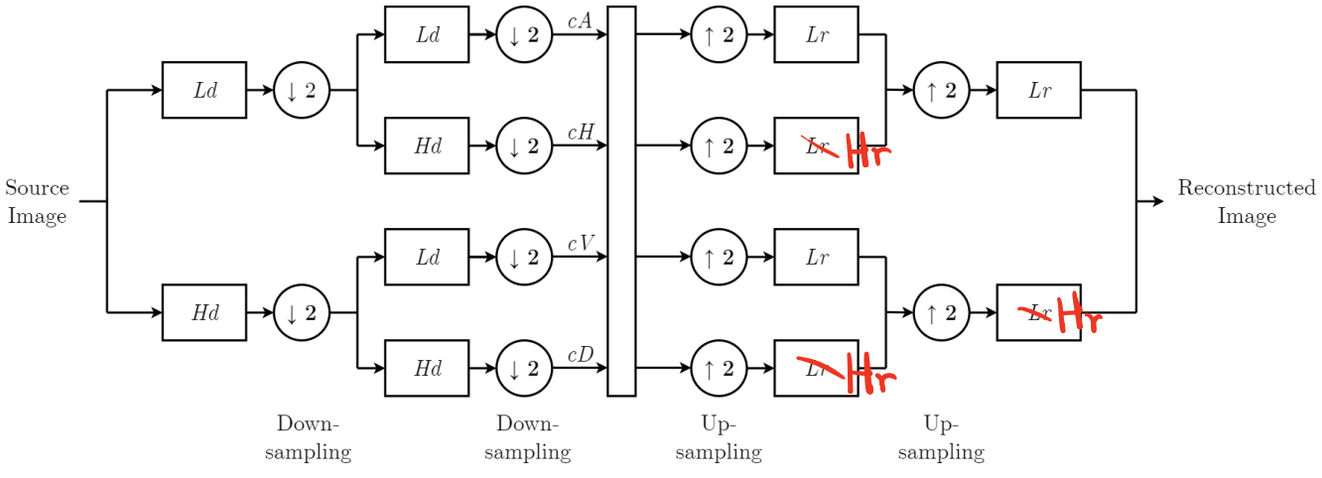



Figure 6: Filter bank for a single level 2D DWT [29].

To obtain the original signal again, the filter bank operations are reversed and reconstruction filters obtained from conjugate quadrature filter relations [24] are used. Here is a simple example:

If we are given a 4 coefficient low pass decomposition filter **Ld** and low pass reconstruction filter **Lr**, shown as:

$$\mathbf{Ld} = \begin{bmatrix} Ld_1 & Ld_2 & Ld_3 & Ld_4 \end{bmatrix} \tag{2.5}$$

$$\mathbf{Lr} = \begin{bmatrix} Lr_1 & Lr_2 & Lr_3 & Lr_4 \end{bmatrix} \tag{2.6}$$

We can then express components in the high pass decomposition filter **Hd** and high pass reconstruction filter **Hr** in terms of $Ld_n$ and $Lr_n$ coefficients:

$$\mathbf{Hd} = \begin{bmatrix} -Lr_1 & Lr_2 & -Lr_3 & Lr_4 \end{bmatrix} \tag{2.7}$$

$$\mathbf{Hr} = \begin{bmatrix} Ld_1 & -Ld_2 & Ld_3 & -Ld_4 \end{bmatrix} \tag{2.8}$$

For JP2, the CDF and LGT wavelet filters are used. They are designed to be short, symmetric and hold the maximum number of vanishing moments given their length [30]. These considerations allow for great reconstruction performance and avoids boundary artefacts.

Other considerations of JP2 wavelet design include order of approximation [31], Riesz Bounds [32], and basis function smoothness [33]. In this project, we will disregard these properties of wavelet design. Normally, wavelets are designed from polynomial basis functions in z-space, then sampled to construct filters [9]. However, in this project we will consider a bottoms up approach: generating discrete wavelet filters without considering any continuous function or accompanying rules.

### 2.1.6 Artificial Neural Networks

Artificial neural networks are machine learning algorithms developed to mimic biological neural structures [34] in the hopes of recreating intelligence. Whether intelligence is indeed recreated is still a hot topic of debate [35]. An example of a fully connected neural network is shown in Figure 7.

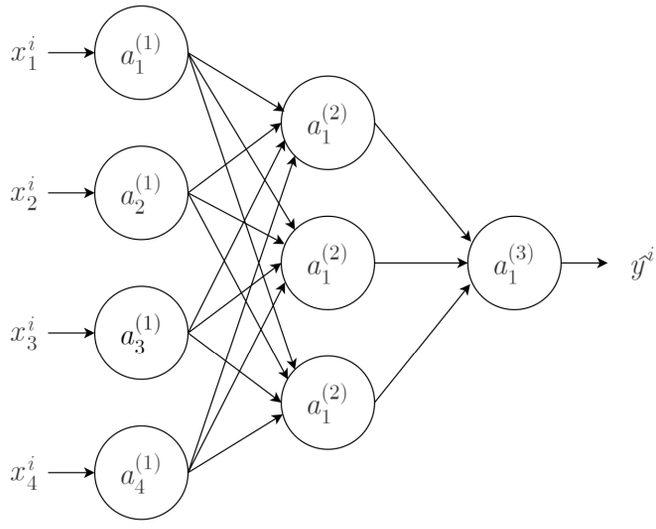



Figure 7: Diagram showing a fully connected neural network, with 4 neurons in the first layer, 3 neurons in the second layer and 1 neuron in the last layer.

Usually, training data **x** is fed into a network and propagated through weights **W**, biases **b** and

activation function $g$ (2.9), obtaining an output $\hat{\mathbf{y}}$. The cost $C$ is obtained by comparing network outputs with labelled ground truth $\mathbf{y}$ (2.10). The cost is then used to offset weights and biases via backpropagation [36]. After training, a validation data set is used to validate the performance of the network and gauge overfitting [37].

This project did not do classification

$$\mathbf{a}^l = g^l(\mathbf{W}^l \cdot \mathbf{a}^{l-1} + \mathbf{b}^l) \qquad \text{Where } l \text{ denotes the } l^{th} \text{ layer.} \tag{2.9}$$

$$C(\hat{\mathbf{y}}, \mathbf{y}) = \|\mathbf{y} - \hat{\mathbf{y}}\| \tag{2.10}$$

Output layers are grouped into 2 sets: classification and regression. An example of classification is predicting whether a dog is in an image. An example of regression is guessing a person's age by looking at an image.

In recent years, machine learning research and adoption has exploded due to lower cost of computing [8], efficient algorithm implementations such as backpropagation [36] and an increased prevalence of data [38]. Popular frameworks include convolutional neural networks (CNNs), which excel at image recognition and segmentation [39] and long short term memory (LSTM) in recurrent neural networks (RNNs) for natural language processing [40], amongst many others. Specialised network layers such as batch normalisation [41], dropout [42] and softmax [43] also exist for network stability and prevent overfitting.

In the context of this image-based project, using a CNN might be the obvious choice. However, with the observation that the 2D DWT actually consists of two 1D transformations in series, coupled with the fact that wavelet convolutions are being carried out on dyadic scales, a CNN might not be the correct tool. A fully connected network was chosen instead.

→ ?

## 2.2 Literature Review

### 2.2.1 JPEG 2000 Developments

A review of JP2 related publications has shown that learning wavelets has not been explored within the JP2 community. Experimentation within JP2 with wavelet filters outside of CDF and LGT is nonexistent. → Everyone else was just using CDF and LGT wavelets!

The majority of developments in JP2 are about standardisation. JPIP was developed to standardise client-server interactions on JP2 content, particularly for multi-resolution and spatially random access tasks [44], while JPSEC was developed for content protection, data integrity checking, authentication,

and conditional access control [45]. With increasing adoption of wireless communication, which is inherently noisy and error prone, the JPWL standard was developed to equip JP2 with error protection and correction tools such as forward error correcting codes and unequal error protection [46].

The topic of software and hardware implementations of JP2 also make up a substantial portion of JP2 related literature. Popular software implementations include JasPer [47] and OpenJPEG [48], while VLSI implementations [49] headline hardware implementations of JP2. Unfortunately, none of the implementations allowed custom wavelet filter coefficients; only the CDF and LGT filter sets were allowed.

Improvements to JP2's core functionalities are comparatively limited. An example of this is the development of EBCOT [16], designed to provide progressive transmission and decomposition capabilities to the encoded bitstream, which SPIHT (EBCOT's predecessor) could not [50]. Another improvement is the implementation of the lifting scheme inside the DWT part of JP2, reducing hardware cost and achieving higher hardware utilisation [51].

### 2.2.2 Wavelet Developments

It appears that the idea of experimenting with wavelet filters in image compression has predated the development of JP2. Villasenor *et al.* [52] experimented with this idea by generating over 4300 wavelets from impulse response, step response and regularity requirements to compress images.

Thielemann [53] has furthered this idea by turning it into a linear least squares optimisation problem, with the help of the lifting scheme. It should be noted that while Thielemann's work focuses on using wavelets to approximate the shape of a arbitrary signals, and not explicitly on image compression, the blending of optimisation and wavelet generation in his work still provides valuable insights on image compression applications. Similairly, Chapa and Rao [54] have matched wavelets to general signals by optimising w.r.t. a closed form solution that defines a scaling function spectrum from a wavelet spectrum.

More recently, Olshaunsen *et al.* [55] constructed wavelet basis functions from recursively combining a small number of spatial functions, based on maximum *a posteriori* (MAP) estimates that minimise code lengths after transformation. This was done specifically for natural images. It should be noted that although this achieved similair signal to noise (SNR) values to the Daubechies 6 wavelet, it was not clear whether this method achieves superior compression performance. It was also only compared with a very limited number of existing wavelets.

Further exploration of related wavelet text has shown that wavelet compression has been adopted for a range of alternate data types, such as digitised radiographs [56], ECG signals [57] and motion

capture data [58], amongst many others. This provides further motivation for the project, as tools developed here could be potentially be transferred to other compression tasks.

### 2.2.3 Machine Learning in Image Compression

Recently, deep learning techniques are being explored as viable means of image compression. Additional to compression performance, latent representations of images might also be beneficial for computer vision tasks [59]. The Joint Photographic Experts Group's (JPEG) recent Call for Evidence on learning-based image coding [59] might be a sign of a paradigm shift to learning-based methods in the image compression community.

Most publications inside this domain involve deep learning systems without consideration to prior image compression frameworks. Toderici *et al.* [60] have demonstrated that Recurrent Neural Networks can outperform JPG in terms of compression, without the aid of entropy encoding. Theis *et al.* [61] has further shown that Compressive Autoencoders can also be utilised to achieve superior compression performance to JPG. However, neither methods could compete with JP2's compression performance.

A small number of papers have tried to combine existing compression methods with recent deep learning tools. Akyazi Ebrahimi [62] used Convolutional autoencoders to compress wavelet decomposition representations of images. However, they were not able to outperform JP2 in terms of compression and reconstruction performance. Qiu *et al.* [63] utilised deep residual learning to recover heavily quantised DCT coefficients with respectable peak signal to noise ratio (PSNR) values of over 31 dB.

### 2.2.4 Literature Review Conclusion

Based on this literature review, it is clear that learning wavelets exhibit some potential for image and other types of compression, and there is a lack of exploration for this in the context of JP2. Although deep learning frameworks are seeing increased adoption in image compression, using deep learning for learning wavelet filters has not been done before. We have now set the scene for a project that combines a deep learning method (artificial neural network) with a former state of the art image compression scheme (JP2).

# Section 3

# Analysing Existing Wavelets

Before we venture to learning wavelets, we must consider a few questions. Is it possible to outperform the CDF and LGT wavelets for compression? Do different image types favour different wavelets for compression? To answer this, all wavelets in the Matlab Wavelet Toolbox [64] were used to compress the dataset shown in Appendix A.

## 3.1 Method

The compression performance of each wavelet will be gauged using the compression score $C$ (3.1) of the transform coefficients, while the SSIM [10] between the reconstructed and original image will be used to gauge reconstruction performance. Families of wavelets that will be tested include Daubechies [65], Biorthogonal [66], Coiflets [67], Symlets [68], Reverse Biorthogonal [69], Fejér-Korovkin [70] and Discrete Meyer [71].

SSIM: structural similarity index measure.

$$C = \frac{\text{Number of pixels that have 0 intensity}}{\text{Total number of pixels}} \times 100\% \tag{3.1}$$

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma xy + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\mu_x^2 + \mu_y^2 + C_2)} \tag{3.2}$$

Where $\mu_x$, $\mu_y$, $\sigma_x$,$\sigma_y$, and $\sigma_{xy}$ are the local means, standard deviations, and cross-covariance for the original image $x$, and reconstructed image $y$.

Ideally we would use an existing JP2 implementation such as OpenJPEG [48] or JasPer [47], and tweak it to allow for more wavelets outside of CDF and LGT. However, available JP2 implementations could not accommodate this modification. Therefore, a much more stripped down version of JP2 was built from the ground, its pipeline shown in Figure 8. As no separate implementations of JP2's non-linear quantisation scheme [19] were available, a simple scalar hard threshold of 1 was used instead.

This is a rough approximation of the JP2 standard, as statistical redundancies between transform coefficients are not exploited here.

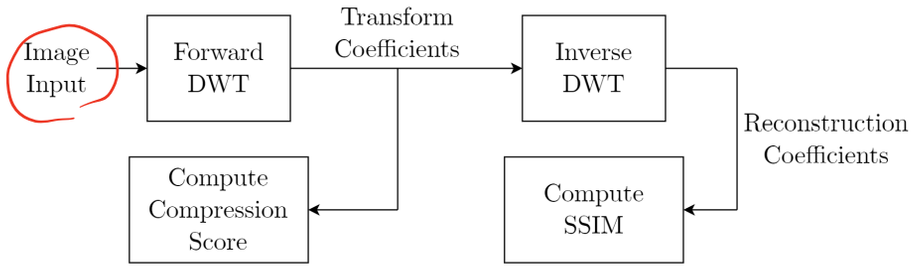



Figure 8: Pipeline for our approximation to the JP2 architecture.

## 3.2 Results

In terms of reconstruction, every wavelet has achieved an SSIM of 0.998 or above, with the vast majority being 0.99+. This means every wavelet has been able to reconstruct an image without perceptible difference to the original.

Results for compression are shown in Figure 9. Not surprisingly, the Biorthogonal family (where CDF and LGT belong) consistently outperform other families. This difference is reflected more in natural images, where the Index of Dispersion $D$ [72] in compression score is higher. However, in graphical and text images, where more regularity is expected, the $D$ values are lower, meaning the choice of wavelet seems to matter less. While Biorthogonal wavelets reign supreme for natural images, this advantage is much less apparent for graphical and text images.

| | Natural | Graphical | Text |
|---|---|---|---|
| $D$ | $4.2 \times 10^{-3}$ | $3.3 \times 10^{-5}$ | $3.1 \times 10^{-5}$ |

Table 3: Table comparing the Index of Dispersion $D$ across different wavelets, for different image types.

For this analysis, the Index of Dispersion $D$ [72] was used instead of the variance $\sigma^2$ to account for different scales in compression score for different image types. This can be thought of as a normalised variance.

$$D = \frac{\sigma^2}{\mu} \qquad \text{where } \sigma^2 \text{ is variance, and } \mu \text{ is mean.} \tag{3.3}$$

Based on Figure 9, CDF (bior4.4) and LGT (bior2.2) do not win all the time. They are consistently outperformed by other Biorthogonal wavelets. However, the fact that LGT performs better than CDF in this test might be an indicator of imperfections in the test. Nonetheless, this test has shown that the choice of wavelet does matter, especially for natural images. In the next section, we will explore the idea of *building* a wavelet, instead of choosing one.

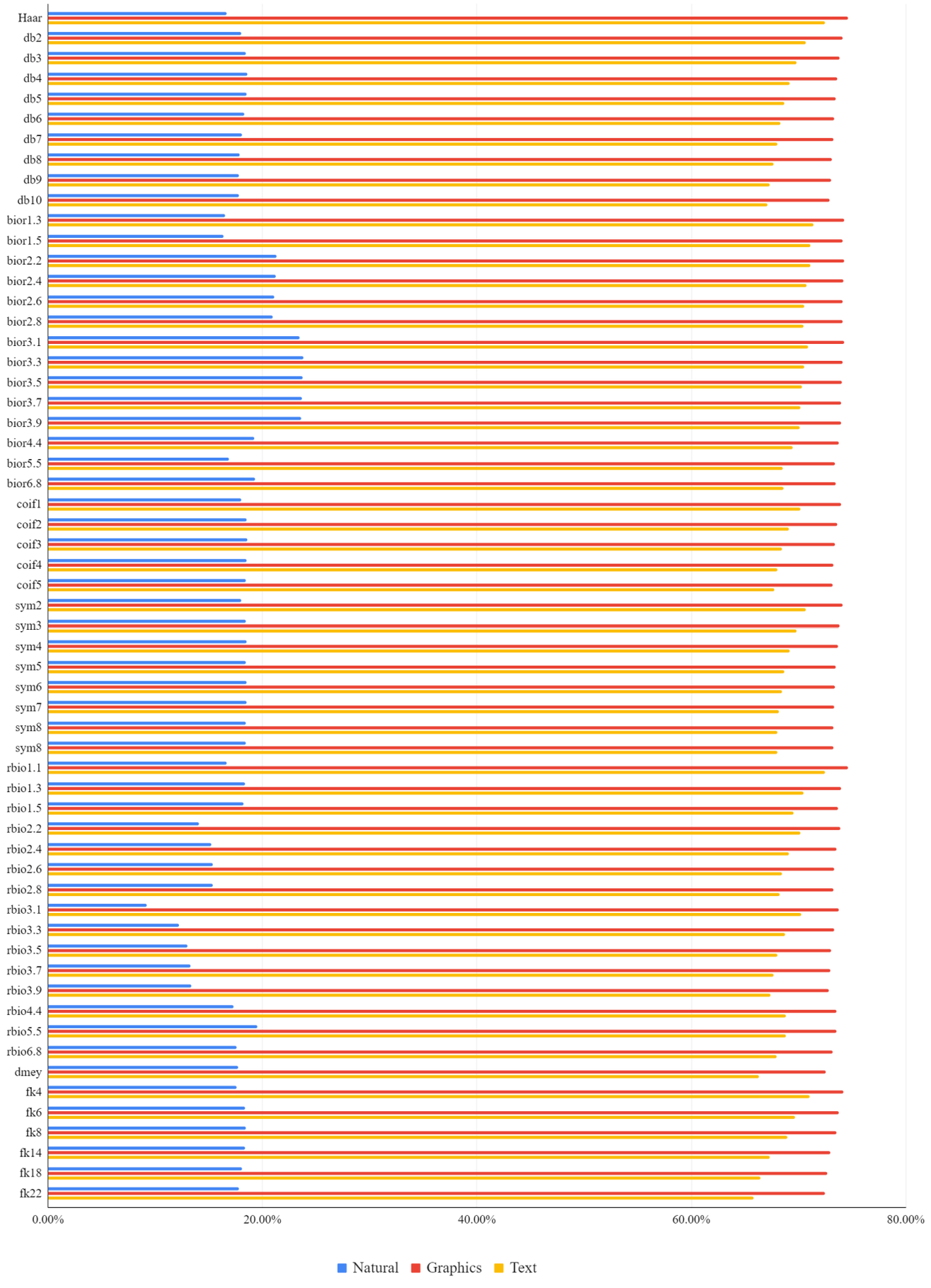



Figure 9: Chart comparing different wavelets. The horizontal axis represents the compression score $C$. The vertical axis indicates different wavelets, grouped into 7 families. The prefices 'db', 'bior', 'coif', 'sym', 'rbio', 'dmey' and 'fk' correspond to Daubechies, Biorthogonal, Coiflet, Symlet, Reverse Biorthogonal, Discrete Meyer and Fejér-Korovkin respectively.

# Section 4

# Learning Wavelets: Formulation

In the previous section we have received some affirmation on the viability of using different wavelets for different images. There are several paths to achieving this.

In a machine learning context, it could be treated as a classification problem. Neural networks could be trained to match images with certain wavelets. A labelled dataset could be obtained by repeating the pipeline in the previous section on more images.

It could also be treated as a multiple regression problem, where neural networks are trained to predict wavelet coefficients for images. This is shown in Figure 10.

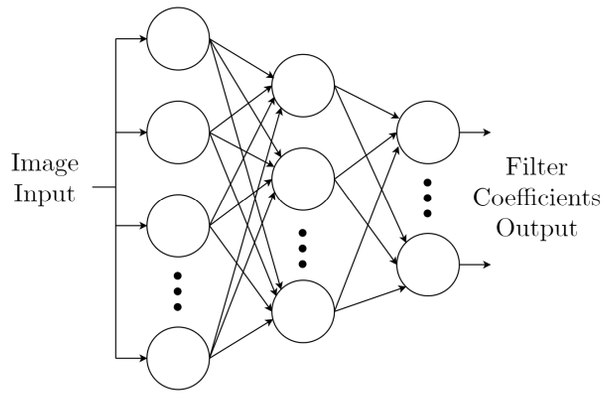



Figure 10: Diagram for a fully connected network that takes in images and outputs filter coefficients.

Solving the classification problem is not difficult in comparison to the regression problem; we simply need to match wavelets to images. The regression problem, on the other hand, is not trivial and holds the potential for the discovery of novel and specialised wavelets. For the rest of the report, we will consider the regression problem. In the next subsection we will take a slight detour in tiling; after that, we will formally dive into formulating a solution to the regression problem.

## 4.1 Tiling

So far, we have only considered images as a whole. However, in both JPG and JP2, tiling operations are done to reduce RAM usage. An example of tiling is shown in Figure 11.

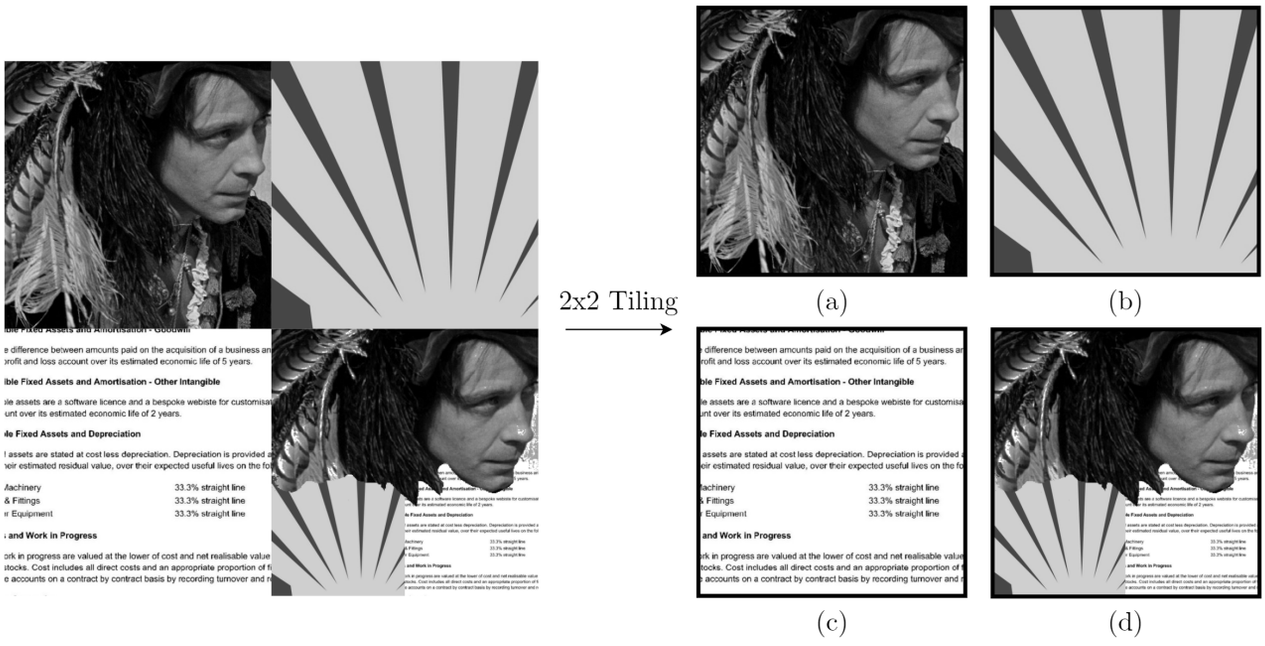



Figure 11: The image on the left indicates an original image with different textures. After tiling, it is separated into 4 parts on the right. (a), (b), (c) and (d) denote naturalistic, graphical, text and mixture tiles respectively.

This could be used to our advantage. Better compression could be achieved by using specialised wavelets for each tile. For example, assume the naturalistic tile (a) is best compressed by CDF, while the graphical tile (b) is best compressed by Haar, and the text-based tile (c) is best compressed by LGT. Instead of using CDF on all tiles (which JP2 does), specialised wavelets could be used on each tile individually to potentially achieve better compression.

Limitations exist. For example, it is very unlikely that different features will be able to be separated perfectly by square tiles. In real life, things might look more like tile (d): a mixture of components, each with different wavelet preferences.

This could be solved by segmenting the features and using separate wavelets to compress each segment, shown in Figure 12. Based on the assumptions made in the previous paragraph, (e) should be compressed by CDF, (f) by Haar and (g) by LGT. It should be noted that the segments should be imposed on blank rectangular frames before compression, because the 2D DWT only works for rectangular arrays. The extra memory required to store the extra blank pixels due to imposition on rectangular frames might become insignificant after encoding, but this has to be confirmed by experiment.

This could also be solved by decreasing the tile sizes. By doing this, a smaller proportion of tiles will be of type (d). However, this might increase the computational complexity and memory cost, as more wavelets will have to be learned, and more space will be required to store them. Compression performance might also decrease, as there could be less statistical redundancies within tiles. This might also introduce additional block artefacts.

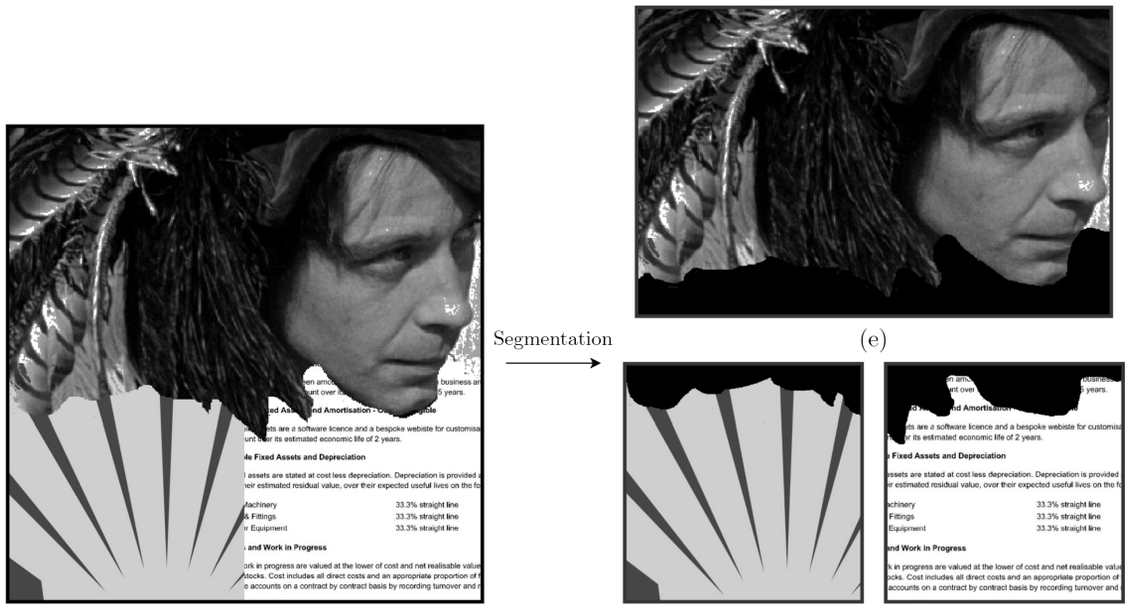



Figure 12: The image (d) is segmented and processed into (e), (f) and (g)

However, before these tiling operations can be considered, a working mechanism for generating wavelets for individual tiles must be present. Further considerations for tiling is outside the scope of the project, but it is an important concept to keep in mind for future work.

## 4.2 Proposed Pipeline

For this project, we will consider two primary aspects of compression: compression and reconstruction. Failure to adhere to any of them would result in corruption in data, or unsatisfactory compression performance.

For example, a system that only cared about reconstruction might predict an identity transformation; achieving perfect reconstruction, but produce a latent space representation with little sparseness. Conversely, a system tuned only for compression might predict no-pass filters that map any image to a blank latent space representation. Perfect for encoding, but no information is retained.

Aside from compression and reconstruction, other important factors include computational complexity and memory requirement. However, these topics are outside the scope of this project.

A system that attempts to incorporate these two pillars is proposed in Figure 13. Although the desire for features such as region of interest coding and multiple resolution representation were not manifested in the system, the very nature of JP2 implies these features will be conserved, irrespective of wavelet choice.

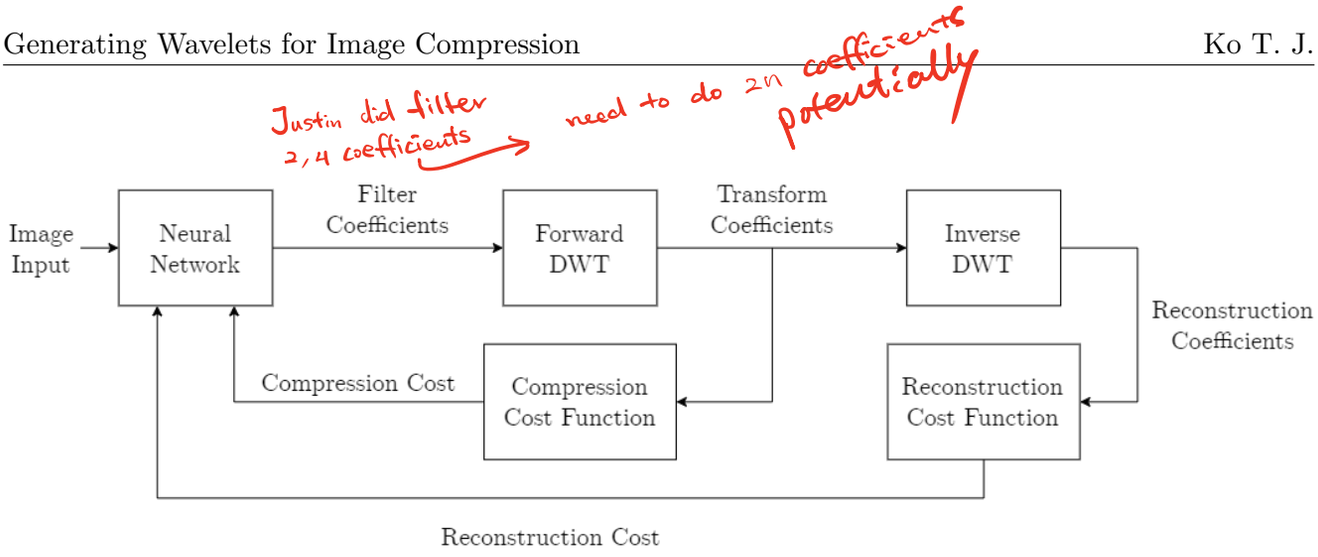



Figure 13: Schematic of the proposed pipeline for this project.

To eliminate redundancy, the output layer will only be used to define **Ld** and **Lr**. Conjugate quadrature relations will be used to infer **Hd** and **Hr**. This is explained in Section 2.4.

One of the most challenging aspects of the regression problem is defining a cost function that is able to differentiate the contribution of each output neuron to the overall compression or reconstruction performance. The following section will be devoted to slaying this beast.

# Section 5

# Reconstruction

In this section we will take a high level view on the derivation of a reconstruction cost function that is differentiable w.r.t. the outputs of the network, representing individual wavelet filter coefficients. The full derivations are shown in Appendix B. As we must always begin from the simplest case, the derivation will be performed on a 1D 1-level DWT using filters of length 2. After this, an extension towards filters of length 4 will be performed, providing evidence that this derivation could be extended to filter lengths of 2n. The cost function will then be used to train a neural network to reconstruct synthetic 1D signals, and stretched to tackle 2D images.

## 5.1 Defining Variables

$$\mathbf{I} = \begin{bmatrix} I_1 & I_2 & ... & I_n & ... & I_N \end{bmatrix} \quad \text{Original signal of length N} \quad (5.1)$$

$$\mathbf{I_d}(n) = \begin{bmatrix} I_{2n-1} & I_{2n} \end{bmatrix} \quad \text{General 2-part dyadic partition of original signal} \quad (5.2)$$

$$\mathbf{R} = \begin{bmatrix} R_1 & R_2 & ... & R_n & ... & R_N \end{bmatrix} \quad \text{Reconstructed signal of length N} \quad (5.3)$$

$$\mathbf{R_d}(n) = \begin{bmatrix} R_{2n-1} & R_{2n} \end{bmatrix} \quad \text{General 2-part dyadic partition of rec. signal} \quad (5.4)$$

$$\mathbf{Ld} = \begin{bmatrix} Ld_1 & Ld_2 \end{bmatrix} \quad \text{Low-pass decomposition filter of length 2} \quad (5.5)$$

$$\mathbf{Lr} = \begin{bmatrix} Lr_1 & Lr_2 \end{bmatrix} \quad \text{Low-pass reconstruction filter of length 2} \quad (5.6)$$

$$\mathbf{Hd} = \begin{bmatrix} -Lr_1 & Lr_2 \end{bmatrix} \quad \text{High-pass decomposition filter of length 2} \quad (5.7)$$

$$\mathbf{Hr} = \begin{bmatrix} Ld_1 & -Ld_2 \end{bmatrix} \quad \text{High-pass reconstruction filter of length 2} \quad (5.8)$$

$A_n$ Approx. coefficient at nth dyadic operation (5.9)

$D_n$ Detail coefficient at nth dyadic operation (5.10)

$y(Ld_i)$ Cost for $Ld_i$ (5.11)

Because of quadrature mirror relations, we can write down expressions for **Hd** and **Hr** in terms of coefficients in **Ld** and **Lr**. This is explained in Section 2.4.

## 5.2 Dyadic Operations

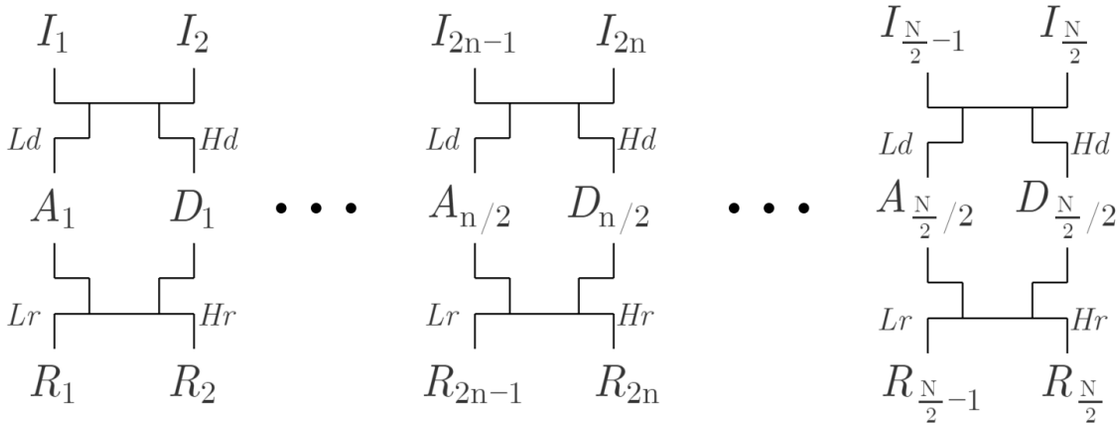



Figure 14: Illustration of dyadic operations.

In the construction of the cost function, we will exploit the dyadic nature of the DWT operations, shown in Figure 14. For example, if we know $I_1$, $I_2$, **Ld**, **Hd**, **Lr** and **Hr**, then we can work out $A_1$, $D_1$ and therefore work out $R_1$, $R_2$. In the general case, $R_{2n-1}$ and $R_{2n}$ can be expressed as functions of the previously mentioned variables (5.12) (5.13).

$$R_{2n-1} = f(I_{2n-1}, I_{2n}, Ld, Hd, Lr) \tag{5.12}$$

$$R_{2n} = f(I_{2n-1}, I_{2n}, Ld, Hd, Hr) \tag{5.13}$$

We are now ready to proceed to the derivation of the cost function.

## 5.3 Cost function

If we take partial derivatives to $R_n$ and $R_{n+1}$ w.r.t. each **Ld** and **Lr** component, we will arrive at (5.14). We have now made the reconstruction pixel's intensity differentiable w.r.t. to each wavelet

coefficient. But what is the cost to each wavelet coefficient? In other words, how do we decide the change in each wavelet coefficient?

$$\frac{\partial R_{2n-1}}{\partial Ld_i}, \frac{\partial R_{2n}}{\partial Ld_i}, \frac{\partial R_{2n-1}}{\partial Lr_j}, \frac{\partial R_{2n}}{\partial Lr_j} \tag{5.14}$$

We do this by taking the product of the displacement between $I_n$ and $R_n$, and each partial derivative, then sum it up over the entire signal for each wavelet coefficient (5.15) (5.16). We now have costs for each wavelet coefficient, ready to be sent back to a neural network for backpropagation.

$$y(Ld_i) = \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})\frac{\partial R_{2n-1}}{\partial Ld_i} + (R_{2n} - I_{2n})\frac{\partial R_{2n}}{\partial Ld_i}] \tag{5.15}$$

$$y(Lr_j) = \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})\frac{\partial R_{2n-1}}{\partial Lr_j} + (R_{2n} - I_{2n})\frac{\partial R_{2n}}{\partial Lr_j}] \tag{5.16}$$

To sum it up: within a dyadic operation, a reconstructed signal is determinate from the signal coefficients and the filter coefficients. This means the reconstructed signal can be expressed as a function of the signal coefficients and filter coefficients. If we take the partial derivative of the reconstructed signal w.r.t. the filter coefficients, then take the product of said partial derivative and the displacement between original and reconstructed pixel, then iterate along to the next dyadic operation until the entire signal has been processed, we will obtain a cost for each wavelet coefficient. Actual expansions are shown in (5.17) - (5.20).

$$y(Ld_1) = \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})(I_{2n-1}Lr_1) + (R_{2n} - I_{2n})(I_{2n-1} + I_{2n-1}Ld_1 + I_{2n}Ld_2)] \tag{5.17}$$

$$y(Ld_2) = \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})(I_{2n}Lr_1) + (R_{2n} - I_{2n})(I_{2n}Ld_1 + I_{2n-1}Lr_1 - I_{2n}Lr_2)] \tag{5.18}$$

$$y(Lr_1) = \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})(I_{2n-1}Ld_1 + I_{2n}Ld_2 - I_{2n-1}Lr_2) + (R_{2n} - I_{2n})(I_{2n-1}Ld_2)] \tag{5.19}$$

$$y(Lr_2) = \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})(-I_{2n-1}Lr_1 + 2I_{2n}Lr_2) + (R_{2n} - I_{2n})(-I_{2n-1}Ld_2)] \tag{5.20}$$

The full derivation of this is shown in Appendix B.1.

### 5.3.1 Translation to 4 coefficients

Being able to predict wavelet filters with only 2 coefficients is not enough. There are not enough degrees of freedom and we might be constrained with trivial identity transforms. We must move to 4

wavelet coefficients.

In the derivation for the cost function for 4 coefficients, the same logic as the 2 coefficient derivation is followed. However, the existing literature does not adequately explain how 2+ coefficient wavelet transforms work. The operations, which include padding and dyadic upscaling, were reverse engineered from a Matlab function that performs a 1D, 1 level DCT [73], and shown in the full derivation in appendix B.2.

To confirm the validity of the reversed engineered operations, a function carrying out the operations was written and trialed with random filter coefficients. The output of the function is then compared to the official Matlab code, which match every time. The difference between the official Matlab function and the reverse engineered code is: the reverse engineered code can only perform a 1D DWT with filter length 4, and when the signal length is a multiple of 4. The Matlab version is much more versatile with filter and signal lengths.

The cost functions for wavelet coefficients in a 1D DWT of filter length 4 are shown in (5.21) and (5.22). The full expansions are presented in Appendix B.2.

$$\begin{aligned} y(Ld_i) = \sum_{n=1}^{N/4} [&(R_{4n-3} - I_{4n-3}) \frac{\partial R_{4n-3}}{\partial Ld_i} + (R_{4n-2} - I_{4n-2}) \frac{\partial R_{4n-2}}{\partial Ld_i} \\ &+ (R_{4n-1} - I_{4n-1}) \frac{\partial R_{4n-1}}{\partial Ld_i} + (R_{4n} - I_{4n}) \frac{\partial R_{4n}}{\partial Ld_i}] \end{aligned} \tag{5.21}$$

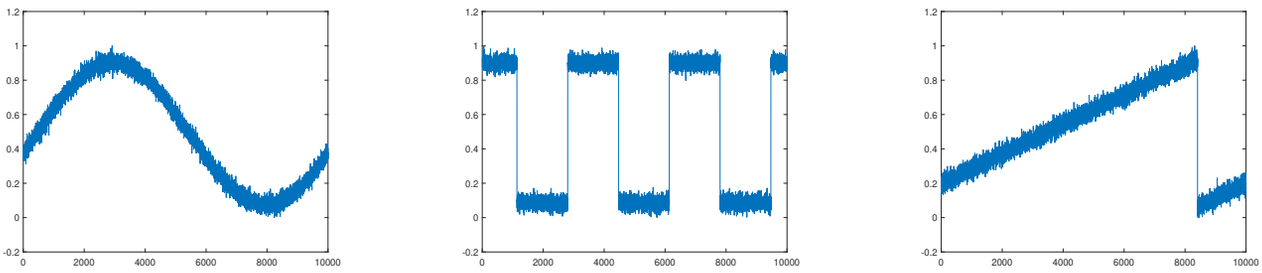
$$\begin{aligned} y(Lr_j) = \sum_{n=1}^{N/4} [&(R_{4n-3} - I_{4n-3}) \frac{\partial R_{4n-3}}{\partial Lr_j} + (R_{4n-2} - I_{4n-2}) \frac{\partial R_{4n-2}}{\partial Lr_j} \\ &+ (R_{4n-1} - I_{4n-1}) \frac{\partial R_{4n-1}}{\partial Lr_j} + (R_{4n} - I_{4n}) \frac{\partial R_{4n}}{\partial Lr_j}] \end{aligned} \tag{5.22}$$

Once we know we can go from 2 to 4 coefficients, we can also infer that using the same logic we can derive the cost function to all even number coefficients. It is expected that the cost functions for odd-numbered coefficients can also be derived using the same logic.

## 5.4 Training Datasets

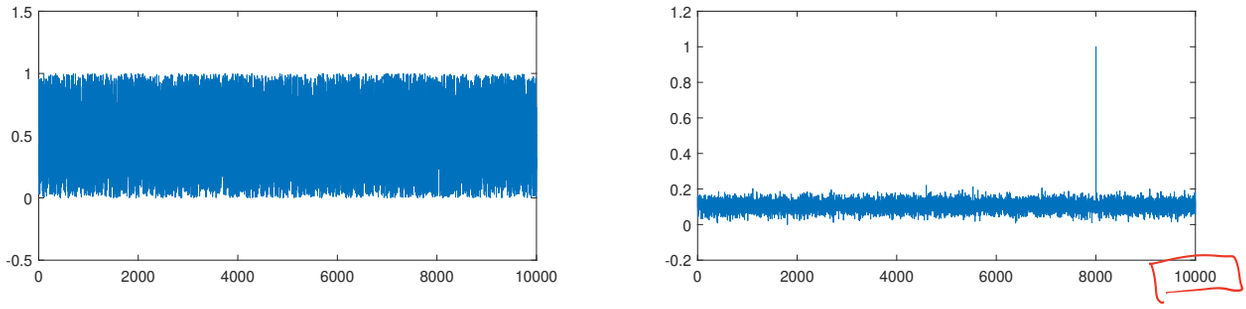
Intuitively, it might be a good idea to train the network using strips obtained from real images. However, obtaining datasets will be more difficult, and in practice networks trained from image strips did not perform as well as networks that were trained from synthetic data.

To train this 1D cost function, we must come up with 1D datasets. Signals that mimic components of natural images have been fabricated. These include sine waves in Figure 15a, square waves in Figure 15b, sawtooth waves in Figure 15c, pulse signals in Figure 16b and random noise in Figure 16a. To allow for better generalisation in the network across different signal types, gaussian noise at an SNR of 30 has been added to the signals [74]. The signals have been normalised to be between 0 and 1, with their frequency and time-shift randomised.



(a) Sinusoidal wave (b) Square wave (c) Sawtooth Wave

Figure 15: Sample signals with added gaussian noise at SNR = 30.



(a) Noise Signal (b) Pulse Signal

Figure 16: Sample signals with added gaussian noise at SNR = 30.

## 5.5 Prototype Implementation

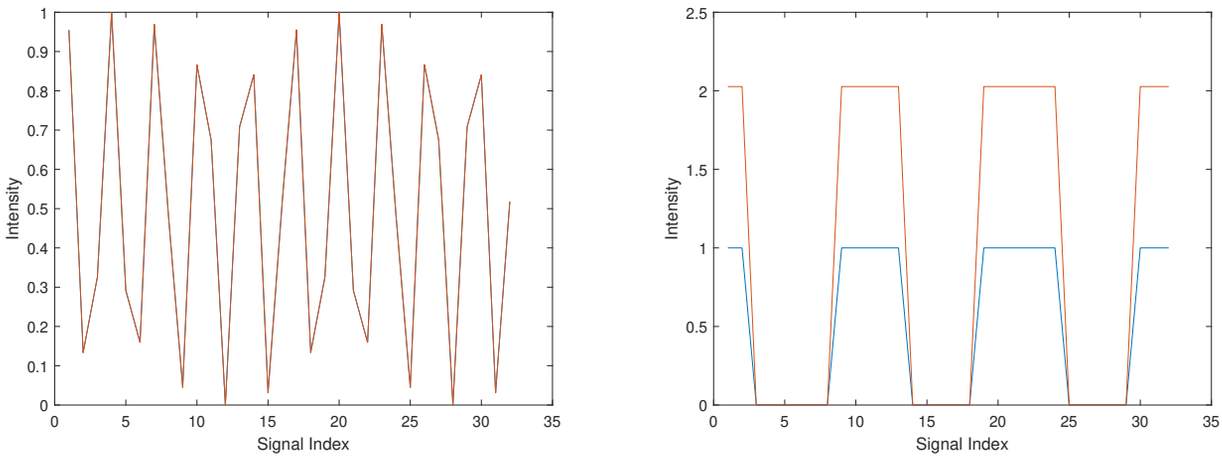
Initially, an unofficial MATLAB neural network script by Langelaar [75] was modified to fit our cost function. This was done because of the flexibility it offers in defining a new cost function, a desirable property for prototyping. However, only the ELU activation function was available; techniques such

as variable learning rate and regularisation were not available. The architecture of the network used to train a 2 coefficient filter set is shown in Table 4. This architecture was empirically found to feature the least number of neurons, while still being able to reach convergence.

| Number | Layer Type | Activations | Weights | Bias |
|---|---|---|---|---|
| 0 | Sequence Input | 32 | - | - |
| 1 | Fully Connected | 5 | 5×32 | 5×1 |
| 2 | ELU Activation | 5 | - | - |
| 3 | Fully Connected | 5 | 5×5 | 5×1 |
| 4 | ELU Activation | 5 | - | - |
| 5 | Regression Output | 4 | - | - |

Table 4: Architecture of network for 2 coefficient case

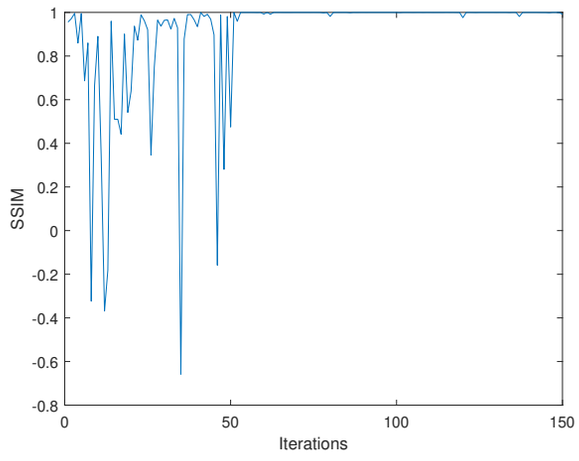
The network was able to converge for sinusoidal signals in the 2 coefficient case, shown in Figure 17a, giving some affirmation that the cost function can demonstrate learning. However, the network failed to converge for other types of signals, such as square waves in Figure 20b and for the 4 coefficient case.



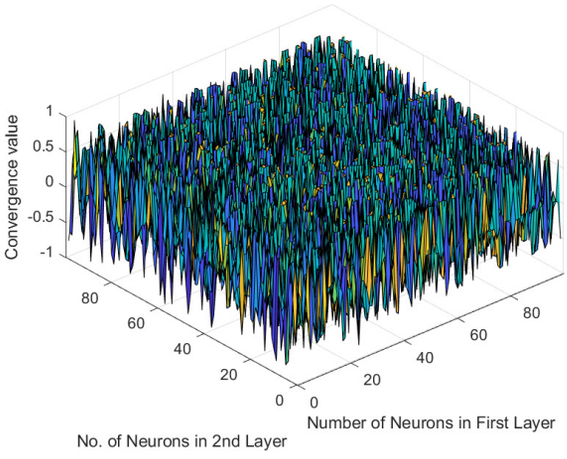

(a) Reconstruction of sinusoidal signal

(b) Square wave

Figure 17: Reconstruction performance for different signal types. Blue and orange curves represent the original and reconstructed signals respectively.




(a) Sinusoidal wave




(b) Hyperparameter Search

Figure 18: (a) Shows the learning performance of the 2 coeff. network when trained with sinusoidal signals exclusively. (b) Shows a surface obtained from a hyperparameter search based on training with sinusoidal signals. The Z axis indicates convergence: A value close to 1 indicates successful convergence. The x and y axes represent the no. of neurons in the first and second layer respectively.

Langelaar's code not robust

Looking at the learning progress shown in Figure 18a, it is clear that the network lacks stability. After doing a hyperparameter search, it's results shown in Figure 18b, it is also revealed that the network topology does not seem to matter for convergence. Because of this, we simply cannot trust this implementation. There is no guarantee that the neural network is implemented robustly.

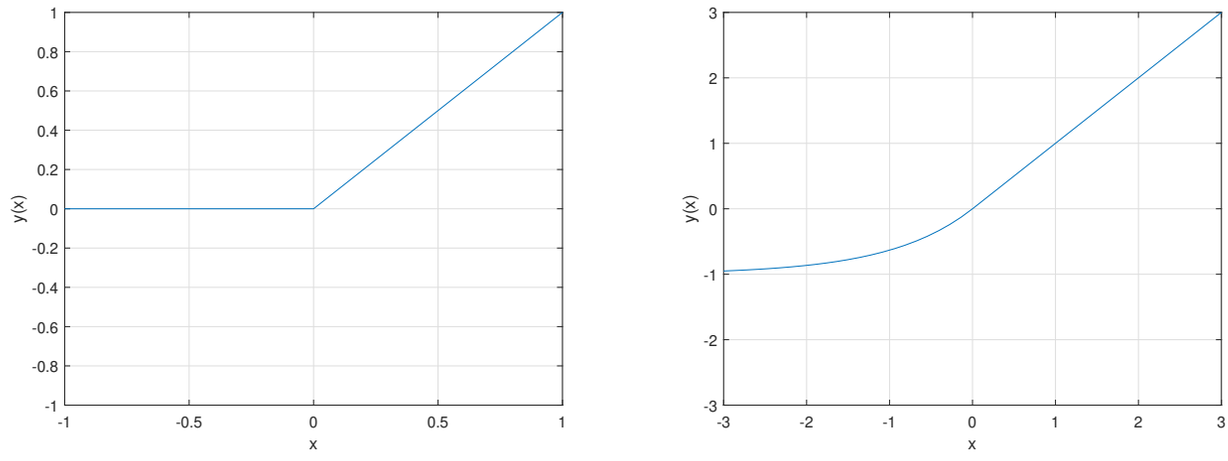
## 5.6 Network Architecture and Parameters

To overcome the limitations of the prototype implementation, the MATLAB Deep Learning Toolbox [76] was chosen to carry out further implementations. As the bulk of the project so far has been conducted in MATLAB, this serves to keep all of the software under the same roof.

A custom regression output layer was built, to encapsulate the cost function derived in Section 5.3. Layer validity tests [77], which ensure correct syntax, dimensional and categorical consistency across outputs and training data, were carried out and passed.

### 5.6.1 Activation Functions

Activation functions such as ReLU and ELU have been utilised, their plots shown in Figures 19a and 19b respectively. Specifically, ELU was chosen as the final activation function to allow for negative activations in the output layer. Negative activations are required because wavelet coefficients are often times negative. ELU is not perfect, because it is bounded at $-1$, which might limit its ability to output true wavelet coefficients, who are not bounded at $-1$. Nonetheless, it is the best option amongst the activation functions at our disposal in the Matlab Deep Learning Toolbox.




(a) ReLU Activation Function

(b) ELU Activation Function

Figure 19: Different activation functions.

Apart from ELU, tanh was also a candidate for introducing negative activations in the output layer, as it also allowed negative activations. However, introducing tanh as a final activation function prevented convergence in practice; this is probably due to it being bounded at +1 and -1. For other layers, ReLU was chosen because of its proven success and reliability in the literature [78].

### 5.6.2 Specialised layers & mini-batches

Specialised layers such as batch normalisation [41] and max pooling [79] were considered, but they were not used as they are only designed for 2D and 3D arrays in the Matlab Deep Learning Toolbox, while our network is designed for analysing 1D signals. Dropout layers were trialed to help with regularisation [42], however in practice they seemed to prevent convergence.

For each gradient descent step, a mini-batch of 64 was used. This means instead of doing gradient descent every time a signal is passed through, it is done once every 64 times. This has been shown to drastically decrease computational cost [80]. Multiple epochs were passed through to take advantage of stability gained from learn rate decay [81]. For this reason, each trial consisted of passing 2 epochs of 100 iterations (1 iteration = processing 1 mini-batch of 64 signals) through the network. For our custom regression layer, it was not clear how validation datasets were partitioned and called in the MATLAB Deep Learning Toolbox; an official validation pipeline could not be built. Validation was instead done on a custom script.

### 5.6.3 Optimisation & Hardware

Stochastic Gradient Descent with Momentum (SGDM) was used as the optimisation solver, because of its speed, reliability and distaste for poor local minima, when compared to other gradient descent

methods [82]. The following parameters are obtained empirically, and seemed to provide the best reliability and route to convergence. A momentum of 0.9 was used, while the initial learn rate is 0.03. The learn rate drops by a factor of 0.2 after an epoch is passed through. The gradient is clipped at 1 via L2 norm gradient thresholding. In an attempt to prevent overfitting, an L2 regularisation factor of 0.1 was picked.

Due to signals being short and confined to 1D and the networks being relatively small, training required little computing power. Training was performed on a single Intel Core i7-7700HQ CPU @ 2.80GHz, 4 Cores, 8 Logical Processors.

### 5.6.4 Architecture

| Number | Layer Type | Activations | Weights | Bias |
|---|---|---|---|---|
| 0 | Sequence Input | 32 | - | - |
| 1 | Fully Connected | 5 | 5×32 | 5×1 |
| 2 | ReLU Activation | 5 | - | - |
| 3 | Fully Connected | 5 | 5×5 | 5×1 |
| 4 | ReLU Activation | 5 | - | - |
| 5 | Fully Connected | 4 | 4×5 | 4×1 |
| 6 | ELU Activation | 4 | - | - |
| 7 | Reconstruction Regression Output | 4 | - | - |

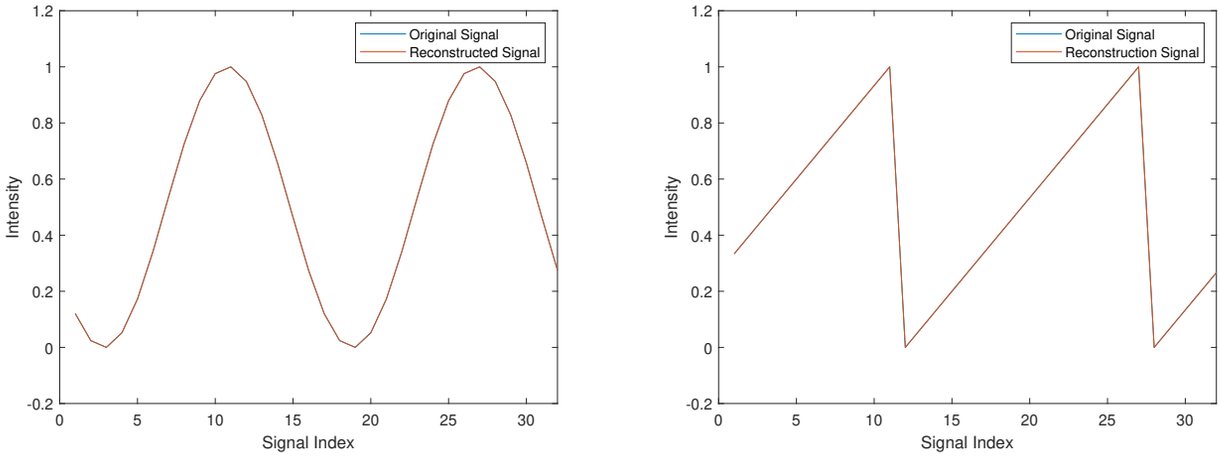
Table 5: Architecture of network for 2 coefficient case

The number of neurons in each layer and the number of hidden layers are arrived at empirically. An initial trial of 2 hidden layers has demonstrated convergence for the 2 coefficient cost function. This is repeated with a decrease in the number of neurons, until the network ceased to converge. The architecture shown in Table 5 is the simplest architecture that achieves convergence for the 2 coefficient cost function. For the 4 coefficient case, the number of neurons are simply doubled w.r.t. the 2 coefficient architecture to take into account for added complexity, shown in Table 6. This is also able to demonstrate convergence. Arguments for choosing the ELU and ReLU activation functions at their respective positions are presented in Section 5.6.1.

| Number | Layer Type | Activations | Weights | Bias |
|---|---|---|---|---|
| 0 | Sequence Input | 32 | - | - |
| 1 | Fully Connected | 10 | 10×32 | 10×1 |
| 2 | ReLU Activation | 10 | - | - |
| 3 | Fully Connected | 10 | 10×10 | 10×1 |
| 4 | ReLU Activation | 10 | - | - |
| 5 | Fully Connected | 8 | 8×10 | 8×1 |
| 6 | ELU Activation | 8 | - | - |
| 7 | Reconstruction Regression Output | 8 | - | - |

Table 6: Architecture of network for 4 coefficient case

## 5.7 Results




(a) Sinusoidal wave

(b) Square wave

Figure 20: (a) and (b) show original signals (in blue) superimposed on reconstruction signals (in orange).

Both networks are able to reconstruct all signal types. Figures 20a and 20b show the 4 coefficient network in action. It is clear that the MATLAB Deep Learning implementation has far surpassed the prototype implementation, by demonstrating convergence for both networks across all signal types.

Instead of using the SSIM, the Mean Squared Error (MSE) (5.23) was used to gauge reconstruction performance. This is because results in this section consistently achieve SSIM values near 1, and at that stage using the MSE would allow us to differentiate performance better.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(I_i - \hat{I}_i) \tag{5.23}$$

Where $n$ is the length of the signal, $I_i$ denotes the $i$th element of the original signal, and $\hat{I}_i$ denotes the $i$th element of the reconstructed signal.

Although the networks still suffer from instability, we can afford to repeat training until convergence as the computing cost is low. Figure 21 shows the learning performance of a 4 coefficient network. Its corresponding validation performance is shown in Figure 22; it is clear that this network consistently produces reconstructions with MSE in the magnitude between $10^{-4}$ and $10^{-5}$.

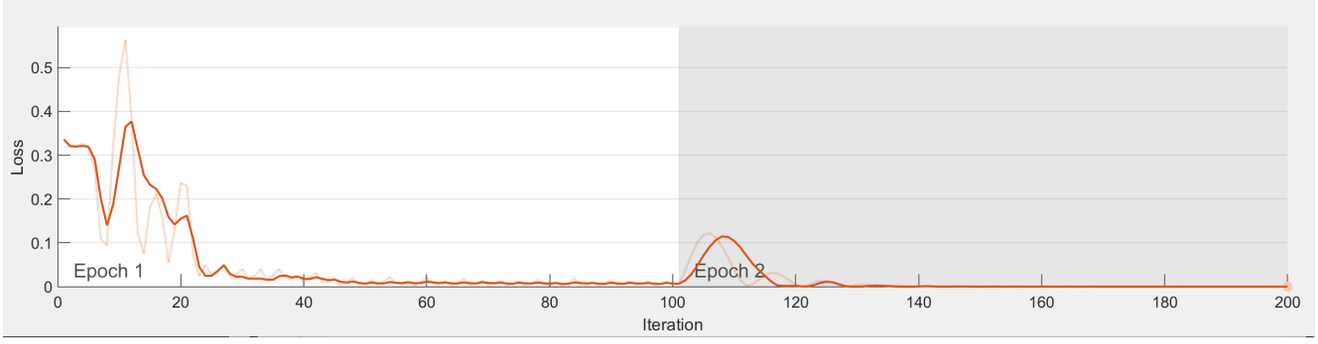



Figure 21: Learning performance for 4 coeff network. The loss is defined as MSE between original and reconstructed signal (5.23). This trial has achieved a final mini-batch loss of MSE = 7.0e-06.

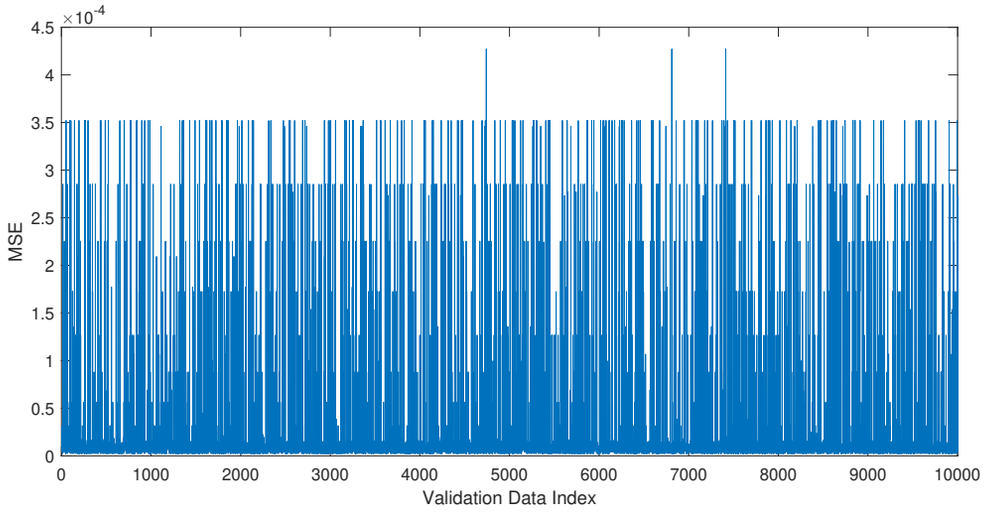



Figure 22: Validation for 4 coeff network.

### 5.7.1 Translation to Images

Somewhat surprisingly, wavelets generated to reconstruct 1d signals were also exceptionally good at reconstructing 2d signals. This could be explained by the fact that the 2D DWT is but a series of 1D DWTs. Figure 23c showcases the reconstruction performance of a set of 4 coefficient filters that we have learned. Although the example used here is a natural image, the same filters are also able to achieve the same level of reconstruction with graphical and text images.

Another interesting property of filter coefficients generated is: they do not seem to be signal specific. For example, filters generated to reconstruct sinusoidal signals can also reconstruct sawtooth signals at the same degree of accuracy.

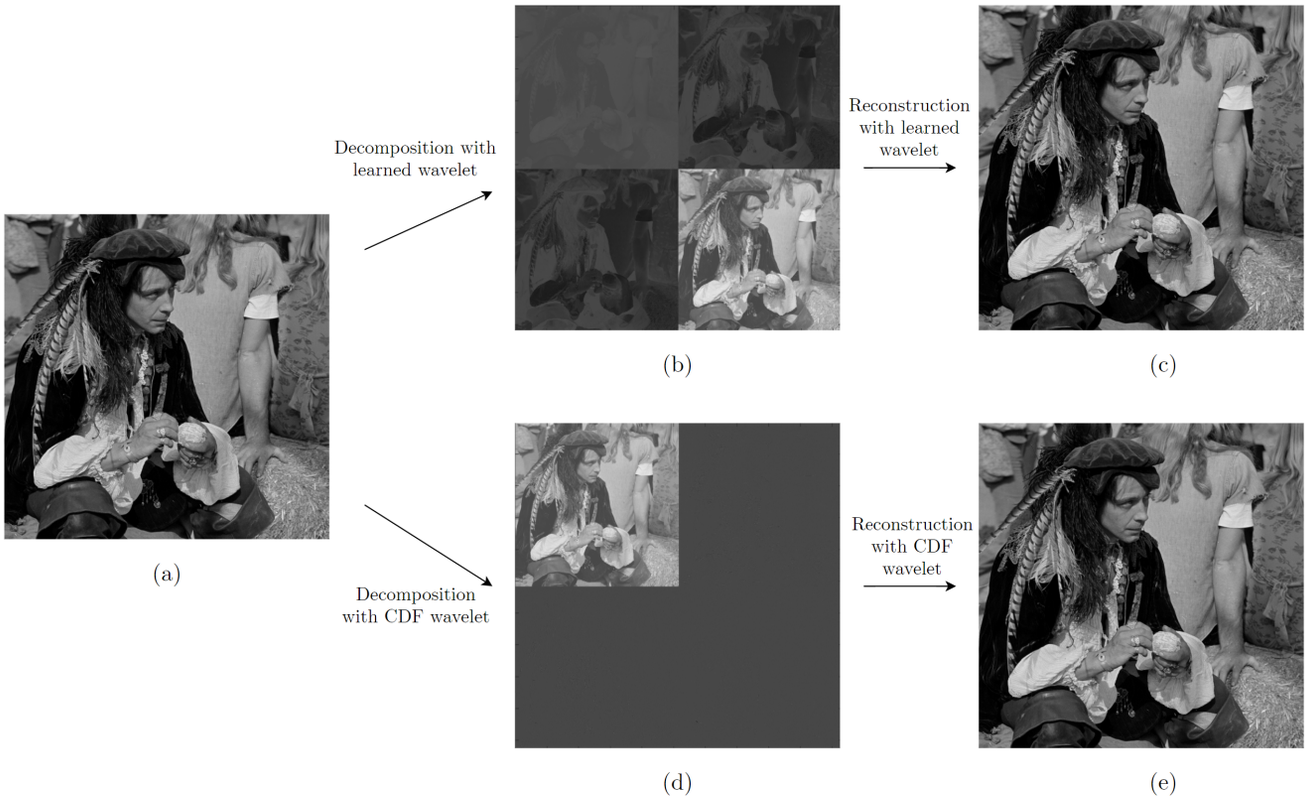



Figure 23: The original image is shown in (a). (b) and (d) are obtained by performing 2D, 1 level DWT with a 4 coeff. learned wavelet and CDF wavelet respectively. (c) and (e) are obtained by reconstruction with their respective wavelets. Both wavelets were able to achieve SSIM = 1.000.

It is also surprising to see that the network has achieved this without cheating, *i.e.* without outputting the Haar identity wavelet. However, it is evident when we compare Figure 23b and Figure 23d, that the learned wavelet is not able to produce a sparse latent representation in comparison to CDF. This is expected, as the cost function has not been told to favour sparseness.

To be certain that wavelet filters generated are not copies of existing wavelet filters, additional checks were carried out. Each generated filter set was compared with all existing wavelet filter sets of the same length via computing the MSE. An arbitrary MSE threshold of 0.1 was picked. If the MSE dropped below 0.1 for all 4 filter pairs in the set, it might mean an existing filter set had been generated, but this had not occurred. The filter coefficients used to obtain Figure 23b from Figure 23a are shown in Table 7.

| $k$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **Ld** | -0.4987862 | -0.4047294 | 0.4578148 | 0.4578148 |
| **Hd** | 0.8799217 | 0.7140718 | 0.9033192 | -0.3255504 |
| **Lr** | -0.8799217 | 0.7140718 | -0.9033192 | -0.3255504 |
| **Hr** | -0.4987862 | 0.4047294 | 0.4578148 | 0.1649629 |

Table 7: Table showing the numerical values of wavelet coefficients. $k$ represents the index of a coefficient.

# Section 6

# Compression

Compression can be interpreted as a reduction in entropy in the latent space [83]. Reducing entropy is less straightforward than ensuring reconstruction, as there isn't a clear ground truth for the network to learn from. Although entropy reduction has not been fully tackled in this project, this section serves to introduce some methods that were attempted. Hopefully, this material can serve as inspiration for any further work.

## 6.1 Cost Function

The cost function for entropy reduction will be similair to the cost function for reconstruction. The major difference being our operating in the latent space $A_{n/2}$ and $D_{n/2}$, circled in red in Figure 24, instead of the reconstruction space. Instead of taking the displacement between the reconstructed pixel and the original pixel (6.1), we will take the displacement between pixels in the latent space and arbitrary arrays $Da$ and $Aa$ that we define (6.2).

$$y(Ld_i) = \sum_{n=1}^{N/2} [(R_{2n-1} - I_{2n-1}) \frac{\partial R_{2n-1}}{\partial Ld_i} + (R_{2n} - I_{2n}) \frac{\partial R_{2n}}{\partial Ld_i}] \tag{6.1}$$

$$y(Ld_i) = \sum_{n=1}^{N/2} [(D_{n/2} - Da_{n/2}) \frac{\partial D_{n/2}}{\partial Ld_i} + (A_{n/2} - Aa_{n/2}) \frac{\partial A_{n/2}}{\partial Ld_i}] \tag{6.2}$$

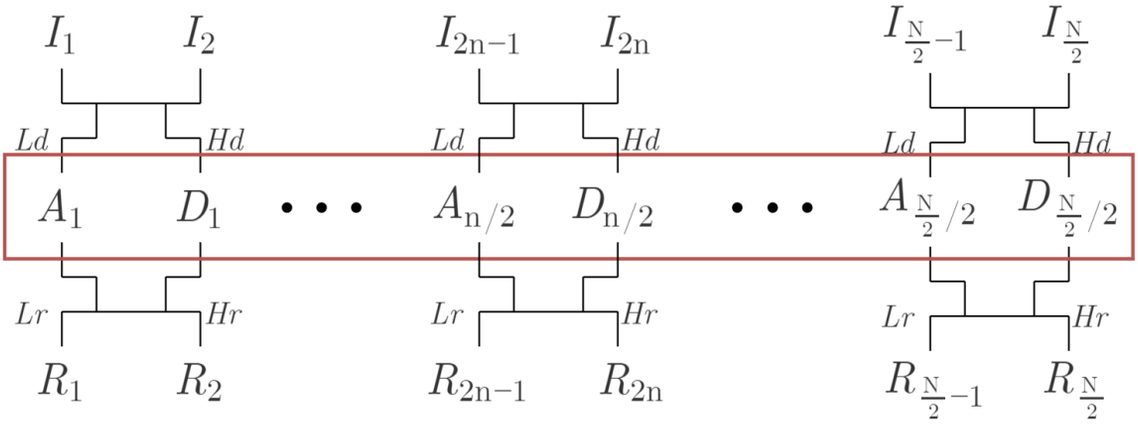
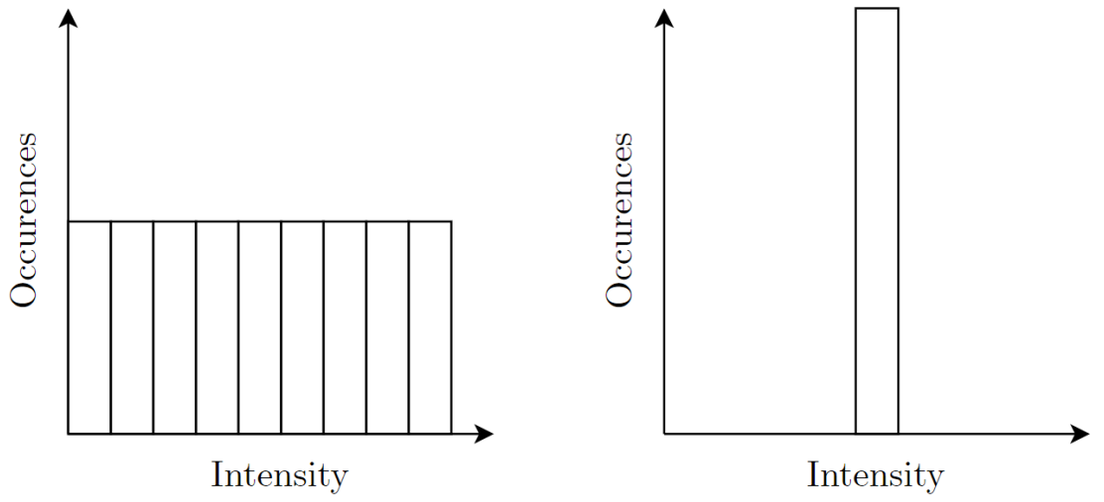



Figure 24: Neural network

Compression could be achieved by using the entropy as a measure of performance. Consider an image histogram of the latent space representation: the maximum entropy scenario is a histogram with equal probabilities, shown in Figure 25a. The minimum entropy scenario is a histogram with a single peak, shown in Figure 25b.

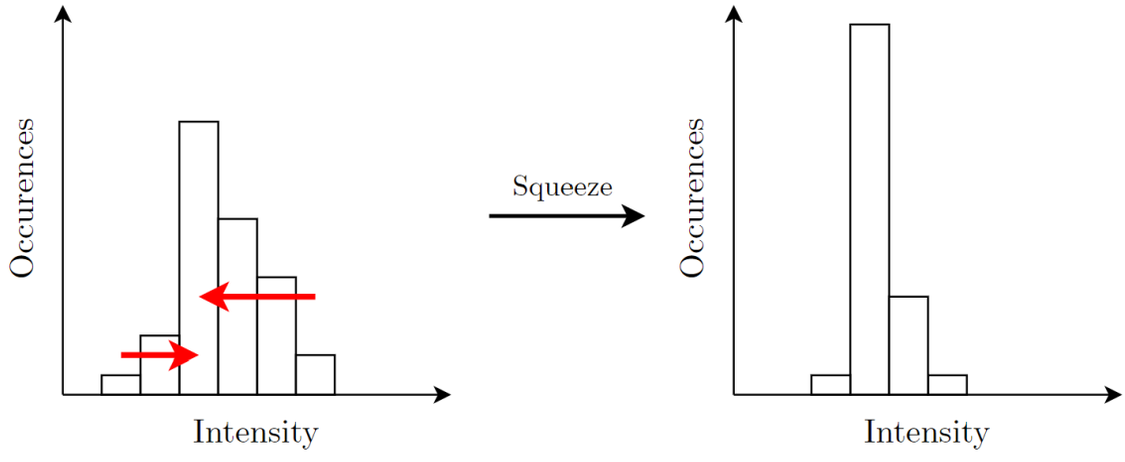



(a) Histogram for maximum entropy (b) Histogram for minimum entropy

Figure 25: Two extremes in entropy. (a) would yield the highest possible entropy, while (b) would yield an entropy of 0.

A latent space representation of a real signal will be somewhere between the best and worst case, shown in Figure 26a. One possible way of lowering the entropy is to set the pixels in the arbitrary arrays $Aa_n/2$ and $Da_n/2$ as the mode of the histograms $\hat{A}_{n/2}$ and $\hat{D}_{n/2}$ (6.1). The aim is to 'squeeze' the histogram; lowering the entropy while retaining useful information.

$$Aa_{n/2} = \hat{A}_{n/2} \qquad Da_{n/2} = \hat{D}_{n/2} \tag{6.3}$$




(a) Histogram for real signal and red arrows indicating squeeze direction

(b) Histogram for squeezed signal

Figure 26: (a) Simulates the histogram of a latent signal. (b) Shows what this cost function is trying to achieve: A squeeze effect.

### 6.1.1 Training

The training dataset and network architecture will be identical to the ones used for reconstruction, shown in Section 5.6.4. This is because the cost function employed for entropy reduction has almost identical characteristics to the cost function for reconstruction.

## 6.2 Results

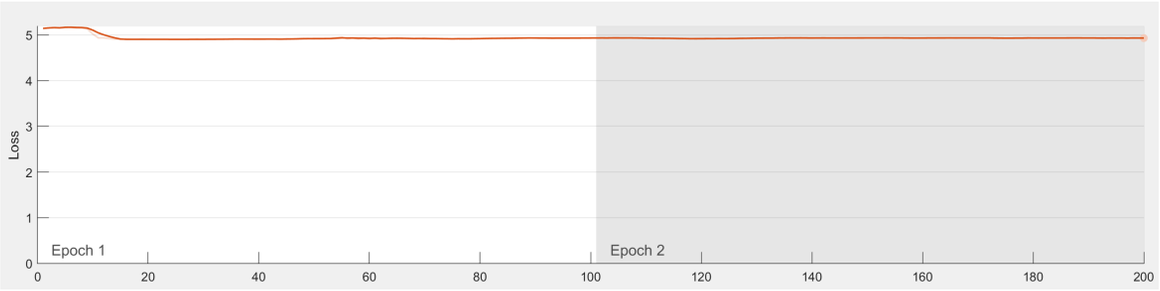
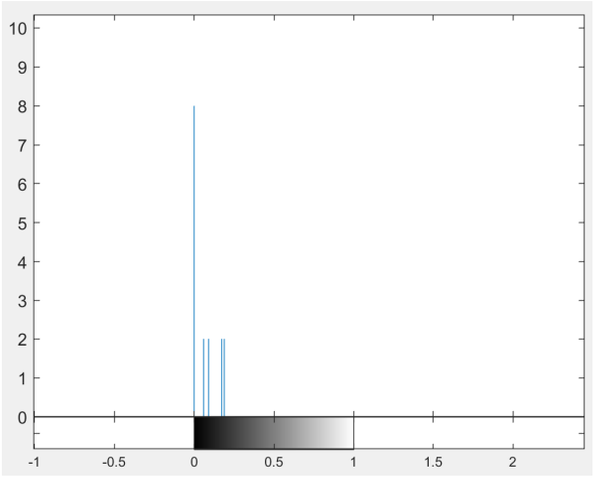
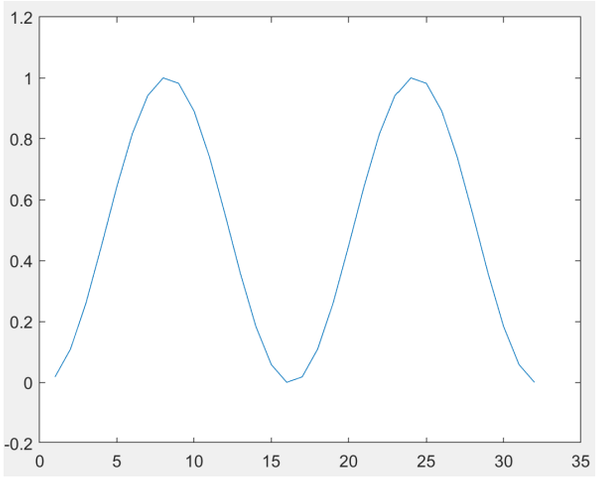



Figure 27: Learning performance of 2 coeff. network for entropy reduction. The loss is defined as the sum of the entropies of the approximation $A_n$ and detail coefficients $D_n$. Evidently, this cost function has failed to demonstrate any signs of learning.

Figure 27 shows the learning performance of this cost function. It does not demonstrate any sign of learning. Wavelets generated by this network were not able to meaningfully diminish entropy. This might be an indicator that the cost function has not been successful at fulfilling its goal. It is also possible that the network architecture has not been configured optimally to recognise the abstractions within the cost function. It is also not clear whether it is possible to predict wavelet filters that favour a uni-modal latent distribution at all.




(a) Histogram of approximation coefficients of sinusoidal signal in (b)



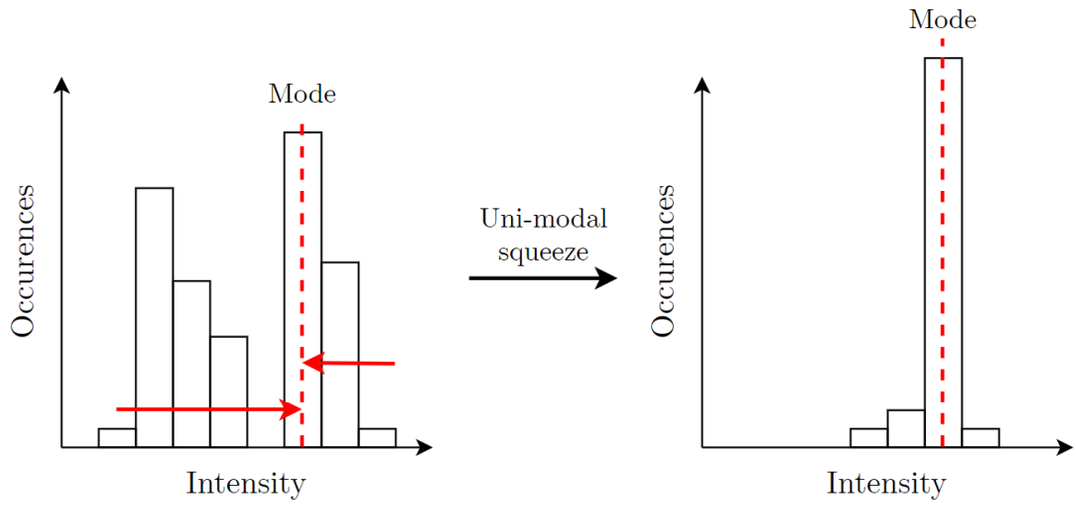

(b) Original sinusoidal signal

Figure 28: (a) Shows the histogram of approximation coefficients obtained from (b), using wavelet coefficients suggested by an entropy reduction network.

The histogram of detail coefficients from a sinusoidal signal is shown in Figure 28a. We can see that the histogram is too sparse for any meaningful 'squeezing' to take place. Another problem is the possibility of multi-modal distributions; in this situation our uni-modal model would break down.

### 6.2.1 Improving the Cost Function

One of the most obvious limitations of the cost function are multi-modal latent distributions, an example shown in Figure 29. In this case, simply favouring pixels to move toward the mode of the histograms would lead to information loss in the other peaks.

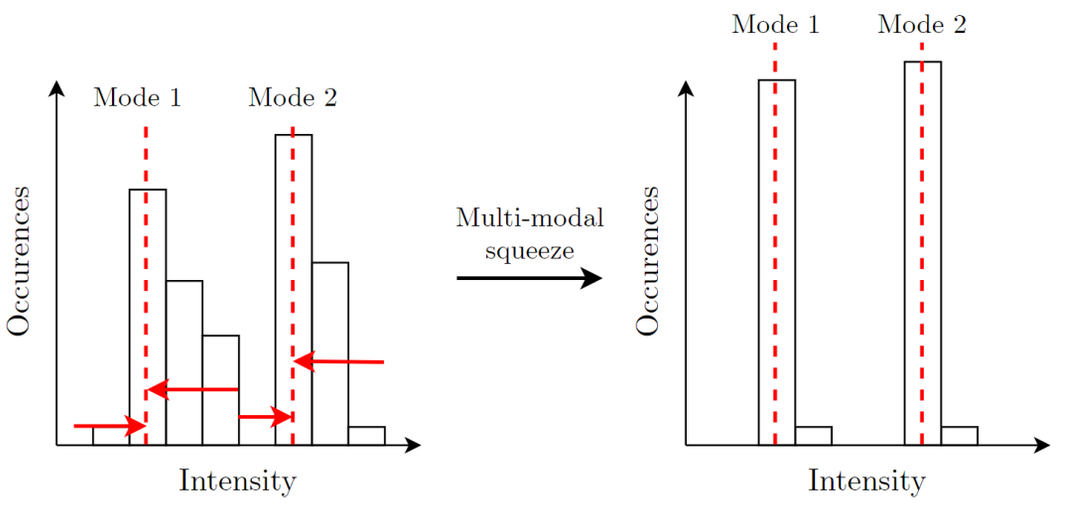



(a) Histogram for multi-modal latent signal with dashed line indicating mode of the histogram and arrows indicating squeeze direction

(b) Histogram for squeezed signal

Figure 29: This illustrates when a uni-modal prediction could lead to disastrous effects.

A way to combat this is to use multi-modal detection tools such as neural networks [84] to predict multiple peaks, shown in Figure 30.

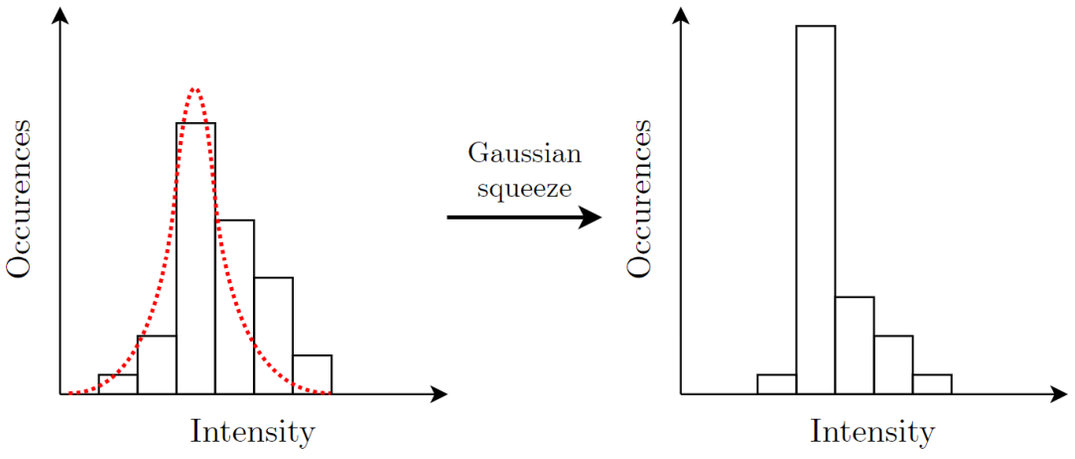



(a) Histogram for multi-modal latent signal with dashed line indicating modes of the histogram and arrows indicating squeeze direction

(b) Histogram for multi-modal squeezed signal

Figure 30: This illustrates how multi-modal detection could overcome problems shown in Figure 29.

Another limitation of this would be its tendency to favour extreme unimodal distributions, shown in Figure 26b. A way to overcome this is to synthesise ground truth arrays that fit a gaussian through the mode of the latent space representation, shown in Figure 31.

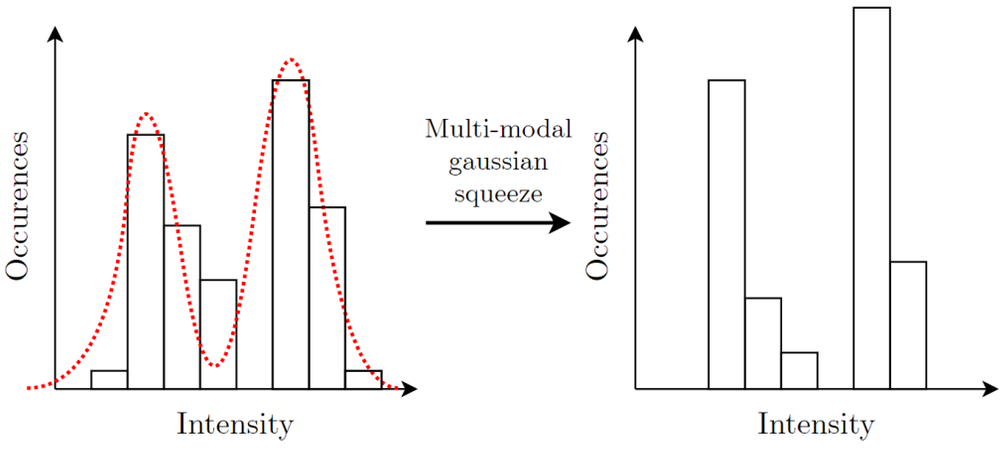



(a) Histogram for uni-modal latent signal with dashed line indicating gaussian fit

(b) Histogram for gaussian squeezed signal

Figure 31: This illustrates how a gaussian fit could overcome the problem of extreme unimodal distributions.

This could be combined with the previous multi-modal improvement, shown in Figure 32, giving us tool more robust for building ground truth arrays based on image histograms of latent representations.




(a) Histogram for multi-modal latent signal with dashed line indicating multi-modal gaussian fit

(b) Histogram for multi-modal gaussian squeezed signal

Figure 32: This combines multimodal prediction and gaussian fitting for an even more robust system.

We have now reached the end for suggestions for the entropy-focused cost function; hopefully this theoretical material can serve as a guide for any future practical implementations. We shall now move on to further suggestions that are outside the scope of our cost function, but nevertheless still relevant to the project.

## 6.3 Further Suggestions

An alternative method would be to use an empty array as ground truth. Although this might encourage the network to 'cheat' by outputting empty filters, this method might hold more value than meets the eye. This is because histograms of latent representation of real 2D images often peak at 0 intensity, shown in Figure 33c. When paired with a functional reconstruction cost function, the phenomena of 'cheating' might go away.


(a) Original Image (b) Vertical Latent Coefficients (c) Histogram of (b)

Figure 33: (b) represents vertical latent coefficients obtained from using CDF to transfrom (a). It should be noted that (b) is not a blank array; closer inspection would reveal sparse components.

### 6.3.1 Combining Cost Functions

Combining the two entropy and reconstruction cost functions would not bear fruit, as only one of them is working properly. The following is meant as a guide to combining the two cost functions in the case of triumph over the entropy problem in the future, and a closer to this project's main body of work.

Under a certain SSIM, information loss would be considered unacceptable. Let's call this limit $S_0$. While SSIM $\leq$ S$_0$, the network should only focus on reconstruction. The entropy cost function should only be allowed to influence backpropagation after SSIM $\geq$ S$_0$. With this principal we can define the piecewise function $\mathbf{p}$ (6.3.1) that takes the reconstruction cost $\mathbf{r}$, entropy cost $\mathbf{e}$ and SSIM as inputs, to output a single vector cost for backpropagation to the network.

$$\mathbf{p}(\mathbf{r}, \mathbf{e}, \text{SSIM}) = \begin{cases} \mathbf{r} & 0 \leq \text{SSIM} \leq S_0 \\ (a\mathbf{r} + b\mathbf{e})/(a+b) & S_0 \leq \text{SSIM} \leq 1 \end{cases} \tag{6.4}$$

Where $a$ and $b$ are arbitrary parameters for controlling the influence of the entropy cost $\mathbf{e}$ and reconstruction cost $\mathbf{r}$. It might be helpful to define them as a function of SSIM.

# Section 7

# Conclusion and Further Work

## 7.1 Conclusion

This project was successful at learning wavelet filter coefficients of length 2 and 4 for near perfect reconstruction of 1D and 2D arrays, with the help of fully connected neural networks. This is perhaps best summarised by Figure 35. This project has demonstrated that it is possible to differentiate parameters such as reconstruction performance with respect to individual wavelet filter coefficients.

(a) Original Image (b) Latent Representation

Figure 34: The fruit of this project's labour: Learning a wavelet that could decompose (a) into (b), and achieve near perfect reconstruction from (b).

However, designing a cost function to learn 2 coefficient wavelet filters for entropy reduction could not be achieved. This could be due to an incorrect cost function, or a faulty neural network architecture. Hopefully, this content can serve as a reasonable starting point for further work in the realm of image compression and beyond.

## 7.2 Further work

Apart from the obvious glaring problem of defining a cost function for entropy reduction, alternative ideas for exploration are presented here.

Firstly, the derivations carried out in this project only cater to even-numbered filter lengths. Existing wavelets often contain odd-numbered filter lengths; for example, the CDF wavelet filters have lengths of 9 [66]. Carrying out derivations for odd-numbered filters could be beneficial.

Secondly, our neural network architectures hold no dimensional invariance. For example, the topology used in Section 5.6.4 can only be used for inputs of length 32. When presented with inputs of other lengths, the network is rendered useless. Perhaps more work in normalisation and feature scaling [85] could be done.

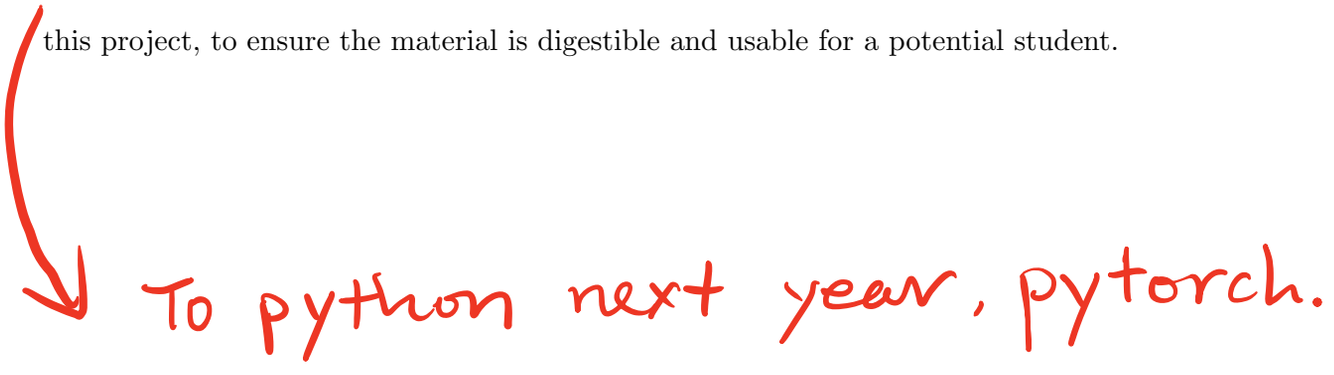
Thirdly, neural network architectures in this project were obtained from trial and error. We cannot determine the optimality of an architecture with this method [86]. In the future, more systematic methods such as genetic algorithms [87] and reinforcement learning [88] could be used to tune neural network architectures. Continuing this work outside of the Matlab Deep Learning Toolbox to more widely adopted machine learning frameworks could be a means of incorporating these methods.

Fourthly, the means to combine cost functions outlined in Section 6.3.1 remain trial and error based. Multi-objective optimisation [89] could be employed instead to obtain Pareto-optimal solutions [90].

Additionally, fabricating composite signals that mimic nature, or simply sampling from the real world, could speed up training and increase accuracy, when compared to elementary sinusoidal and sawtooth signals used in this project.

Furthermore, developing custom dyadic convolutional layers could be a way to introduce spatial awareness into the network. Although convolutional layers were considered for this project, they were rejected due to their inability to process sequential data in the Matlab Deep Learning Toolbox, and their incompatibility with dyadic operations inherent in the DWT.

Lastly, this project might be continued next year. Efforts will be taken to polish code written for this project, to ensure the material is digestible and usable for a potential student.



# References


[1] Cam Cullen. Global internet phenomena report. *Sandvine, Tech. Rep.*, 2018.

[2] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[3] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[4] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

[5] Mark J Shensa et al. The discrete wavelet transform: wedding the a trous and mallat algorithms. *IEEE Transactions on signal processing*, 40(10):2464–2482, 1992.

[6] Michael W Marcellin, Michael J Gormish, Ali Bilgin, and Martin P Boliek. An overview of jpeg-2000. In *Proceedings DCC 2000. Data Compression Conference*, pages 523–541. IEEE, 2000.

[7] Michael J Gormish, Daniel Lee, and Michael W Marcellin. Jpeg 2000: overview, architecture, and applications. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 2, pages 29–32. IEEE, 2000.

[8] Robert R Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6):52–59, 1997.

[9] Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

[10] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010.

[11] J.T. Ko. Learning wavelet filters from fully connected networks for image compression. `https://github.com/justinko9/wavelets4yp`, 2021.

[12] Jason Eppink. A brief history of the gif (so far). *Journal of visual culture*, 13(3):298–306, 2014.

[13] G Parsons and J Rafferty. Tag image file format (tiff)–f profile for facsimile. Technical report, RFC 2306, March, 1998.

[14] Mark R Nelson. Lzw data compression. *Dr. Dobb's Journal*, 14(10):29–36, 1989.

[15] Stuart C Hinds, James L Fisher, and Donald P D'Amato. A document skew detection method using run-length encoding and the hough transform. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume 1, pages 464–468. IEEE, 1990.

[16] David Taubman. High performance scalable image compression with ebcot. *IEEE Transactions on image processing*, 9(7):1158–1170, 2000.

[17] Li Lian and Wei Shilei. Webp: A new image compression format based on vp8 encoding. *Microcontrollers & Embedded Systems*, 3, 2012.

[18] Thomas Boutell and T Lane. Png (portable network graphics) specification version 1.0. *Network Working Group*, pages 1–102, 1997.

[19] David Taubman. *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer, 2002.

[20] Allan G Weber. The usc-sipi image database version 5. *USC-SIPI Report*, 315(1), 1997.

[21] Xnconvert website. `https://www.xnview.com/en/xnconvert/`. Accessed: 2021-05-16.

[22] Joel Askelöf, Mathias Larsson Carlander, and Charilaos Christopoulos. Region of interest coding in jpeg 2000. *Signal Processing: Image Communication*, 17(1):105–111, 2002.

[23] A Robert Calderbank, Ingrid Daubechies, Wim Sweldens, and Boon-Lock Yeo. Lossless image compression using integer to integer wavelet transforms. In *Proceedings of International Conference on Image Processing*, volume 1, pages 596–599. IEEE, 1997.

[24] Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[25] Kai Liu, Yu Zhou, Yun Song Li, and Jian Feng Ma. A high performance mq encoder architecture in jpeg2000. *Integration*, 43(3):305–317, 2010.

[26] Emmanual Bacry, Stephane Mallat, and George Papanicolaou. A wavelet based space-time adaptive numerical method for partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 26(7):793–834, 1992.

[27] Avijit Chakraborty and David Okaya. Frequency-time decomposition of seismic data using wavelet-based methods. *Geophysics*, 60(6):1906–1916, 1995.

[28] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[29] Gilbert Strang and Truong Nguyen. *Wavelets and filter banks*. SIAM, 1996.

[30] Michael Unser and Thierry Blu. Mathematical properties of the jpeg2000 wavelet filters. *IEEE transactions on image processing*, 12(9):1080–1090, 2003.

[31] M. Islam, S. Ahemmed, and S. Rahman. Comparison of wavelet approximation order in different smoothness spaces. *International Journal of Mathematics and Mathematical Sciences*, 2006, 07 2006.

[32] RA Zalik. Riesz bases and multiresolution analyses. *Applied and Computational Harmonic Analysis*, 7(3):315–331, 1999.

[33] George C Kyriazis. Wavelet coefficients measuring smoothness inhp (open face rd). *Applied and Computational Harmonic Analysis*, 3(2):100–119, 1996.

[34] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[35] Stephen Ed Grossberg. *Neural networks and natural intelligence*. The MIT press, 1988.

[36] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[37] Christopher M Bishop. Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222, 1994.

[38] Kc Morris, Craig Schlenoff, and Vijay Srinivasan. A remarkable resurgence of artificial intelligence and its impact on automation and autonomy. *IEEE Transactions on Automation Science and Engineering*, 14:407–409, 04 2017.

[39] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee, 2017.

[40] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[41] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[43] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018.

[44] David S Taubman and Robert Prandolini. Architecture, philosophy, and performance of jpip: Internet protocol standard for jpeg 2000. In *Visual Communications and Image Processing 2003*, volume 5150, pages 791–805. International Society for Optics and Photonics, 2003.

[45] Frederic Dufaux, Susie J Wee, John G Apostolopoulos, and Touradj Ebrahimi. Jpsec for secure imaging in jpeg 2000. In *Applications of Digital Image Processing XXVII*, volume 5558, pages 319–330. International Society for Optics and Photonics, 2004.

[46] Frederic Dufaux and Didier Nicholson. Jpwl: Jpeg 2000 for wireless applications. In *Applications of Digital Image Processing XXVII*, volume 5558, pages 309–318. International Society for Optics and Photonics, 2004.

[47] Michael D Adams and Faouzi Kossentini. Jasper: A software-based jpeg-2000 codec implementation. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 2, pages 53–56. IEEE, 2000.

[48] Openjpeg website. `https://www.openjpeg.org/`. Accessed: 2021-05-16.

[49] Quanping Huang, Rongzheng Zhou, and Zhiliang Hong. Low memory and low complexity vlsi implementation of jpeg2000 codec. *IEEE transactions on Consumer Electronics*, 50(2):638–646, 2004.

[50] Pooja Rawat, Arti Rawat, and Swati Chamoli. Analysis and comparison of ezw, spiht and ebcot coding schemes with reduced execution time. *International Journal of Computer Applications*, 130:24–29, 11 2015.

[51] Chung-Jr Lian, Kuan-Fu Chen, Hong-Hui Chen, and Liang-Gee Chen. Lifting based discrete wavelet transform architecture for jpeg2000. In *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No. 01CH37196)*, volume 2, pages 445–448. IEEE, 2001.

[52] John D Villasenor, Benjamin Belzer, and Judy Liao. Wavelet filter evaluation for image compression. *IEEE Transactions on image processing*, 4(8):1053–1060, 1995.

[53] Henning Thielemann. Optimally matched wavelets. In *PAMM: Proceedings in Applied Mathematics and Mechanics*, volume 4, pages 586–587. Wiley Online Library, 2004.

[54] J.O. Chapa and R.M. Rao. Algorithms for designing wavelets to match a specified signal. *IEEE Transactions on Signal Processing*, 48(12):3395–3406, 2000.

[55] Bruno A Olshausen, Phil Sallee, Michael S Lewicki, et al. Learning sparse image codes using a wavelet pyramid architecture. *Advances in neural information processing systems*, pages 887–893, 2001.

[56] Mark A Goldberg, Misha Pivovarov, William W Mayo-Smith, Meenakshi P Bhalla, Johan G Blickman,

Robert T Bramson, GW Boland, HJ Llewellyn, and Elkan Halpern. Application of wavelet compression to digitized radiographs. *AJR. American journal of roentgenology*, 163(2):463–468, 1994.

[57] Zhitao Lu, Dong Youn Kim, and William A Pearlman. Wavelet compression of ecg signals by the set partitioning in hierarchical trees algorithm. *IEEE transactions on Biomedical Engineering*, 47(7):849–856, 2000.

[58] Philippe Beaudoin, Pierre Poulin, and Michiel van de Panne. Adapting wavelet compression to human motion capture clips. In *Proceedings of Graphics Interface 2007*, pages 313–318, 2007.

[59] Jpeg 89th meeting. `https://jpeg.org/items/20201014_press.html`. Accessed: 2021-05-16.

[60] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5306–5314, 2017.

[61] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[62] Pinar Akyazi and Touradj Ebrahimi. Learning-based image compression using convolutional autoencoder and wavelet decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, number CONF, 2019.

[63] Han Qiu, Qinkai Zheng, Gerard Memmi, Jialiang Lu, Meikang Qiu, and Bhavani Thuraisingham. Deep residual learning-based enhanced jpeg compression in the internet of things. *IEEE Transactions on Industrial Informatics*, 17(3):2124–2133, 2021.

[64] Matlab wavelet toolbox. `https://uk.mathworks.com/products/wavelet.html`. Accessed: 2021-05-16.

[65] Cédric Vonesch, Thierry Blu, and Michael Unser. Generalized daubechies wavelet families. *IEEE Transactions on Signal Processing*, 55(9):4415–4429, 2007.

[66] Albert Cohen. Biorthogonal wavelets. *Wavelets: A Tutorial in Theory and Applications*, 2:123–152, 1992.

[67] Abhinav Dixit and Swatilekha Majumdar. Comparative analysis of coiflet and daubechies wavelets using global threshold for image denoising. *International Journal of Advances in Engineering & Technology*, 6(5):2247, 2013.

[68] Rajeev Singh, Ramon E Vasquez, and Reena Singh. Comparison of daubechies, coiflet, and symlet for edge detection. In *Visual Information Processing VI*, volume 3074, pages 151–159. International Society for Optics and Photonics, 1997.

[69] Robert Szewczyk, Kamil Grabowski, Malgorzata Napieralska, Wojciech Sankowski, Mariusz Zubert, and Andrzej Napieralski. A reliable iris recognition algorithm based on reverse biorthogonal wavelet transform. *Pattern Recognition Letters*, 33(8):1019–1026, 2012.

[70] S Olhede and AT Walden. The hilbert spectrum via wavelet projections. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 460(2044):955–975, 2004.

[71] Christopher E Heil and David F Walnut. Continuous and discrete wavelet transforms. *SIAM review*, 31(4):628–666, 1989.

[72] B Selby. The index of dispersion as a test statistic. *Biometrika*, 52(3/4):627–629, 1965.

[73] Matlab dwt function. `https://uk.mathworks.com/help/wavelet/ref/dwt.html`. Accessed: 2021-05-16.

[74] Lasse Holmstrom, Petri Koistinen, et al. Using additive noise in back-propagation training. *IEEE transactions on neural networks*, 3(1):24–38, 1992.

[75] Matlab mnist neural network training and testing. `https://uk.mathworks.com/matlabcentral/fileexchange/73010-mnist-neural-network-training-and-testing`. Accessed: 2021-05-16.

[76] Phil Kim. Matlab deep learning. *With machine learning, neural networks and artificial intelligence*, 130:21, 2017.

[77] Matlab mnist neural network training and testing. `https://uk.mathworks.com/help/deeplearning/ref/checklayer.html`. Accessed: 2021-05-16.

[78] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[79] Jawad Nagi, Frederick Ducatelle, Gianni A Di Caro, Dan Cireşan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jürgen Schmidhuber, and Luca Maria Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 342–347. IEEE, 2011.

[80] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8), 2012.

[81] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

[82] Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

[83] Donald S Ornstein and Benjamin Weiss. Entropy and data compression schemes. *IEEE Transactions on Information Theory*, 39(1):78–83, 1993.

[84] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.

[85] Alice Zheng and Amanda Casari. *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.", 2018.

[86] D Stathakis. How many hidden layers and nodes? *International Journal of Remote Sensing*, 30(8):2133–2147, 2009.

[87] Frank Hung-Fat Leung, Hak-Keung Lam, Sai-Ho Ling, and Peter Kwong-Shun Tam. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural networks*, 14(1):79–88, 2003.

[88] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[89] Patrick Ngatchou, Anahita Zarei, and A El-Sharkawi. Pareto multi objective optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pages 84–91. IEEE, 2005.

[90] Kalyanmoy Deb and Himanshu Gupta. Searching for robust pareto-optimal solutions in multi-objective optimization. In *International conference on evolutionary multi-criterion optimization*, pages 150–164. Springer, 2005.

# Appendix A Image Dataset


Figure 35: Dataset used in comparing compression schemes and comparing wavelets. The first 10 images (from the top) are categorised as natural. The next 10 images are categorised as graphical. The last 10 images are categorised as text.

# Appendix B Derivations

## B.1 Reconstruction cost function with filters of length 2

For 1D 1-level DWT. Variables are as outlined in Section 5.1.

We first perform the forward transform to get $A_n$ and $D_n$ from $I_2n-1$ and $I_2n$,using **Ld** and **Hd** (**Hd** expressed as components in **Lr**):

$$A_n = I_{2n-1}Ld_1 + I_{2n}Ld_2 \tag{B.1}$$

$$D_n = -I_{2n-1}Lr_1 + I_{2n}Lr_2 \tag{B.2}$$

We then perform the inverse transform on $A_n$ and $D_n$ to get $R_2n-1$ and $R_2n$, using **Lr** and **Hr** (**Hr** expressed as components in **Ld**):

$$\begin{aligned} R_{2n-1} &= A_nLr_1 + D_nLr_2 \\ &= (I_{2n-1}Ld_1 + I_{2n}Ld_2)Lr_1 + (I_{2n}Lr_2 - I_{2n-1}Lr_1)Lr_2 \end{aligned} \tag{B.3}$$

$$\begin{aligned} R_{2n} &= A_nHr_1 + D_nHr_2 \\ &= (I_{2n-1}Ld_1 + I_{2n}Ld_2)Ld_1 - (I_{2n}Lr_2 - I_{2n-1}Lr_1)Ld_2 \end{aligned} \tag{B.4}$$

We now take the product of the partial derivatives and displacement between reconstruction and original pixels, and sum over all elements of the image and filter coefficients:

$$\begin{aligned} y(Ld_1) &= \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})\frac{\partial R_{2n-1}}{\partial Ld_1} + (R_{2n} - I_{2n})\frac{\partial R_{2n}}{\partial Ld_1}] \\ &= \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})(I_{2n-1}Lr_1) + (R_{2n} - I_{2n})(I_{2n-1} + I_{2n-1}Ld_1 + I_{2n}Ld_2)] \end{aligned} \tag{B.5}$$

$$\frac{\partial R_{2n-1}}{\partial Ld_1} = I_{2n-1}Lr_1$$

$$\frac{\partial R_{2n}}{\partial Ld_1} = 2Ld_1 \cdot I_{2n-1} + I_{2n}Ld_2$$

$$R_{2n-1} = A_nLr_1 + D_nLr_2$$
$$= (I_{2n-1}Ld_1 + I_{2n}Ld_2)Lr_1 + (I_{2n}Lr_2 - I_{2n-1}Lr_1)Lr_2$$

$$R_{2n} = A_nHr_1 + D_nHr_2$$
$$= (I_{2n-1}Ld_1 + I_{2n}Ld_2)Ld_1 - (I_{2n}Lr_2 - I_{2n-1}Lr_1)Ld_2$$

- $\frac{\partial R_{2n-1}}{\partial Ld_1} = I_{2n-1} Lr_1$

  $\frac{\partial R_{2n}}{\partial Ld_1} = 2I_{2n-1} Ld_1 + I_{2n} Ld_2$

- $\frac{\partial R_{2n-1}}{\partial Ld_2} = I_{2n} Lr_1$

  $\frac{\partial R_{2n}}{\partial Ld_2} = I_{2n} Ld_1 - (I_{2n} Lr_2 - I_{2n-1} Lr_1)$

- $\frac{\partial R_{2n-1}}{\partial Lr_1} = I_{2n-1} Ld_1 + I_{2n} Ld_2 - I_{2n-1} Lr_2$

  $\frac{\partial R_{2n}}{\partial Lr_1} = I_{2n-1} Ld_2$

- $\frac{\partial R_{2n-1}}{\partial Lr_2} = 2 I_{2n} Lr_2 - I_{2n-1} Lr_1$

  $\frac{\partial R_{2n}}{\partial Lr_2} = -I_{2n} Ld_2$

$$y(Ld_1) = \sum_{n=1}^{N/2} [(R_{2n-1} - I_{2n-1}) \frac{\partial R_{2n-1}}{\partial Ld_1} + (R_{2n} - I_{2n}) \frac{\partial R_{2n}}{\partial Ld_1}]$$
$$= \sum_{n=1}^{N/2} [(R_{2n-1} - I_{2n-1})(I_{2n-1} Lr_1) + (R_{2n} - I_{2n})(I_{2n-1} + I_{2n-1} Ld_1 + I_{2n} Ld_2)]$$

diff — odd — even — changed

$$y(Ld_2) = \sum_{n=1}^{N/2} [(R_{2n-1} - I_{2n-1}) \frac{\partial R_{2n-1}}{\partial Ld_2} + (R_{2n} - I_{2n}) \frac{\partial R_{2n}}{\partial Ld_2}]$$
$$= \sum_{n=1}^{N/2} [(R_{2n-1} - I_{2n-1})(I_{2n} Lr_1) + (R_{2n} - I_{2n})(I_{2n} Ld_1 + I_{2n-1} Lr_1 - I_{2n} Lr_2)] \quad \text{(B.6)}$$

diff — odd — even

$$y(Lr_1) = \sum_{n=1}^{N/2} [(R_{2n-1} - I_{2n-1}) \frac{\partial R_{2n-1}}{\partial Lr_1} + (R_{2n} - I_{2n}) \frac{\partial R_{2n}}{\partial Lr_1}]$$
$$= \sum_{n=1}^{N/2} [(R_{2n-1} - I_{2n-1})(I_{2n-1} Ld_1 + I_{2n} Ld_2 - I_{2n-1} Lr_2) + (R_{2n} - I_{2n})(I_{2n-1} Ld_2)]$$

odd — even

$$y(Lr_2) = \sum_{n=1}^{N/2} [(R_{2n-1} - I_{2n-1}) \frac{\partial R_{2n-1}}{\partial Lr_2} + (R_{2n} - I_{2n}) \frac{\partial R_{2n}}{\partial Lr_2}]$$
$$= \sum_{n=1}^{N/2} [(R_{2n-1} - I_{2n-1})(I_{2n} Lr_1) + (R_{2n} - I_{2n})(I_{2n} Ld_1 + I_{2n-1} Lr_1 - I_{2n} Lr_2)]$$

odd — even

Wrong in the report

Right in the code.

$$\begin{aligned} y(Ld_2) &= \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})\frac{\partial R_{2n-1}}{\partial Ld_2} + (R_{2n} - I_{2n})\frac{\partial R_{2n}}{\partial Ld_2}] \\ &= \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})(I_{2n}Lr_1) + (R_{2n} - I_{2n})(I_{2n}Ld_1 + I_{2n-1}Lr_1 - I_{2n}Lr_2)] \end{aligned} \tag{B.6}$$

$$\begin{aligned} y(Lr_1) &= \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})\frac{\partial R_{2n-1}}{\partial Lr_1} + (R_{2n} - I_{2n})\frac{\partial R_{2n}}{\partial Lr_1}] \\ &= \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})(I_{2n-1}Ld_1 + I_{2n}Ld_2 - I_{2n-1}Lr_2) + (R_{2n} - I_{2n})(I_{2n-1}Ld_2)] \end{aligned} \tag{B.7}$$

$$\begin{aligned} y(Lr_2) &= \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})\frac{\partial R_{2n-1}}{\partial Lr_2} + (R_{2n} - I_{2n})\frac{\partial R_{2n}}{\partial Lr_2}] \\ &= \sum_{n=1}^{N/2}[(R_{2n-1} - I_{2n-1})(I_{2n}Lr_1) + (R_{2n} - I_{2n})(I_{2n}Ld_1 + I_{2n-1}Lr_1 - I_{2n}Lr_2)] \end{aligned} \tag{B.8}$$

We have now arrived at a cost function tailor made for each wavelet coefficient in a 1D 1-level DWT using filters of length 2.

## B.2 Reconstruction cost function for filters of length 4

For 1D 1-level DWT. Most definitions outlined in section will be followed, with a few minor changes to increase the dimensions to 4.

$$\mathbf{I} = \begin{bmatrix} I_1 & I_2 & ... & I_n & ... & I_N \end{bmatrix} \quad \text{Original diagonal of length N} \tag{B.9}$$

$$\mathbf{I_d}(n) = \begin{bmatrix} I_{4n-3} & I_{4n-2} & I_{4n-1} & I_{4n} \end{bmatrix} \quad \text{General dyadic partition of original signal} \tag{B.10}$$

$$\mathbf{R} = \begin{bmatrix} R_1 & R_2 & ... & R_n & ... & R_N \end{bmatrix} \quad \text{Reconstructed signal of length N} \tag{B.11}$$

$$\mathbf{R_d}(n) = \begin{bmatrix} R_{4n-3} & R_{4n-2} & R_{4n-1} & R_{4n} \end{bmatrix} \quad \text{General dyadic partition of reconstructed signal} \tag{B.12}$$

$$\mathbf{Ld} = \begin{bmatrix} Ld_1 & Ld_2 & Ld_3 & Ld_4 \end{bmatrix} \quad \text{Low-pass decomposition filter of length 4} \tag{B.13}$$

$$\mathbf{Lr} = \begin{bmatrix} Lr_1 & Lr_2 & Lr_3 & Lr_4 \end{bmatrix} \quad \text{Low-pass reconstruction filter of length 4} \tag{B.14}$$

$$\mathbf{Hd} = \begin{bmatrix} -Lr_1 & Lr_2 & -Lr_3 & Lr_4 \end{bmatrix} \quad \text{High-pass decomposition filter of length 4} \tag{B.15}$$

$$\mathbf{Hr} = \begin{bmatrix} Ld_1 & -Ld_2 & Ld_3 & -Ld_4 \end{bmatrix} \quad \text{High-pass reconstruction filter of length 4} \tag{B.16}$$

$$\mathbf{A_n} = \begin{bmatrix} A_{n,1} & A_{n,2} & A_{n,3} \end{bmatrix} \quad \text{Approx. coefficients at nth dyadic operation} \tag{B.17}$$

$$\mathbf{D_n} = \begin{bmatrix} D_{n,1} & D_{n,2} & D_{n,3} \end{bmatrix} \quad \text{Detail coefficients at nth dyadic operation} \tag{B.18}$$

$$y(Ld_i) \quad \text{Cost for } Ld_i \tag{B.19}$$

We begin by obtaining $\mathbf{I_p}$ by padding $\mathbf{I}$ by 3

$$I_p = \begin{bmatrix} I_{4n-1} & I_{4n-2} & I_{4n-3} & I_{4n-3} & I_{4n-2} & I_{4n-1} & I_{4n} & I_{4n} & I_{4n-1} & I_{4n-2} \end{bmatrix} \tag{B.20}$$

$\mathbf{A_n}$ is obtained by downscaling; taking the 2nd, 4th and 6th element in $\mathbf{I_p} * \mathbf{Ld}$, giving us:

Wrong assumption

$$A_{n,1} = I_{4n-2}Ld_4 + I_{4n-3}Ld_3 + I_{4n-3}Ld_2 + I_{4n-2}Ld_1 \tag{B.21}$$

$$A_{n,2} = I_{4n-3}Ld_4 + I_{4n-2}Ld_3 + I_{4n-1}Ld_2 + I_{4n}Ld_1 \tag{B.22}$$

$$A_{n,3} = I_{4n-1}Ld_4 + I_{4n}Ld_3 + I_{4n}Ld_2 + I_{4n-1}Ld_1 \tag{B.23}$$

$\mathbf{D_n}$ is obtained by downscaling; taking the 2nd, 4th and 6th element in $\mathbf{I_p} * \mathbf{Hd}$ , giving us:

Wrong assumption

$$D_{n,1} = I_{4n-2}Lr_4 - I_{4n-3}Lr_3 + I_{4n-3}Lr_2 - I_{4n-2}Lr_1 \tag{B.24}$$

$$D_{n,2} = I_{4n-3}Lr_4 - I_{4n-2}Lr_3 + I_{4n-1}Lr_2 - I_{4n}Lr_1 \tag{B.25}$$

$$D_{n,3} = I_{4n-1}Lr_4 - I_{4n}Lr_3 + I_{4n}Lr_2 - I_{4n-1}Lr_1 \tag{B.26}$$

We now have expressions for each element in $\mathbf{A_n}$ and $\mathbf{D_n}$, which means we have successfully carried out the forward transform. To perform the inverse transform, we begin with applying dyadic upscaling to $\mathbf{A_n}$ and $\mathbf{D_n}$, giving us $\mathbf{A_{nu}}$ and $\mathbf{D_{nu}}$:

$$\mathbf{A_{nu}} = \begin{bmatrix} 0 & 0 & 0 & A_{n,1} & 0 & A_{n,2} & 0 & A_{n,3} & 0 & 0 & 0 \end{bmatrix} \tag{B.27}$$

$$\mathbf{D_{nu}} = \begin{bmatrix} 0 & 0 & 0 & D_{n,1} & 0 & D_{n,2} & 0 & D_{n,3} & 0 & 0 & 0 \end{bmatrix} \tag{B.28}$$

$\mathbf{A_{ni}}$ is obtained by taking the 3rd - 6th element in $\mathbf{A_{nu}} * \mathbf{Hr}$, giving us:

$$A_{ni,1} = A_{1,n}Ld_3 + A_{2,n}Ld_1 \tag{B.29}$$

$$A_{ni,2} = -A_{1,n}Ld_4 - A_{2,n}Ld_2 \tag{B.30}$$

$$A_{ni,3} = A_{2,n}Ld_3 + A_{3,n}Ld_1 \tag{B.31}$$

$$A_{ni,4} = -A_{2,n}Ld_4 - A_{3,n}Ld_2 \tag{B.32}$$

$\mathbf{D_{ni}}$ is obtained by taking the 3rd - 6th element in $\mathbf{D_{nu}} * \mathbf{Lr}$, giving us:

$$D_{ni,1} = D_{n,1}Lr_3 + D_{n,2}Lr_1 \tag{B.33}$$

$$D_{ni,2} = D_{n,1}Lr_4 + D_{n,2}Lr_2 \tag{B.34}$$

$$D_{ni,3} = D_{n,2}Lr_3 + D_{n,3}Lr_1 \tag{B.35}$$

$$D_{ni,4} = D_{n,2}Lr_4 + D_{n,3}Lr_2 \tag{B.36}$$

The reconstructed pixels are then given by:

$$R_{4n-3} = A_{ni,1} + D_{ni,1} \tag{B.37}$$

$$R_{4n-2} = A_{ni,2} + D_{ni,2} \tag{B.38}$$

$$R_{4n-1} = A_{ni,3} + D_{ni,3} \tag{B.39}$$

$$R_{4n} = A_{ni,4} + D_{ni,4} \tag{B.40}$$

The cost is compiled by taking the product of the partial derivatives and displacement between reconstruction and original pixels, and summing over all elements of the image and filter coefficients:

$$y(Ld_i) = \sum_{n=1}^{N/4}[(R_{4n-3} - I_{4n-3})\frac{\partial R_{4n-3}}{\partial Ld_i} + (R_{4n-2} - I_{4n-2})\frac{\partial R_{4n-2}}{\partial Ld_i} + (R_{4n-1} - I_{4n-1})\frac{\partial R_{4n-1}}{\partial Ld_i} + (R_{4n} - I_{4n})\frac{\partial R_{4n}}{\partial Ld_i}] \tag{B.41}$$

$$y(Lr_j) = \sum_{n=1}^{N/4}[(R_{4n-3} - I_{4n-3})\frac{\partial R_{4n-3}}{\partial Lr_j} + (R_{4n-2} - I_{4n-2})\frac{\partial R_{4n-2}}{\partial Lr_j} + (R_{4n-1} - I_{4n-1})\frac{\partial R_{4n-1}}{\partial Lr_j} + (R_{4n} - I_{4n})\frac{\partial R_{4n}}{\partial Lr_j}] \tag{B.42}$$

The full expansions are shown below:

$$\begin{aligned} y(Ld_1) = \sum_{n=1}^{N/4}[&(R_{4n-3} - I_{4n-3})(I_{4n-2}Lr_3 + I_{4n}Lr_1 + D_{n,2}) \\ &+(R_{4n-2} - I_{4n-2})(I_{4n-2}Lr_4 + I_{4n}Lr_2) \\ &+(R_{4n-1} - I_{4n-1})(I_{4n}Lr_3 + I_{4n-1}Lr_1 + D_{n,3}) \\ &+(R_{4n} - I_{4n})(I_{4n}Lr_4 + I_{4n-1}Lr_2)] \end{aligned} \tag{B.43}$$

$$\begin{aligned} y(Ld_2) = \sum_{n=1}^{N/4}[&(R_{4n-3} - I_{4n-3})(I_{4n-3}Lr_3 + I_{4n-1}Lr_1) \\ &+(R_{4n-2} - I_{4n-2})(I_{4n-3}Lr_4 + I_{4n-1}Lr_2 - D_{n,2}) \\ &+(R_{4n-1} - I_{4n-1})(I_{4n-1}Lr_3 + I_{4n}Lr_1) \\ &+(R_{4n} - I_{4n})(I_{4n-1}Lr_4 + I_{4n}Lr_2) - D_{n,3}] \end{aligned} \tag{B.44}$$

$$\begin{aligned} y(Ld_3) = \sum_{n=1}^{N/4}[&(R_{4n-3} - I_{4n-3})(I_{4n-3}Lr_3 + I_{4n-2}Lr_1 + D_{n,1}) \\ &+(R_{4n-2} - I_{4n-2})(I_{4n-3}Lr_4 + I_{4n-2}Lr_2) \\ &+(R_{4n-1} - I_{4n-1})(I_{4n-2}Lr_3 + I_{4n}Lr_1 + D_{n,2}) \\ &+(R_{4n} - I_{4n})(I_{4n-2}Lr_4 + I_{4n}Lr_2)] \end{aligned} \tag{B.45}$$

$$\begin{aligned} y(Ld_4) = \sum_{n=1}^{N/4}[&(R_{4n-3} - I_{4n-3})(I_{4n-2}Lr_3 + I_{4n-3}Lr_1) \\ &+(R_{4n-2} - I_{4n-2})(I_{4n-2}Lr_4 + I_{4n-3}Lr_2 - D_{n,1}) \\ &+(R_{4n-1} - I_{4n-1})(I_{4n-3}Lr_3 + I_{4n-1}Lr_1) \\ &+(R_{4n} - I_{4n})(I_{4n-3}Lr_4 + I_{4n-1}Lr_2) - D_{n,2}] \end{aligned} \tag{B.46}$$

$$
\begin{aligned}
y(Lr_1) = \sum_{n=1}^{N/4} [&(R_{4n-3} - I_{4n-3})(-I_{4n-2}Ld_3 - I_{4n}Ld_1 + A_{n,2}) \\
&+(R_{4n-2} - I_{4n-2})(I_{4n-2}Ld_4 + I_{4n}Ld_2) \\
&+(R_{4n-1} - I_{4n-1})(-I_{4n}Ld_3 - I_{4n-1}Ld_1 + A_{n,3}) \\
&+(R_{4n} - I_{4n})(I_{4n}Ld_4 + I_{4n-1}Ld_2)]
\end{aligned} \tag{B.47}
$$

$$
\begin{aligned}
y(Lr_2) = \sum_{n=1}^{N/4} [&(R_{4n-3} - I_{4n-3})(I_{4n-3}Ld_3 + I_{4n-1}Ld_1) \\
&+(R_{4n-2} - I_{4n-2})(-I_{4n-3}Ld_4 - I_{4n-1}Ld_2 + A_{n,2}) \\
&+(R_{4n-1} - I_{4n-1})(I_{4n-1}Ld_3 + I_{4n}Ld_1) \\
&+(R_{4n} - I_{4n})(-I_{4n-1}Ld_4 - I_{4n}Ld_2 + A_{n,3})]
\end{aligned} \tag{B.48}
$$

$$
\begin{aligned}
y(Lr_3) = \sum_{n=1}^{N/4} [&(R_{4n-3} - I_{4n-3})(-I_{4n-3}Ld_3 - I_{4n-2}Ld_1 + A_{n,1}) \\
&+(R_{4n-2} - I_{4n-2})(I_{4n-3}Ld_4 + I_{4n-2}Ld_2) \\
&+(R_{4n-1} - I_{4n-1})(-I_{4n-2}Ld_3 - I_{4n}Ld_1 + A_{n,2}) \\
&+(R_{4n} - I_{4n})(I_{4n-2}Ld_4 + I_{4n}Ld_2)]
\end{aligned} \tag{B.49}
$$

$$
\begin{aligned}
y(Lr_4) = \sum_{n=1}^{N/4} [&(R_{4n-3} - I_{4n-3})(I_{4n-2}Ld_3 + I_{4n-3}Ld_1) \\
&+(R_{4n-2} - I_{4n-2})(-I_{4n-2}Ld_4 - I_{4n-3}Ld_2 + A_{n,1}) \\
&+(R_{4n-1} - I_{4n-1})(I_{4n-3}Ld_3 + I_{4n-1}Ld_1) \\
&+(R_{4n} - I_{4n})(-I_{4n-3}Ld_4 - I_{4n-1}Ld_2 + A_{n,2})]
\end{aligned} \tag{B.50}
$$

We have now arrived at a cost function tailor made for each wavelet coefficient in a 1D 1-level DWT using filters of length 4.

Department of Engineering Science

Supplementary Questions for 4th Year Project Students

UNIVERSITY OF OXFORD

| Factor | Answer | Things to Consider | Record details here |
|---|---|---|---|
| Has the checklist covered all the problems that may arise from working with the VDU? | ☐ Yes ☐ No | | |
| Are you free from experiencing any fatigue, stress, discomfort or other symptoms which you attribute to working with the VDU or work environment? | ☐ Yes ☐ No | Any aches, pains or sensory loss (tingling or pins and needles) in your neck, back shoulders or upper limbs. Do you experience restricted joint movement, impaired finger movements, grip or other disability, temporary or permanently | |
| Do you take adequate breaks when working at the VDU? | ☐ Yes ☐ No | Periods of two minutes looking away from the screen taken every 20 minutes and longer periods every 2 hours<br><br>Natural breaks for taking a drink and moving around the office answering the phone etc. | |
| How many hours per day do you spend working with this computer? | ☐ 1-2 ☐ 3-4<br>☐ 5-7 ☐ 8 or more | | |
| How many days per week do you spend working with this computer? | ☐ 1-2 ☐ 3-5<br>☐ 6-7 | | |
| Please describe your typical computer usage pattern | | | |

| **Student Declaration and Academic Approval** | |
|---|---|
| Student Declaration:<br><br>I have completed the DSE Workstation Checklist and the Supplementary Questions for my computer-related risk assessment for 4YP Project Number indicated below:<br><br>4YP Project Number: ……<br><br>4YP Student's Name *(please print)* ……..<br><br>4YP Student's Signature: …… | Academic Approval<br><br>I confirm my approval of this 4YP DSE Risk Assessment.<br><br>Academic Supervisor's Name: *(please print)*<br><br>……<br><br>Academic Supervisor's Signature:<br><br>…… |