UPPSALA
UNIVERSITET

UPTEC IT 22009

Examensarbete 30 hp

Juni 2022

Bildkomprimering med autoencoder och discrete cosine transform

Martin Larsson

Civilingenjörsprogrammet i informationsteknologi

UPPSALA
UNIVERSITET

Image compression with autoencoder & the discrete cosine transform

Martin Larsson

Abstract

Millions of pictures are captured each year for different purposes, making digital images an ubiquitous part of modern day life. This proliferation was made possible by image compression standards since these images need to be stored somewhere and somehow. In this thesis I explore the use of machine learning together with the discrete cosine transform to compress images.

An autoencoder was developed which was able to compress images with results comparable to the JPEG standard. The results lend credence to the hypothesis that the combination of a simple autoencoder and the discrete cosine transform offers a simple and effective method for image compression.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala/Visby

Handledare: Erik Löthbom & Mikael Axelsson, Consid AB Ämnesgranskare: Filip Malmberg

Examinator: Lars-Åke Nordén

Sammanfattning

Miljontals bilder fångas varje år för diverse ändamål vilket har gjort att digitala bilder numera är en allestädes närvarande del av vårt moderna samhälle. Denna utveckling möjliggjordes av bildkomprimeringsstandarder eftersom alla dessa bilder måste lagras någonstans och på något sätt. I denna rapport utforskar jag möjligheten att använda maskininlärningstekniker ihop med den diskreta cosinustransformen för att komprimera bilder. En autoencoder togs fram som kunde komprimera bilder tillräckligt väl för att jämföras med JPEG standarden. Resultaten visar att en kombination utav en autoencoder och det diskreta cosinustransformen kan erbjuda en simpel och effektiv metod för bildkomprimering.

Contents

1 Introduction

The first digital recreation of an image was in 1957 at the National Institute of Standards and Technology (NIST) where a picture of a baby was scanned and displayed [13]. 35 years later, a photo of the performance group called the Cernettes, comprised of administrative assistants and significant others of scientists at CERN, was uploaded to the newly invented World Wide Web [15]. That same year the Joint Photographic Experts Group introduced the JPEG image standard which would lead to the proliferation of digital images we experience today where millions of images are taken, stored and sent over the internet every day [22].

This process can seem somewhat magical and relatively few understand just how much work and technology goes into making that possible: all the protocols in the different layers of the internet protocol suite down to the complex schematics of the hardware it is sent over and stored upon. Whenever one of the components of this process (or any piece of information technology) advances it creates incentive and opportunity for the others to catch up.

A similar development has taken place recently in the field of machine learning. Access to cloud computing has truly democratised access to the resources necessary to study and develop new techniques and put them to use in new areas. Granted, machine learning had already been applied to a plethora of areas and problems before cloud computing but it allows for those without access to supercomputers or data centers to try their methods as well.

This thesis explores the viability of one of the areas that machine learning has long been applied to but has yet to see widespread use, namely image compression. In particular, I explore the use of the discrete cosine transform (used in the traditional JPEG standard) to enhance the performance of a compressing autoencoder. Ultimately, a model was developed that takes inspiration from existing standards in order to enhance the performance of the machine learning algorithms.

1.1 Purpose & Goal

Millions of images are captured every day for a number of different purposes and all this data needs to be stored somewhere and somehow. Raw image data is useful for professionals when editing and processing images, however for most usage some form of compression is needed for the use of images to scale. Compressed images require less storage and generate less traffic when sent between devices, two driving factors of cost in modern applications.

Machine learning has the potential to offer a more flexible and general compression framework than traditional image compression methods. In particular, this thesis explores the viability of using an autoencoder in conjunction with the discrete cosine transform in order to remove the least amount of detail while retaining visual fidelity. The same transform was used in the JPEG standard which uses a fixed filter to remove components of the image which are ordinarily not very important (i.e. lower frequency coefficients). An autoencoder could instead offer more flexibility by analysing each image and remove components in such a way that least harms the visual fidelity of that particular image.

Achieving a better compression rate than traditional methods using an autoencoder and the discrete cosine transform may also provide further insights into other areas of image analysis using the same method. A machine learning model has no inherit purpose but rather it is imbued with a purpose during training to achieve a specific goal. By slightly pivoting the design of the autoencoder it could be applied to image classification, image reconstruction and feature detection, areas that has yet to be thoroughly explored using machine learning along with the discrete cosine transform.

The goal of this project is to develop an autoencoding model that can reduce image data to entropy, thus compressing it and then reconstruct it well enough for the user to recognise the image. The resulting reconstruction should be at least comparable to the well known JPEG standard [22] in such a way that compression down to similar file sizes should produce comparable visual fidelity.

There exists established image compression standards and studies has been done into using autoencoder for this purpose. This project will continue on this work and explore the viability of using the discrete cosine transform as a step in the process of the autoencoder.

1.2 Delimitations

The model proposed in this project does not provide progressive encoding in the like of the JPEG 2000 standard. Progressive encoding enables the biggest features of an image to be stored, sent and rendered separately from smaller ones which is especially useful in cases where an image is sent over an unreliable or poor internet connection. Rather than the user having to wait for the entire image to load or seeing the image being rendered piece by piece the larger elements such as sky or foreground can be rendered first, followed by details such as people or flowers. Since the model computes all parts of the image simultaneously this is not possible.

Further, since the lossy part of the compression in the model is performed by the encoder or analysis process, variable rate compression is not readily available. Models with differing layer structures would need to be developed and trained in order to control the trade-off between quality and resulting size of the compressed entropy.

Finally, since the main goal of this project is to explore the feasibility of using the discrete cosine transform in a compressing autoencoder the implementation will not entail any end to end image compression functionality. If used in practice the image would most likely be split into several smaller 16x16 or 32x32 segments, each fed to the autoencoder separately, then combined to form the entire file. However, this is beyond the scope of this thesis and would have to be implemented in the future.

2 Background

In this section I will discuss the background of the project and relevant areas such as digital images and machine learning. The first time an image was recreated digitally was in 1957 at the National Institute of Standards and Technology (NIST) where a picture of a baby was scanned and displayed [13]. Data compression dates back even further, with techniques such as Huffman coding introduced in 1952 [9] and later in 1965 transform coding was popularized with the Fast Fourier Transform (FFT) [6]. Most relevant to this paper is the transform coding technique known as the Discrete Cosine Transform (DCT), first proposed by Nasir Ahmed in 1972 [2] which was used in the image compression format JPEG in 1992 [22].

Figure 1 To the left: the first image uploaded on the World Wide Web of the band The Cornettes, consisting of administrative assistants and significant others to scientists working at the CERN supercollider. To the right: the first digitally scanned image from 1957 of a baby at National Institute of Standards and Technology (NIST).

2.1 Compression

In computer science, compression is the process of encoding data in such a way that fewer bits are needed to represent the information, either losslessly or with loss in the information (i.e. lossy compression). Lossless compression encodes the data by removing statistically redundant information in such a way that the original signal can be reconstructed perfectly. Lossy compression disregards less important information in order to compress the signal further, consequently making it impossible to perfectly reconstruct the original signal.

For data such as images there is almost always some redundancy. Rather than describing an entirely black images with a value for each of the 500 pixels the image can be described by simply stating it contains 500 black pixels. This is how lossless compression can reduce the size of the data, by performing statistical analysis and finding patterns. Lossy and lossless compression can also be used in conjunction with each other. For example, small or imperceivable details in an image can be discarded first followed by lossless compression performed on the remaining data.

Often there is redundancy in terms of detail in images due to biological limitations of our visual system and other circumstantial reasons such as viewing the image from a distance. Different uses also requires different amounts of data, capturing and editing photos in high quality gives the photographer more precision but 50 megapixel images are not needed for most every day use cases.

2.2 Fourier transform

A Fourier transform is a mathematical method to express a function in the time or space domain as frequency components in the spatial or frequency domain. It's origins dates back to 1822, when Joseph Fourier first showed that all periodic function can be expressed as a weighted sum of sines and cosines [7].

The weights of this sum is called the Fourier coefficients and together these form the Fourier series of any particular function. More complicated functions require more coefficients in order to describe it accurately and so the series has an inherent interval associated with it that controls the accuracy to which the function can be replicated.

The Fourier transform is an extension to the series that decomposes the signal into these coefficients, thus representing the signal in the frequency domain rather than the time or spatial domain. For example, a chord played on a guitar exists as we hear it in the time domain, where oscillations of the strings form audible sound waves. By using the

Fourier transform it is possible to express the signal in the frequency domain where the sound is decomposed into its underlying frequencies and thus discern what note is being played.

The Fourier transform is defined as:

$$F(\omega) = \mathcal{F}(f(t)) = \int_{-\infty}^{\infty} f(t) \cdot e^{-i\omega t} dt \tag{1}$$

where ω denotes the transform into the frequency domain. The inverse transform back to the time domain is defined as:

$$f(t) = \mathcal{F}^{-1}(F(\omega)) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) \cdot e^{i\omega t} dt \tag{2}$$

There are also discrete versions of the Fourier transform (DFT) with one widely used algorithm being the Fast Fourier Transform (FFT) which was introduced in 1965 [6]. The FFT algorithm was a turning point in digital signal processing due to the increased performance and reduced time complexity. Traditional DFT methods computed the transform in $\mathcal{O}(N^2)$ whereas FFT was able bring down the complexity to $\mathcal{O}(N \log N)$.

2.3 Visual perception and color representation

There are several ways of quantifying and representing color. Which representation is best depends on the use case, for example splitting up the color into its red, green and blue components is often used when displaying an image on a display while a cyan, yellow, magenta and key (black) representation is used in color print. In popular technologies such as liquid-crystal display (LCD) the screen is comprised of thousands of pixels, each divided into three segments of red, green and blue. Together, at a distance, these segments appear to the human eye as single color.

However, human visual perception is more complicated than simply segmenting the colors. For example, we are able to detect nuances in brightness to a greater degree than we are at nuances in color [14]. This means that for compression purposes, if some of the color information is removed while retaining the brightness information the visual fidelity of the image should remain comparable.

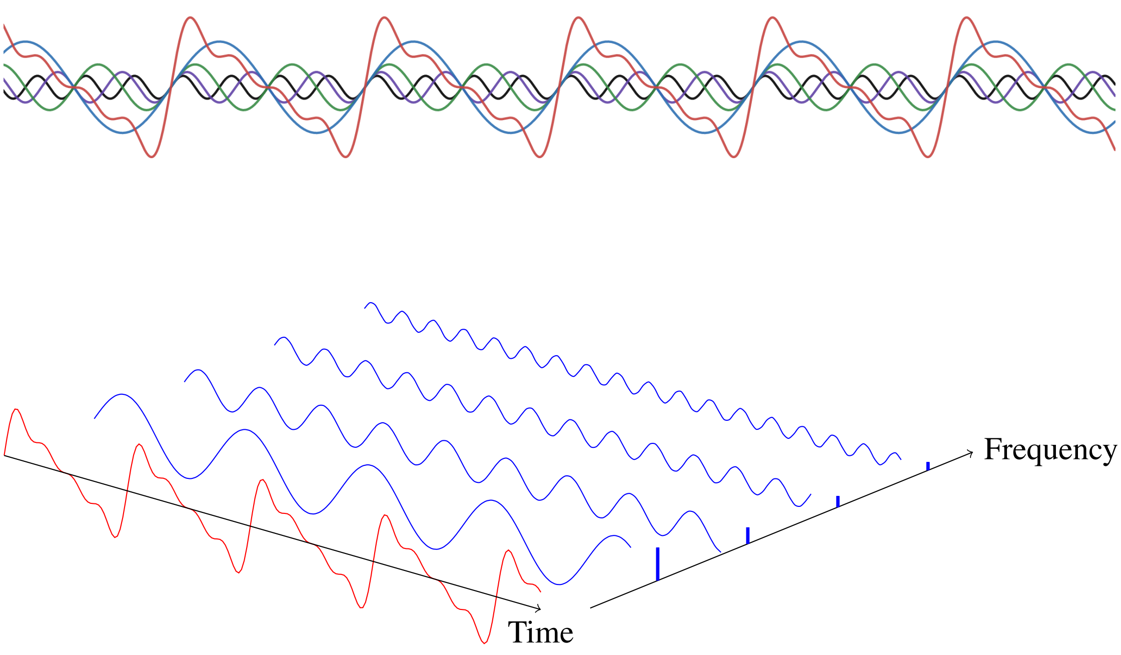

Figure 2 Demonstration of Fourier series decomposition. On the top is the functions sinusoid in red which is composed of the other sinusoids. Below is the decomposition of the signal into its frequency components. On one axis is the original time domain, where the sinusoid oscillates back and forth. On the other axis is the frequency domain, showing the amplitude of each of the sinusoids in the red signal, i.e. the blue bars are the Fourier coefficients of the red signal.

2.4 JPEG

The JPEG standard was first introduced in 1992 by a committee known as the Join Photographic Experts Group. The goal of JPEG was to establish the first international compression standard for continuous-tone still images in both gray-scale and color [22]. JPEG is a lossy compression, meaning concessions in image quality and data retention are made when performing the compression.

JPEG takes advantage of the fact that we humans are better at distinguishing between nuances in brightness than we are at color by converting the image from RGB to the color space Y'C_bC_r. This color space divides each pixel into three channels, Y' being the brightness and C_b and C_r being the blue-difference and red-difference chroma channels respectively. The C_b and C_r components can be reduced by some ratio in order to be able to compress the image more efficiently, again without compromising the perceived visual quality significantly.

After the image has been converted into Y'C_bC_r each channel is processed in blocks and transformed to the frequency domain using the discrete cosine transform (DCT). The size of the block depends on implementation, however 16x16 is a common block size. After transforming the block and considering the data as a signal, large features of the block are represented by lower frequencies and details such as noise are represented by small frequencies. Using DCT, the coefficients of these cosines can be calculated revealing which frequencies the signal is constructed of and producing a matrix of coefficients. DCT is further explained in section 4.2.

Lower frequencies often contribute more than higher frequencies to our perception of the image, since larger features are more noticeable than details on images. Consequently, the coefficients of lower frequencies can usually be disregarded and set to zero, making it possible to compress the signal. This is done by dividing each value in the coefficient matrix by its corresponding value in whats called a quantization matrix and rounding the result. By dividing the higher frequencies coefficients by larger values than lower frequencies they have a greater chance of being non-zero after rounding. An example of this can be seen in figure 3.

Since natural images usually don't differ significantly between blocks the assumption is that this block compression will not be very noticeable. It can however result in the compressed JPEG images having a 'blocky' appearance since the compression of subsequent segments had differing results. Further, the assumption of images being continuous between segments of course doesn't hold when there is sudden contrast in the image, e.g. when there is a wall next to a sky or black text on a bright background.

$$G = \begin{bmatrix} 51 & 54 & 60 & 56 \\ 63 & 69 & 51 & 61 \\ 55 & 54 & 57 & 60 \\ 57 & 63 & 64 & 66 \end{bmatrix} Q = \begin{bmatrix} 10 & 15 & 30 & 50 \\ 15 & 15 & 30 & 60 \\ 30 & 30 & 50 & 150 \\ 50 & 60 & 150 & 200 \end{bmatrix} B = \begin{bmatrix} 5 & 4 & 2 & 1 \\ 4 & 5 & 2 & 1 \\ 2 & 2 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Figure 3 Example of quantization using DCT with 4x4 blocks. The transformed block data G and quantization matrix Q is used to produce B, such that $B_{ij} = round(G_{ij}/Q_{ij})$. The values in Q are chosen in such a way that values in the lower right of B is likely to evaluate to 0.

2.5 JPEG 2000

JPEG 2000 was introduced in 2001 as a successor to JPEG. Rather than transforming the signal with DCT, JPEG 2000 uses the discrete wavelet transform (DWT). This transform is done in steps, first extracting lower frequency attributes then higher frequencies for each subsequent step. By transforming the signal in steps large features can be rendered first, followed by finer and finer details. This is called progressive processing and is not possible with JPEG due to the block wise processing.

The progressive encoding allows for more flexibility of the codestream when transferring the image. For example, if an image is sent over a poor internet connection the stream can be truncated at any point and the data that has already been sent can be rendered to the user, with the only consequence being the lower resolution of the image. However, this flexibility comes with the cost of more complex codecs that require more computing,

2.6 Artificial intelligence

Artificial intelligence (AI) is an umbrella term in the field of computer science that refers to the appearance of human intelligence in machines. As a concept AI has been discussed and theorized for centuries, however as it pertains to computers it was first introduced by Alan Turing in the 1940's, then started in earnest as an academic field in 1955 [10]. Any program that exhibit traits that are associated with human intelligence such as problem-solving, learning patterns or generalising are often called artificial intelligence, making the exact definition ill defined and nebulous.

There is also a tendency for AI algorithms to be considered mundane over time as novel, more advanced algorithms are developed. Technology or algorithms that is considered revolutionary now will most likely be taken for granted and held in lower regard some years from now as new techniques are developed. Despite the vague definition however (or perhaps because of it), AI has since its conception seen widespread use in a range of areas, from machine learning and natural language processing to playing games and performing simulations.

2.7 Machine learning

Machine learning is a part of artificial intelligence that studies algorithms that learn through experience. The core of the algorithms is a model that can make predictions based on inputs by first training on sample data known as “training data.” [3] There are three main categories of machine learning, depending on the end goal of the algorithm.

In *supervised learning* the model is presented with an input and the desired output in order for a general rule to be generated that can map inputs to outputs [12]. This is in contrast to *unsupervised learning* where no previous patterns or rules are known and it’s up to the model to find patterns. Lastly, a model based on *reinforcement learning* is presented with an input, delivers an output and receives feedback based on the output which the model will try to maximise, thus learning to react to a dynamic environment.

Machine learning has for a long time been used in a number of applications, however the technology has not seen true widespread use until recent years with the advent of cloud computing. Today machine learning sees use in a plethora of areas, both in commercial and academic settings. Taste prediction for music or video streaming services, playing games, natural language processing, marketing and malady detection are some examples, just to name a few.

2.8 Consid AB

This thesis was carried out at Consid AB, an IT consulting company first established in Vetlanda in 2000 by Peter Hallgren and Henrik Sandell. Today Consid has around 1000 employees in 22 cities in Sweden [1].

3 Related work

In this section I will discuss previous contributions to the field of image compression using machine learning. Traditional methods of compressing images using some form of transform, quantization and bit encoding greatly resembles the concept of an encoder-decoder paradigm where the signal is analysed through a bottleneck then reconstructed to an approximation of the original signal. Thus, it has long been speculated that auto-encoders have great potential for image compression although results long remained unsatisfactory. Recent advancements in deep learning and convolutional networks has invigorated this potential however, with works showing results that exceed established formats.

3.1 Lossy image compression with autoencoder

Traditional image compression methods broadly resemble the structure of an autoencoder however the devil is in the details. Several issues may arise when directly translating one method to another, for example the derivative of the rounding function used in quantization is 0 everywhere except at integers where it's undefined, which poses a problem in back propagation. There is also the issue of constructing a good objective function that can accurately measure the distortion resulting from compressing the signal.

Ballé et al. developed a method of measuring the distortion from an autoencoder using a normalised Laplacian pyramid, providing a measurement more in line with human vision than PSNR or SSIM [4].

Toderici et al. [20] used a recurrent neural network (RNN) for minimising the distortion when compressing the image down to a set number of bits. The encoding and decoding was done iteratively and the decoding was designed to take residuals into account for each subsequent iteration, together allowing for progressive coding of the image. They also used long short term memory (LSTM) models for both the encoder and decoder, the design of which was further explored in later works by comparing LSTM with associative LSTM and found different use cases for each method [21]. They found that associative LSTM was mostly useful in the decoder.

Theis et al. further explored the use of deep convolutional networks and developed a way of dealing with the non-differentiability by using a continuous and differentiable approximation [19].

Rippel et al. developed an autoencoder with a pyramidal analysis, resembling the DWT used in JPEG 2000, along with an adaptive coding module that analyses nearby pixels in order to estimate the value of the current pixel. Adversarial training was also used to supplement the model [16].

3.2 Machine learning involving DCT

The focus on deep learning in regards to image analysis has largely revolved around analysing raw pixels. Methods involving DCT has been explored somewhat, such as by Zou et al. who used a 2-layer sparse autoencoder for feature extraction with promising results [24]. Before that Karami et al. used a 3D-DCT method for compressing hyperspectral image (images that cover the entire light spectrum) along with a support vector machine (SVM) to good results [11]. A SVM resembles a neural network and are used for similar problems, giving credence to the idea of using DCT in an autoencoder.

4 Method & techniques

In this section I will discuss the different techniques used in the system. Constructing a well-performing machine learning model is somewhat of an art form with no real guidelines other than previous work and experience. Therefor, the model presented was intentionally kept quite simple in order to keep the effect of using the discrete cosine transform in focus.

4.1 YUV color encoding

YUV is a color space resulting from transforming the RGB color space into a luma component *Y* (brightness) and two chroma components, blue (U) and red (V). As discussed in section 2.3 human eyes are more sensitive to nuances in brightness than they are in color, a fact that was used when JPEG was first introduced with the use of the Y'$\mathrm{C_bC_r}$ color space. The YUV component values are calculated by

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3}$$

4.2 Discrete Cosine Transform

The discrete cosine transform (DCT) is a Fourier-like transform first introduced in 1972 by Nasir Ahmed [2]. It is widely used in modern signal processing, especially in lossy compression as it is used in the JPEG image compression standard [22].

As discussed in section 2.2, a Fourier transform is mathematical method to express a function in the time or space domain as frequency components in the spatial or frequency domain. For example, a recording of a piano can be expressed in terms of its volume and frequency being played at any particular moment. Another example would be to decompose an image into its underlying frequencies where larger frequencies would represent larger features such as a gradient in the sky and smaller ones would represent details such as leaves on a tree. By transforming the signal into the spatial or frequency domain it becomes easier to disregard certain components of the signal without harming the fidelity, thus compressing the signal.

Ordinary Fourier transforms expresses the signal in an infinite set of sinusoids with different amplitudes, something that is not possible in a discrete setting. Therefore, any discrete version of a Fourier transform (DFT) will express the signal in a finite set of sinusoids, as is the case with DCT. However, as opposed to a DFT where the signal is expressed as both cosines and sines by using complex exponentials, DCT uses only cosines as the name implies.

DCT closely approximates the optimal principal component analysis (PCA) of the input through its compact representation. The transform often results in most of the coefficients being very small since a small subset of them contain most of the visually significant information. By identifying these coefficients in a deep network one can achieve good feature extraction [24] which can be used for compression by disregarding some coefficients.

There are several variants of DCT with DCT-II being the most common and is often referred to as just DCT. This version (for 2D inputs) is defined as

$$X_{i,j} = \sum_{m=0}^{W-1} \sum_{n=0}^{H-1} x_{i,j} \cos\left[\frac{\pi}{W}\Big(m + \frac{1}{2}\Big)i\right] \cos\left[\frac{\pi}{H}\Big(n + \frac{1}{2}\Big)j\right] \tag{4}$$

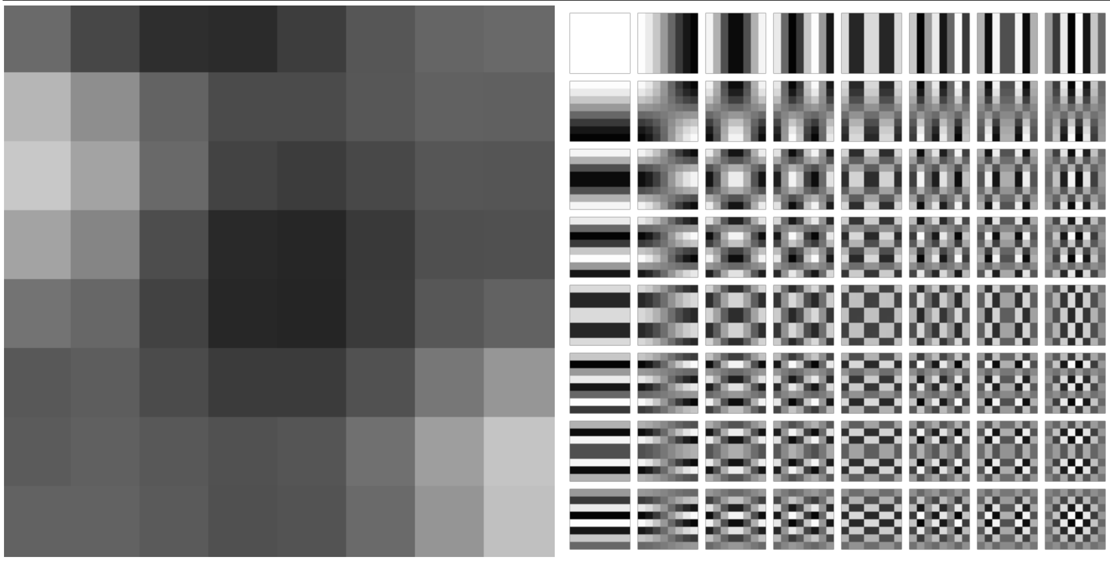

Figure 4 To the left is an example of a 8x8 block of a 8-bit grayscale image. To the right is a visualisation of the two-dimensional DCT basis functions. The DCT transforms the block into a linear combination of these patterns. Normally, the patterns to the upper left play a bigger role in the image since most features are mostly made up of low frequencies.

4.3 Artificial neural network

An artificial neural network, often simply called neural networks are a collection of nodes connected by edges in a directed graph and subsets of nodes are grouped into what's called layers. There are three types of layers: the input layer that receives data, hidden layers that transform it and an output layer. The most basic neural network consists of one of each of these layers and if there are multiple hidden layers it is considered a *deep* neural network. A basic neural network schema can be seen in figure 6.

Different types of neurons exists with special parameters and additional steps, however the most common and basic neuron consists of a vector of n inputs, a bias and an activation function. A weighted sum of the neurons inputs is taken, i.e. the sum of all inputs x_i multiplied by the weight associated with that input w_i, then the bias b is added to the sum which is then passed through an activation function before being sent as output.

$$o_n(x) = b_n + \sum_{i=0}^{n} x_i \cdot w_i \tag{5}$$

The process of passing the input through the network to produce the output is called forward propagation. Initially, unless the weights and biases are carefully chosen, this process will produce random and unuseful results. The network requires a method for automatically tuning the parameters such as weights and biases based on how well it is currently performing.

During training, the output of the network is compared to the expected result associated with the given input and the difference is used to update the weights and other parameters. Updates are first performed on the output layer, then backwards through the nodes and edges of the hidden layers and finally on the input nodes. This process is called backward propagation.

4.4 Autoencoder

An autoencoder is a neural network that has been trained to copy its input to its output. It consists of two main parts, an encoder f that takes an input x and produces entropy e such that $f(x) = e$ and a decoder g that converts the entropy back to a approximation of the original signal x such that $g(e) = \hat{x}$ [8].

The encoder is designed as a bottleneck thus making the entropy smaller than the input.

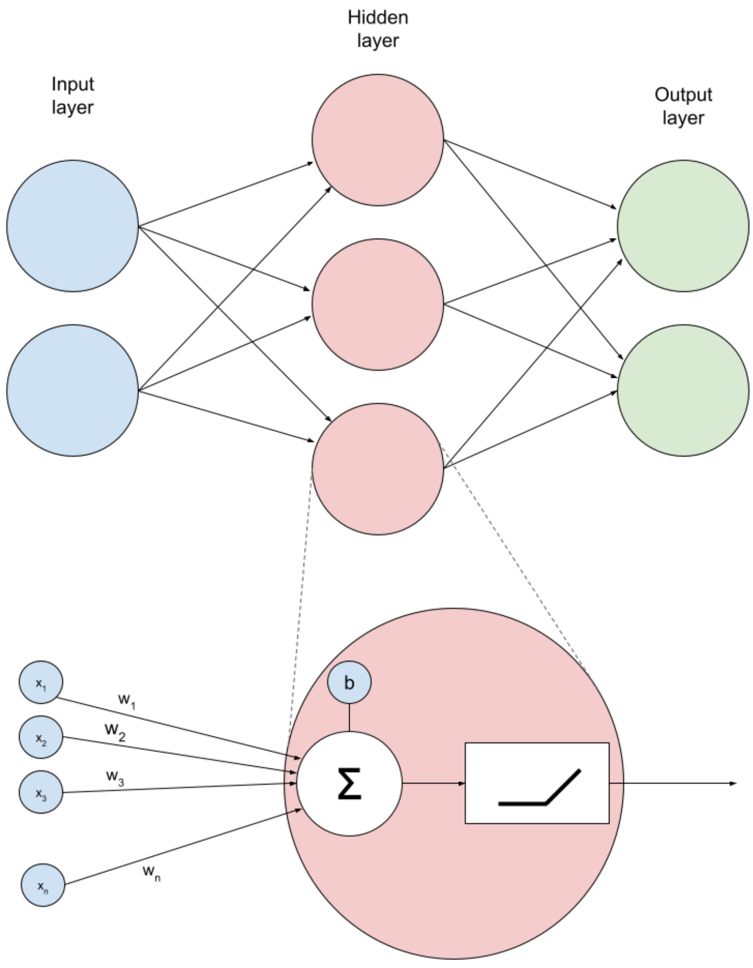

Figure 5 Basic neural network structure with one input, hidden and output layer along with more detailed depiction of a basic neuron. The neuron outputs a weighted sum of the inputs and a bias which is sent through an activation function. Depicted here is the rectifier activation function which returns the positive values for positive arguments and 0 for negative arguments.

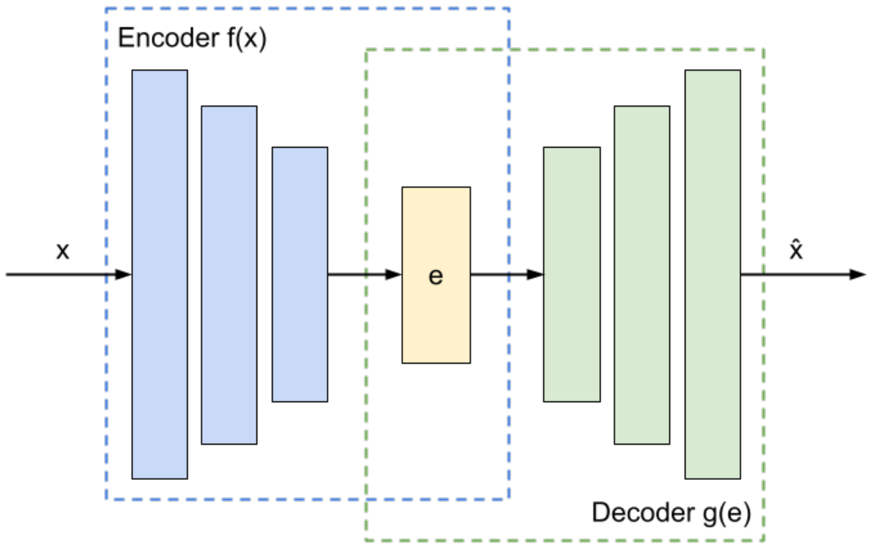

Figure 6 Overview of an autoencoder. The layers of the encoder network act as a bottleneck to compress the signal, ultimately producing the entropy e. This is used as input to the decoder network that produces the reconstructed signal $\hat{x}$.

This design makes the autoencoder resemble traditional compression methods where the original signal is compressed through a bottleneck ten decompressed to its approximation.

4.5 Convolutional neural network

Convolutional neural network (CNN) is a class of network that are very specialized for processing data with a grid-like topology [8] such as images. The name comes from the mathematical operation called convolution that is a linear operation and is often denoted with an asterisk($*$). A discrete convolution s can be seen as a weighted sum of signal input x at time index t with kernel k such that

$$s(t) = (x * k)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{6}$$

The convolution can be expanded to be used over more axis, for example for use with images. For a two dimensional image I we use a two dimensional kernel K and iterate

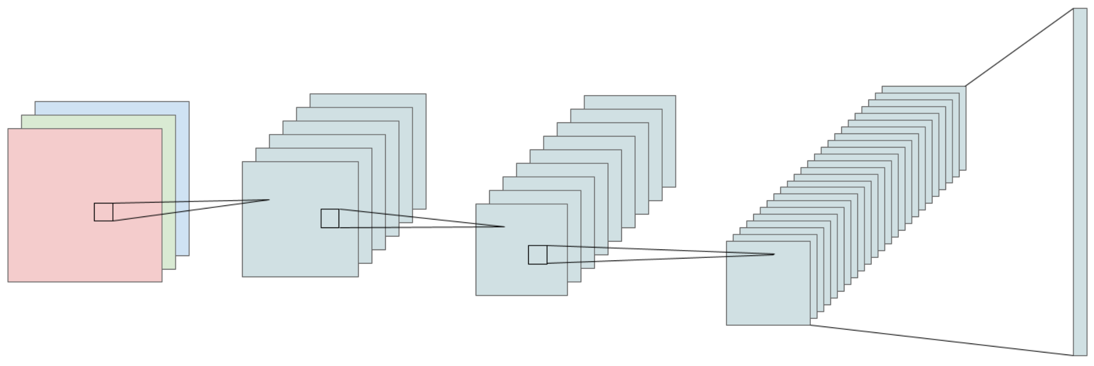

Figure 7 Illustration of a simple convolutional neural network structure. The leftmost layers represent the red, green and blue channels of the input image with subsequent convolutional layers and finally a final, flat output vector.

over pixels i, j such that

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{7}$$

The architecture of a CNN resembles that of a traditional neural network with an input layer, hidden layers and one output layer. Since the topology of the input is different however, the input is a tensor with the shape: width x height x number of channels. For example, a normal 64x64 color image would have the shape 64x64x3. There can also be intermediate layers between the convolutional layers called pooling layers that reduce the dimensionality of the data by combining the several outputs of the previous layers into a fewer amount of neurons.

In practice the 2D convolution is achieved by a matrix multiplication over the input with kernel sizes of around 3×3. Also, in a convolutional layer in a network these operations are called *filters* and there are most often more than one. Each of these filters produces a number of weighted sums by traversing the image with a set step length, called the *stride*. In this paper a convolutional layer is denoted by $K \times K \times F/S$ where $K \times K$ i the size of the kernel, F is the number of filters and S is the stride. An illustration of a standard CNN layer structure can be seen in figure 7.

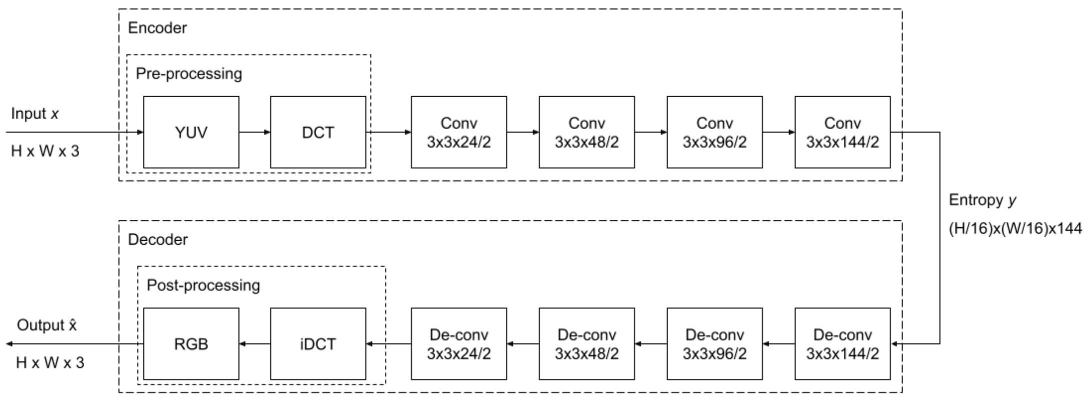

Figure 8 An overview of the autoencoder structure. The convention NxNxK/S denotes a convolutional layer with a kernel with size NxN, K filters and a stride of S. The pre-processing layers consists of transform layers that turns the signal into the YUV color space and DCT coefficients, with the post-processing layers mirroring this transform, turning it back into an RGB spatial signal.

4.6 Microsoft Azure

The model was trained on the Microsoft Azure platform, a cloud computing platform developed by Microsoft in 2008 [17]. The platform offers Software as a Service (SaaS) where usage is paid for by a subscribtion model.

5 Model structure

This section will explain the process of the autoencoder step by step. The model consists of two main parts: the encoder and the decoder. The process of passing the input through the encoder is called *analysis* and the reconstruction when passed through the decoder is called *synthesis*. There are also pre-processing done as part of the analysis, intermediate steps between the analysis and synthesis as well as post-processing during the synthesis.

The layer structure of the autoencoder was purposefully kept simple, both due to time constraints for testing advanced structures and to keep the focus on the effect of DCT. Inspiration was taken from previous work that employ around 5 layers with a stride of 2 and an increasing amount of filters. The structure of the autoencoder can be seen in figure 8.

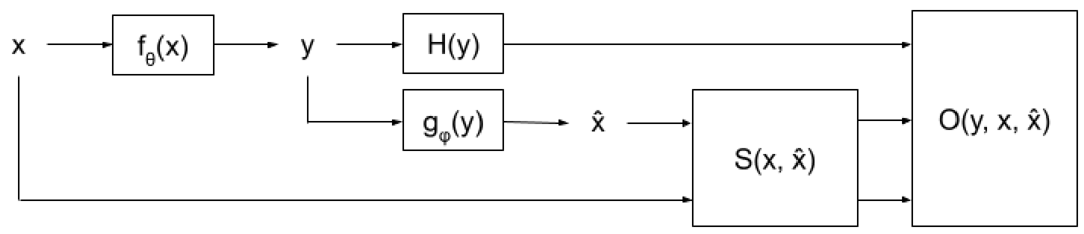

Figure 9 Overview of the different sub-processes of the model. The input signal x is first processed by the *analysis* function $f_\theta(x)$, producing the entropy y. During training, the entropy rate $H(y)$ is calculated in order to be used later in the objective function $O(y, x, \hat{x})$. The entropy y is also used by the *synthesis* function $g_\varphi(y)$ to produce the reconstructed signal $\hat{x}$. In order to measure the distortion of the process, i.e. the difference between the original signal x and the reconstructed $\hat{x}$ the two signal are used in the distortion measurement $S(x, \hat{x}) \in [0, 1]$ where $S(x, \hat{x}) = 1$ means that $g_\varphi(y)$ was able to perfectly reconstruct the image. The entropy rate $H(y)$ and distortion measure $S(x, \hat{x}) \in [0, 1]$ is then used in a weighted sum in the objective function $O(y, x, \hat{x})$.

5.1 Analysis

The analysis is denoted by a parameterised function $f_\theta(x)$ that takes input image x and outputs a entropy vector y such that $f_\theta(x) = y$. The goal of the analysis is not necessarily to compress the signal but rather produce y in such a way that is efficiently compressed, i.e. with as low an entropy rate as possible.

The input signal x is the RGB data from a $n \times n$ image which are processed by the first two layers of the encoder, named YUV and DCT in figure 8. By converting from the RGB color space the channels can be prioritised since humans are better at perceiving nuances in structure than in color, as discussed in section 2.3. More data can be discarded by the convolutional filters in the U and V channels, thus allowing for better compression.

Similarly, the DCT layer achieves also allows for more information to be discarded. However, instead of prioritising channels DCT makes it possible for the filters to discard some of the signals frequencies. Most of the information we perceive in an image is made up of larger features which was exploited when the original JPEG standard was developed, which also used DCT [22].

After the pre-processing layers the signal is passed through a number of convolutional layers, finally producing the entropy signal y.

5.2 Synthesis

The synthesis is denoted by a parameteterised function $g_\varphi(y)$ that takes input entropy y and outputs an estimate $\hat{x}$ of the original image x. The structure of the decoder is, broadley, a mirrored version of the encoder, starting by passing the signal through a number of convolutional layers, then transforming the signal back to the spatial domain through an inverse DCT layer and a YUV to RGB layer.

5.3 Objective function

There are two ways to evaluate the performance of the model: how much can the image be compressed (bitrate) and how well can the original image be reconstructed(distortion). Since the compression is lossy these two are in conflict with each other, the more the image is compressed the higher the distortion will be. Therefore, a balance must be struck between the amount of the entropy in vector y and the distortion produced by g_φ.

The distortion is measured by MS-SSIM, is denoted by $S(x, \hat{x}) \in [0, 1]$ and measures the difference between the two images with $S(x, \hat{x}) = 1 \Rightarrow x = \hat{x}$. MS-SSIM is further discussed in section 6.1.

The entropy of y is measured by

$$H(y) = -\sum_i p_i \cdot \log(p_i) \tag{8}$$

where p_i is the probability of value i in y, i.e. the rate of its occurrence in the array. By minimizing the entropy, y becomes more predictable and can thus be losslessly compressed more. Since this entropy rate estimation is continuous there is no problem in the backpropagation to find the derivative.

Altogether the objective function O can now be expressed as

$$O(y, x, \hat{x}) = \beta \cdot S(x, \hat{x}) - log(H(y)) \tag{9}$$

where β is used to control the trade-off between the quality of the reproduction and the compression rate.

5.4 Training & pre-processing

Development and training was done on the Microsoft Azure cloud platform. A computing instance with 6 cores, 56 GB RAM and a NVIDIA Tesla K80 GP. The dataset CLIC was used as training data during training and testing. This dataset was created for the Challenge on Learned Image Compression 2020 workshop which is an annual event which aims to gather publications in order to further the field of image and video compression [5]. It contains a mixture of RGB images captured by amateurs as well as professionals that are both in color and greyscale. Counting the test, training and validation subsets it consists of around 2000 images [18].

Much of the pre-processing as it pertains to data manipulation is done as part of the analysis process as discussed in section 5.1, however for practical purposes there some preparation had to be done on the data set before training. The images first had to be cropped to 128x128 squares in order to be used as input. This size is quite large but was found to be small enough for it to be feasible to train the model while also being large enough to discern the visual fidelity of the reconstructions. Since the CLIC dataset contains images that are far larger than 128x128 it was possible to split each image up into several smaller squares for use as training data.

6 Requirements and evaluation methods

Measuring image quality and similarity is fraught with difficulty. Simply measuring the difference between two versions of an image, e.g. an original and compressed image, can be done with some simple measurement such as the mean square error (MSE). However, as discussed in section 2.3 our perception of images is more nuanced than euclidean distances. We perceive structure in images and are more sensitive to brightness than color and there are more sophisticated measurement methods that take this into account which will be used in training of the autoencoder.

6.1 Multi-scale structural similarity index measure

The structural similarity index (SSIM) is a measurement of the distortion or perceived quality loss in an image caused by processing such as by compression or data loss during communication. SSIM is a full reference metric which means that it requires both the distorted and original image in order to measure the difference between the two. Rather than computing the absolute value of the distortion, SSIM takes a more sophisticated approach in order to measure the *perceived* change in image quality. The luminance l, contrast c and structure s between two image x and y of equal sizes are calculated by the formulas:

$$\begin{aligned} l(x,y) &= \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \\ c(x,y) &= \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \\ s(x,y) &= \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \end{aligned} \tag{10}$$

with:

- μ_x and μ_y: the average of x and y
- σ_x^2 and σ_y^2: the variance of x and y
- σ_{xy}: the covariance of x and y
- $c_1 = (K_1L)^2$, $c_2 = (K_2L)^2$ and $c_3 = c_2/2$
- $K_1 \ll 1$ and $K_2 \ll 1$: two constant scalars
- L: the dynamic range of the pixel, e.g. 255 for 8bits/pixel images

With the three components l, c and s we can express the SSIM as:

$$SSIM(x,y) = [l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma] \tag{11}$$

where α, β and γ define the relative importance of the three components and for $\alpha = \beta = \gamma = 1$ the formulas can be expressed as:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{12}$$

which satisfies three important conditions:

1. symmetry: $SSIM(x,y) = SSIM(y,x)$

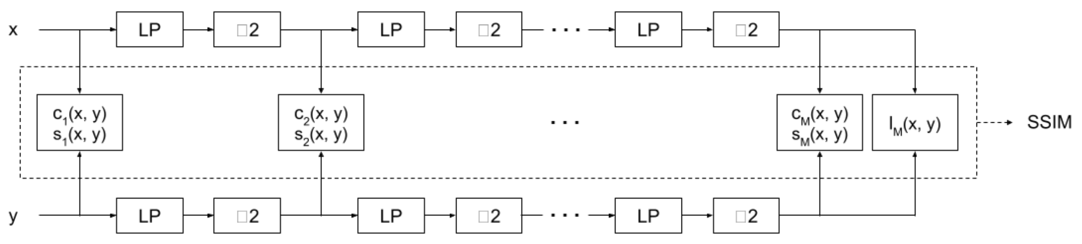

Figure 10 Illustration of the MS-SSIM process. The distortion is measured on several scales (1-M) by taking comparing the contrast c and structure s on each scale and finally the luminance l on the M-th scale. The overall SSIM value for the two images can then be calculated as a weighted product of the product of all c and s values and the final l value.

2. boundedness: $SSIM(x, y) \leq 1$
3. unique maximum: $SSIM(x, y) = 1$ if and only if $x = y$

Multi-scale SSIM (MS-SSIM) performs this measurements on several scales 1-M. For each scale contrast and structure is calculated then the signal is sent through a low-pass filter and is downsampled by 2 before it's sent to the next scale. The luminance is calculated at the final (M-th) scale and is combined with all other values in weighted sum to produce a final value [23]:

$$SSIM(x, y) = \left[l(x, y)^{\alpha_M} \cdot \prod_{j=1}^{M} \left(c_j(x, y)^{\beta_j} \cdot s_j(x, y)^{\gamma_j} \right) \right] \tag{13}$$

The

7 Results and discussion

This section will go through and discuss the results from training and testing the model and observations that were made during development. Machine learning is a time consuming and iterative process since training the model takes hours, days or even weeks. By outputting the current loss it's possible to monitor performance of the current build, however if the loss function is altered it becomes difficult to compare subsequent runs. Consequently, and also due to the limited time scope of this thesis, training time was limited to 8 hours in order to be able to observe major or minor changes to the model between training runs.

7.1 General observations

Following the first training session it became clear that the use of DCT in conjunction with the autoencoder was feasible. Despite limited training time and sub-optimal model structure the model was able to reproduce the image in a recognisable fashion. Idiosyncratic traits emerged, some of which persisted throughout the project and some that dissipated with more training data.

The most obvious trait was the models poor ability to recreate color. Very bright parts of the image was almost always colored magenta, green or teal as seen in figure 11. Since the pixel value of white is 1 for the red, green and blue channels this discoloring is might be due to difficulties in converting white between the different color spaces of YUV and RGB. Simple conversion between the two color spaces does not pose a problem, however they are the very first and last steps of the entire process, lending credence to the possibility that too much information is sometimes lost in the intermediate steps to reconstruct white.

Another trait was the difference in ability to recreate large and small features. This was expected since the reason DCT was chosen was its ability to help discard smaller frequencies in an effective way. For images consisting of large patches of sky or solid walls it was difficult to discern between the original and the recreation, however for images with leaves or some other irregular pattern it was quite clear which image was the recreation. Whenever the intricate patterns followed were more regular the model handled it better, however due to the constant ebb and flow between visual fidelity and bitrate as well as lower frequencies being more difficult to recreate this trait consisted throughout the project and would require more training to solve.

7.2 Results

In this project, the model was not able to compress images better than existing standards such as JPEG. Granted, attempts to train the model to compress down to ratios of 1/16 was thwarted by limitations in training time and data, however as seen in figure 13 the model was not able to reproduce color accurately and a substantial amount of detail was lost.

Lower compression rates were more successful, as seen in figure 12 where images was

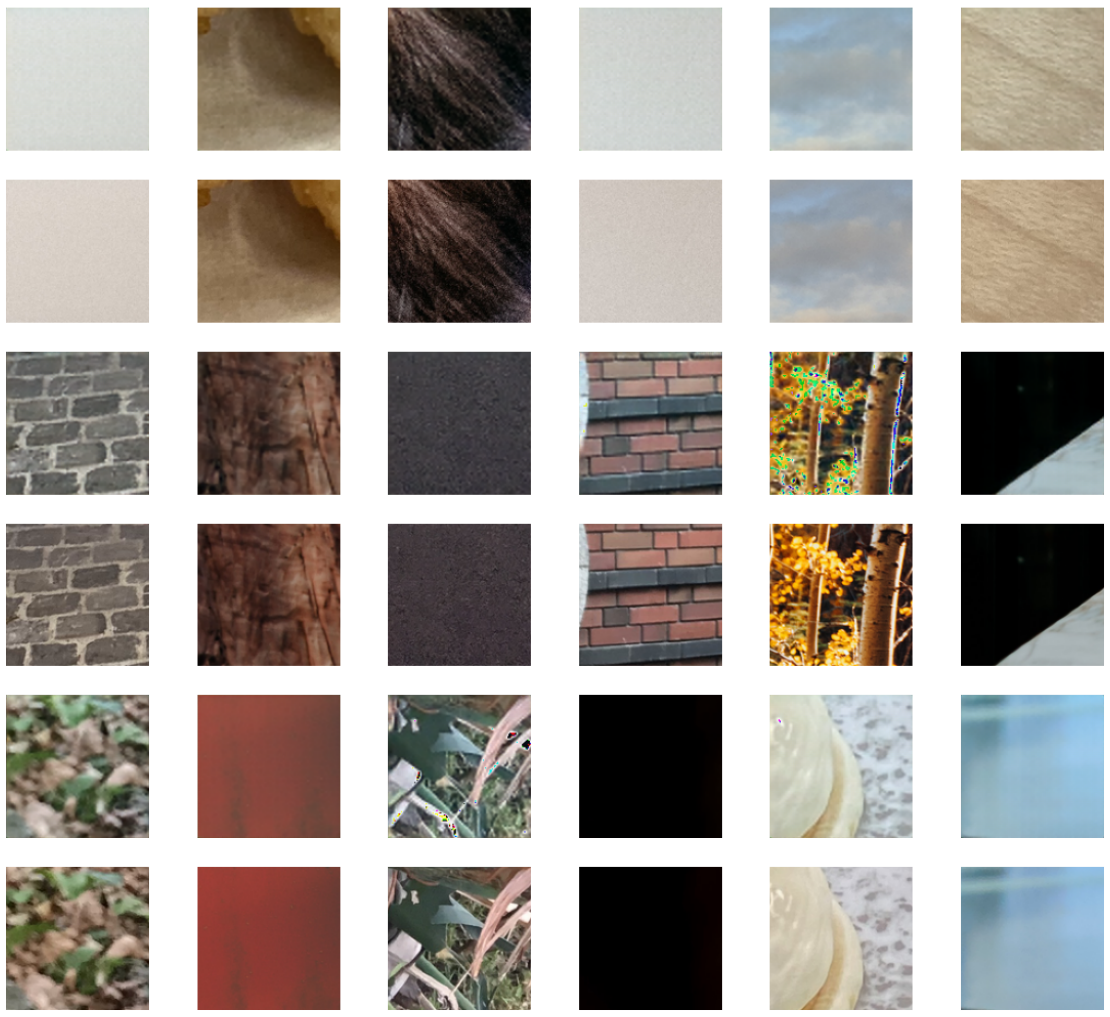

Figure 11 Early version of the model that demonstrates the issue with recreating very bright parts of an image. The reconstructions are overall satisfactory apart from the brightness in the leaves in the second image from the right in the middle row. This version of the model was not designed to compress the image, rather it was designed to test if the autoencoder and it convolutional layers would work in conjunction with DCT.

compressed by a factor 1/2. The model was able to reproduce certain images well enough to be indistinguishable from the original, however it once again had difficulties reproducing correct colors. The model was inconsistent when recreating very bright color, with one example of an almost completely white image being rendered blue and cyan while another very similar image (albeit with a slight blue tint) was recreated far more accurately.

8 Conclusion

The main question posed in this thesis is whether or not it is feasible to use the discrete cosine transform in conjunction with an autoencoder in order to compress and reconstruct images. It was clear from an early stage that transforming the image signal, passing it through some simple convolutional layers and then reconstructing worked quite well with recognisable images albeit with some discrepancies as seen in figure 11. The DCT behaved in large parts as expected, handling larger frequency features such as patches of sky better than lower frequency features such as patterns on a rug.

No version of the model developed during this project yielded results that exceeded existing standards. Compression rates at or below a factor 1/2 yielded reconstructions where color information was lost. However, the image was not entirely unrecognisable and with more training data and time it is very likely performance would improve.

In conclusion, the discrete cosine transform yielded promising results when used in a compressing autoencoder. Thorough comparisons between layer structures and other existing image compression standards fall beyond the scope of this project but preliminary results show that a very simple structure is enough for satisfactory reconstructions.

9 Future work

In this section I will discuss what work the model would need in order to get better in the future. Any machine learning endeavour will be inherently time consuming due to the long training times and experimental nature of testing different layer architectures. Therefor there are quite a few areas that can be expanded or improved upon such as longer training time and implementing a proper image compression library using the autoencoder.

A conscious decision was made in the beginning of this project to keep training time to a minimum in order to be able to make revisions to the model at reasonable intervals.

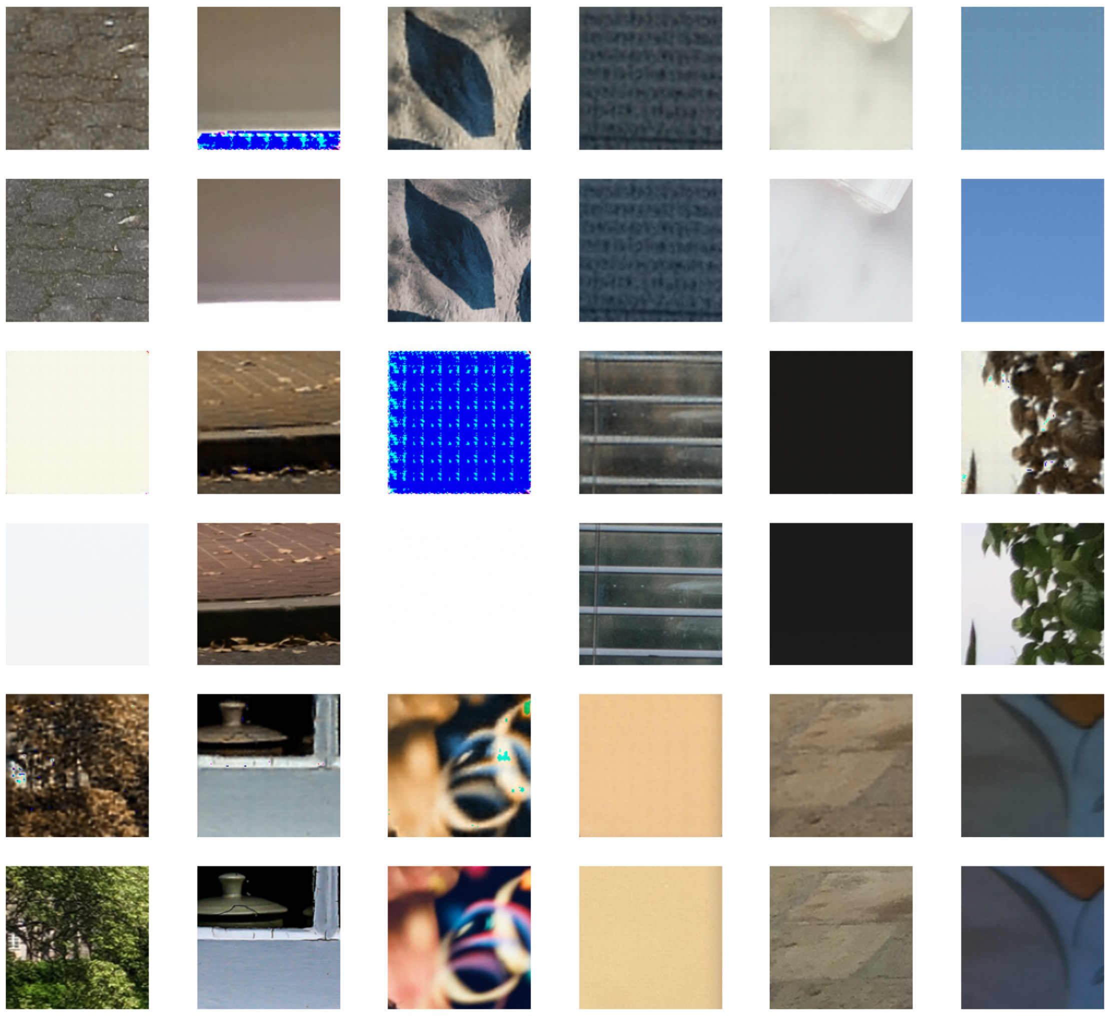

Figure 12 Result of model training to compress by a factor of 1/2, trained over approximately 6 hours. Each sample is taken randomly from the test set, with the lower example being the original and the ones on top being the reconstruction. The model is overall able to recreate the image quite successfully with some obvious exceptions. Very bright parts of an image where pixel values are very close to pure white there is a tendency to reconstruct very poorly, often into some extreme of the RGB color spectrum such as magenta or blue.

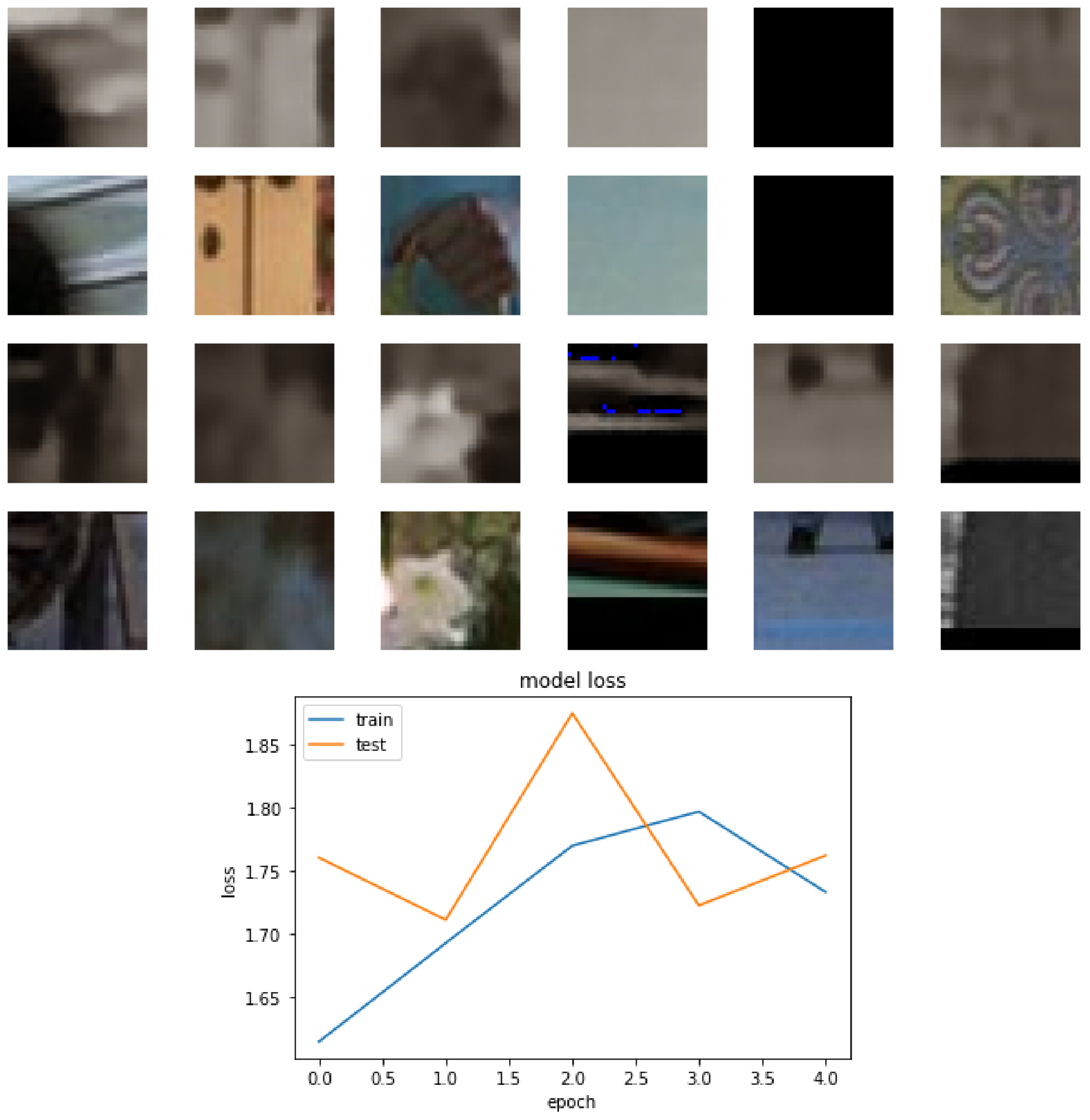

Figure 13 Top: result of model training to compress by a factor of 1/16. Bottom: model loss during training. The model was trained for approximately 2 hours. Each sample is taken randomly from the test set, with the lower example being the original and the ones on top being the reconstruction. The short training time is due to the technique of ’early stopping’, where the model will suspend training if the loss has not improved for a set number of epochs. This is clearly visible in the chart where the train loss only increases after the first epoch. Substantial distortion results in almost all color information being lost for all examples, as well as barely discernible details.

Nevertheless, the model was going to have to be trained which would require time and so after some testing a training time of around 8 hours was discovered to give a good balance between a well trained model while also enabling daily adjustments.

While this time limit enabled faster testing of minor changes, it came with the concession of not training over truly large datasets. For this thesis, where the goal was to determine if it is *feasible* to use the DCT to enhance the performance of the compressing autoencoder, running on very large datasets would be redundant. However, in order to determine what performance gains the DCT would entail, the model would require longer training. Images with poor lighting, of different subjects from all angles or with sharp contrast between features would allow for the model to be more flexible and be able to better recreate a wider range of images.

Further evaluation is also needed to compare the model to existing standards and results. Current results show that the model is not quite able to exceed the compression rates of standards such as JPEG while retaining the same visual fidelity. Once the model would perform well enough there would also need to be a user study in order to compare the two methods not only by metrics but also by real people.

The implementation of the model was kept simple in order to test the original hypothesis regarding DCT. In order for the autoencoder to be used in practice several other components would have to be developed since the images would be split into smaller chunks, each analysed separately then all combined in post processing. This project focused on the core component of the autoencoder and the underlying techniques that enables the compression, however there is a strong possibility that this would also need to be adjusted in order to be used in practice.

References

[1] C. AB. Consid ab about us. Hämtad 2021-01-29. [Online]. Tillgänglig: https://consid.se/en/about-us/

[2] N. Ahmed, T. Natarajan, and K. R. Rao, “Discrete cosine transform,” *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, 1974.

[3] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[4] J. Ballé, V. Laparra, and E. P. Simoncelli, “End-to-end optimization of nonlinear transform codes for perceptual quality,” in *2016 Picture Coding Symposium (PCS)*, 2016, pp. 1–5.

[5] CLIC. Clic - challenge on learned image compression. Hämtad 2021-04-19. [Online]. Tillgänglig: http://compression.cc/

[6] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex fourier series,” *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[7] J. Fourier, *Théorie analytique de la chaleur*. Chez Firmin Didot, père et fils, 1822. [Online]. Tillgänglig: https://books.google.se/books?id=TDQJAAAAIAAJ

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[9] D. A. Huffman, “A method for the construction of minimum-redundancy codes,” *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[10] A. Kaplan and M. Haenlein, “Siri, siri, in my hand: Who’s the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence,” *Business Horizons*, vol. 62, no. 1, pp. 15–25, 2019. [Online]. Tillgänglig: https://www.sciencedirect.com/science/article/pii/S0007681318301393

[11] A. Karami, S. Beheshti, and M. Yazdi, “Hyperspectral image compression using 3d discrete cosine transform and support vector machine learning,” in *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, 2012, pp. 809–812.

[12] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, “Supervised machine learning: A review of classification techniques,” *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[13] NIST. (2007, May) Fiftieth anniversary of first digital image marked. Hämtad 2021-01-27. [Online]. Tillgänglig: https://www.nist.gov/news-events/news/2007/05/fiftieth-anniversary-first-digital-image-marked

[14] C. Párraga, T. Troscianko, and D. Tolhurst, "Spatiochromatic properties of natural images and human vision," *Current Biology*, vol. 12, no. 6, pp. 483 – 487, 2002. [Online]. Tillgänglig: http://www.sciencedirect.com/science/article/pii/S0960982202007182

[15] A. Riesman. (2012) Crossdressing, compression, and colliders: 'the first photo on the web'. Hämtad 2021-04-13. [Online]. Tillgänglig: https://www.vice.com/en/article/ezzza7/crossdressing-compression-and-colliders-the-first-photo-on-the-web?utm_source=reddit.com

[16] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 2922–2930. [Online]. Tillgänglig: http://proceedings.mlr.press/v70/rippel17a.html

[17] A. Srivastava. Introducing windows azure. Hämtad 2021-05-10. [Online]. Tillgänglig: https://archive.is/M0vvP

[18] TensorFlow. Clic. Hämtad 2021-04-19. [Online]. Tillgänglig: https://www.tensorflow.org/datasets/catalog/clic

[19] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," *arXiv preprint arXiv:1703.00395*, 2017.

[20] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," *arXiv preprint arXiv:1511.06085*, 2015.

[21] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5306–5314.

[22] G. K. Wallace, "The jpeg still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[23] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2. Ieee, 2003, pp. 1398–1402.

[24] X. Zou, X. Xu, C. Qing, and X. Xing, "High speed deep networks based on discrete cosine transformation," in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 5921–5925.