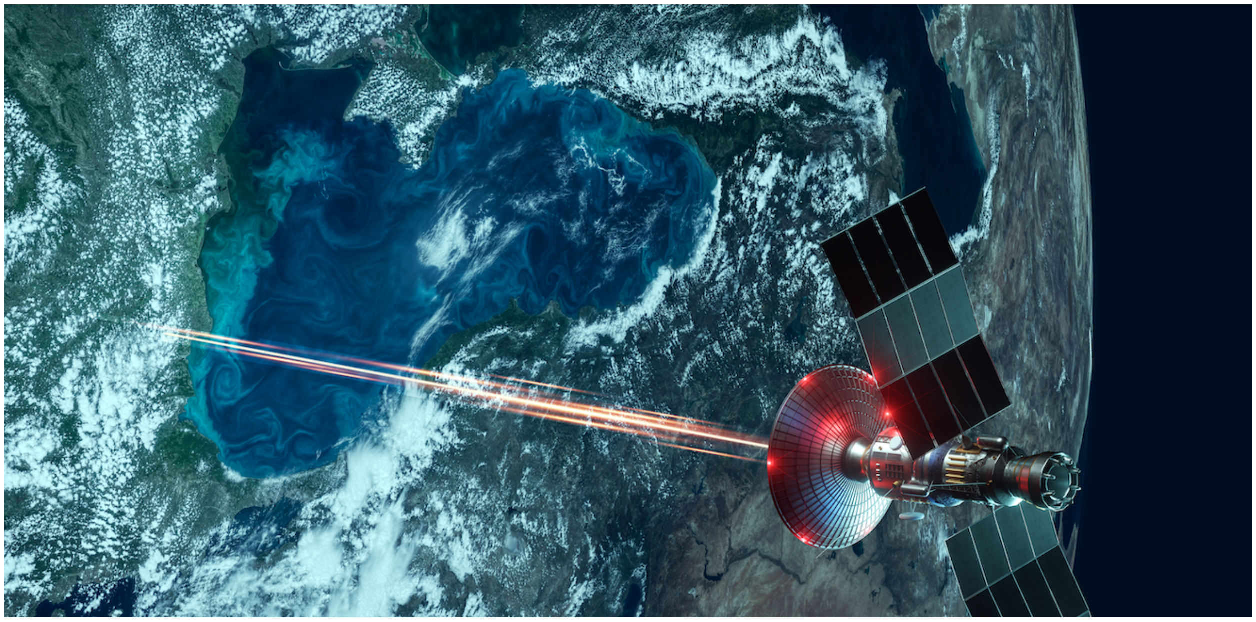


Evaluation and implementation of an auto-encoder for compression of satellite images in the ScOSA project

Mohamed Sobhy Abdelsalam Fouda

Master Thesis

Matriculation: 407201

August 22, 2022

MSE-22-004

Supervisors:

Prof. Dr.-Ing Enrico Stoll

Dr.-Ing Zizung Yoon

Dr.-ing David Krutz

Ing. Karsten Westerdorf

Table of content

Abstract

Satellite imagery is one of the most efficient methods to retrieve an accurate visual representation of the landscape. From these considerations, satellite imagery has found immense popularity in cartography, urban planning, agriculture, emergency response, and climate change studies. However, comprehensive satellite imagery of the highest quality is also associated with the requirement of extensive data storage capacity and upload speed issues. One of the methods to mitigate these limitations is utilizing contemporary image compression techniques. For instance, raw satellite imagery is generally uploaded in the compressed format of NITF (National Imagery Transmission Format) to increase the overall communication speed. Therefore, it is essential to continually progress the technological advancement in this area to resolve the problem of data storage and communication. This thesis is part of ScOSA (Scalable On-Board Computing for Space Avionics), the DLR (German Aerospace Center) research project dealing with on-board computers as a distributed system, which will be part of a DLR CubeSat mission. The thesis evaluates the efficiency of various autoencoder neural networks for image compression regarding satellite imagery. The results highlight the evaluation and implementation of autoencoder architectures and the procedures required to deploy neural networks to reliable embedded devices. The developed autoencoders evaluated, targeting a ZYNQ 7020 FPGA (Field Programmable Gate Array) and a ZU7EV FPGA. The outcome appears in the successful implementation of an autoencoder for compressing infrared and hyperspectral images, with 64 and 21 compression factors, respectively, compared to classical compression techniques. The overall outcome showed promising results for the integration in the ScOSA project.

Keywords: satellite, imagery, compression, neural, networks, autoencoders

Acknowledgement

Foremost, I would like to express my sincere gratitude to my advisors Prof. Enrico Stoll and Dr. Zizung Yoon, for the continuous support of my master's study and research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. Further, I would like to thank my thesis supervisors from the DLR: Dr. David Krutz, Eng. Karsten Westerdorff and all the colleagues who supported me, for their insightful comments and continuous support.

Special thanks to Eng. Cem Avsar, for his encouragement and continues feedback to support me. Moreover, thanks to the MSE team for their continued support.

My sincere thanks also goes to the “Stiftung Rotary Club in Berlin” and the German scholarship team from the career service group in TU Berlin, for offering me the scholarship opportunity, which helped me work on diverse of exciting projects.

Last but not least, I would like to thank my family: my parents, sisters, wife and son, for supporting me spiritually throughout my life.

1 Introduction

This research evaluates hiring neural image compression using autoencoders as software and hardware in the loop (SIL, HIL) to the satellite imagery industry. This thesis is part of ScOSA (Scalable On-Board Computing for Space Avionics), the DLR (German Aerospace Center) research project dealing with on-board computers as a distributed system, which will be part of a DLR CubeSat mission [105]. The thesis evaluates the efficiency of various autoencoder architectures for image compression. As a result of this thesis, the implementation, and evaluation analysis of an autoencoder will be presented. Further, this research discusses data compression neural networks and the required procedures towards deployment on AI-Ready on-board computing platforms.

Overcoming the trade-off between reliability and performance is a current research concern in space engineering. Thus, the German Aerospace Center (DLR) began to investigate a solution to this issue in 2012, then became the On-Board Computer-Next Generation (OBC-NG) project in 2013, continuing in 2017 with the ScOSA project [104]. The outcome of those two projects is an on-board computer architecture offering reliable and high-performance radiation-hardened space-qualified hardware. ScOSA architecture consists of independently interconnected nodes, which consist of either reliable components or high-performance commercial of the self (COTS) components such as the field programmable gate array (FPGA). Each node instantiates its own operating system, which is abstracted to a monolithic execution platform by a middleware. The system supports running multiple concurrent applications. The developer divides the applications into channels and tasks using the provided interface to the execution platform. The middleware handles the reconfigurability of the system during different mission phases, as well as node failure. Additionally, the middleware provides standard services well known in the research area of fault detection, isolation, and recovery (FDIR). The FDIR services incorporate a voter service that enforces a Triple Modular Redundancy and checkpointing service for a distributed state restoration in case of a task or node failure [141].

Furthermore, the middleware is qualified to operate on a heterogeneous cluster of processing nodes and correspondingly encloses a heterogeneous network architecture consisting of either Ethernet, SpaceWire, or both [142]. Also, the middleware is capable of supporting different operating systems. So far, Real-Time Executive for Multiprocessor Systems (RTEMS) and Linux are supported to run on the different nodes. As a successor to the ScOSA project, ScOSA Flight Experiment, began in January 2020. This project aims to conduct a higher Technical Readiness Level (TRL) by preparing the in-orbit demonstration of the developed ScOSA on-board computer.

This thesis is an execution of one objective of the ScOSA project, which is to evaluate, if neural networks can be an alternative to the classical compression algorithm. Current ScOSA advancements require 20W for four nodes, and the maximum neural image compression is required to run on two nodes. The evaluation shall discuss the implementation on the technology-experiment ScOSA as an example system. The experiment is set to conditions of 1 Hz frame rate.

1.1 Motivation

The satellite industry is of great importance to various industries including military, meteorology, safety, climate and environmental monitoring, and landscape mapping. The current trends and advancements in the industries using satellites have proliferated in the recent past, paving the way for new inventions [9,73]. With the use of satellites to capture ground images for landscaping, meteorological or intelligence purposes has significantly grown. Competition in the industry has also demanded unprecedented research in image processing technologies to better retrieve information from images. Satellites mainly capture electromagnetic radiation reflected by the earth's surface [25]. The radiation is emitted by the sun and reflected by the sun; hence, the sensors do not require energy to operate [11].

At present, satellite imagery is one of the most efficient methods to retrieve an accurate visual representation of the landscape. From these considerations, satellite imagery has found immense popularity in cartography, urban planning, agriculture, emergency response, and the studies of climate change [92]. However, comprehensive satellite imagery of the highest quality is also associated with the requirement of extensive data storage capacity and issues of upload speed [94]. One of the methods to mitigate these limitations is by utilizing contemporary techniques of image compression [84]. For instance, raw satellite imagery is generally uploaded in the format of NITF (National Imagery

Transmission Format) to increase the overall communication speed [64,71,74]. Therefore, it is essential to continually progress the technological advancement in this area to resolve the problem of data storage and communication. The thesis discusses the efficiency of various autoencoders and image compression frameworks in regard to satellite imagery.

Autoencoders are comprised of three main parts: an encoder, a bottleneck, and the decoder. The encoder is mainly tasked with compressing the input data, usually a satellite image, into an output that is several times smaller than the input [34]. The bottleneck is the most essential part of an autoencoder, as it houses the knowledge base of the compressed versions of the[input [56]. Its contents are usually compressed and represent whatever is known about a particular representation, mainly, knowledge about the input data. The decoder unzips the compressed data and converts it back to its original format. The output is essential in that it is compared to ground truth for accuracy evaluations [46,52,56]. Hence, the encoder, bottleneck, and decoder must be effective to preserve the information portrayed in the images.

There exist different technologies, tools, and approaches used in studying satellite images. Images captured over different times present a clear change in natural and artificial developments such as soil erosion, changes in levels of carbon emission, infrastructural development, or waste build-up. Irrespective of the study approach employed, there are some static factors that must be considered in studying satellite images [38]. They include scale, patterns, shapes and textures, and colors. The results obtained are always compared to prior knowledge to better understand the imagery [25]. For instance, establishing water bodies such as lakes, rivers, or oceans are primarily dependent on their color and shapes. True color satellite images use visible light wavelengths: red, green, and blue. This implies that the images are similar to what a normal human eye would see from space. In true-color images, features appear in a detailed manner and are easy to decipher and understand. However, images that use false colors may include unanticipated colors which might be difficult to understand from a natural perspective.

The study and interpretation of satellite images have over the years grown with the development of new technologies to model, interpret and understand different details presented therein. Modern sensors capture images with high resolution, translating to large sizes which might be challenging to store or transmit [38]. As a result, image compression has been widely adopted to reduce the image byte size while retaining the image quality at acceptable qualities [5]. The image size reduction allows the files to be stored, transmitted, or processed within confined computing environments while retaining the detailed information contained in the images [4,6].

1.2 Problem Statement

Modern satellite sensors capture high-resolution images which might be challenging to store, process, or transmit. The high-resolution images are large in size (total number of bytes). Such images need to be compressed for processing, storage, or transmission as required. However, even though there exist different file compression technologies, the aftermath quality of the image must be retained [9]. The quality of decompressed images is a necessary factor and helps regain the detailed information presented in the images, while also testing the efficiency of the compression tools and approaches. Satellite imagery remains the most effective way of capturing and representing information on the landscape. The advancement of satellite image-capturing sensors, as mentioned above, results in large-sized files. The advantages of existing image compression technologies must be harnessed to improve them or produce better edge-cutting technologies in the future. On the other hand, the cons must also be mastered to avoid their impact on image compression and decompression.

1.3 Objectives

The main objective of this research is to study the adaption of neural image compression technologies in satellite imagery and the use cases of deep autoencoders. Other objectives include

- Identify different neural image compression techniques.
- Evaluate neural image compression state-of-the-art on satellite images.
- Implement an auto-encoder for satellite image compression.
- Validate the produced neural network model on random satellite images.
- Discussion on neural model migration to embedded devices.
- Recommend scenarios of possible integration with the ScOSA project.

1.4 Structure

The thesis consists of six main chapters: introduction, background, literature review, methodology, results and analysis, conclusion and future work. The first chapter presents an introductory overview of image compression and the trends in the number and sizes of satellite images in recent years. The chapter also presents the study objectives, which guide the background, literature review, methodology, and analysis sections. The second chapter is the background, which presents the theoretical background of the classical and neural image compression techniques. The third chapter is the literature review and presents previous works on image capturing, compression, storage, processing, and decompression. The methodology chapter presents the research philosophy, approach, and the different approaches employed therein. It also presents the data collection techniques used. The results and analysis evaluation chapter presented the techniques used to evaluate the performance of the autoencoders compared to different image compression technologies. It also paves the way for the conclusion, future work and recommendations.

2 Background

As the number of satellites capturing earth images increased, the size of datasets also increased exponentially. The quality of satellite images has improved, corresponding to the creation of larger file sizes. The resolution of satellite imagery has also improved from several meters to just a few inches, indicating the images contain finer details than ever before [13,72]. Usually, image information is represented in pixels whose number is inversely proportional to spatial resolution. Each pixel represents a color in the RGB scale. The more detailed an image is, the bigger the file size [14,74,76]. as a result, the computing resources required to process such images are high and might not be available in most computing scenarios.

Without compression, a 1024 pixel x 1024 pixel x 24-bit image would take 3 MB of storage and 7 minutes to transmit over a high-speed, 64 Kbit/s ISDN connection. When a picture is compressed at a 10:1 compression ratio, the storage demand is lowered to 300 KB and the transmission time is decreased to less than 6 seconds. Seven 1 MB photos can be compressed and transmitted on a floppy disk in less time than it takes to deliver an uncompressed version of one of the original files over an AppleTalk network [3].

Large picture files continue to be a key bottleneck within systems in a distributed setting. Compression is a key component of the methods available for establishing file sizes that are manageable and transmittable. Broadening the bandwidth is also another solution, but it is very expensive. The mobility and performance of the platform are significant considerations when choosing a compression/decompression approach. The simplest technique to minimize the size of the picture file is to shrink the image directly. By reducing the image size, fewer pixels must be saved, and as a result, the file will load faster [3,6].

Image compression uses different standards and algorithms to reduce the actual size of images without affecting their quality. Lossy and lossless compression is the main digital compression technique [78,81]. As the name suggests, the lossless compression technique does not result in any loss in quality. This implies that the complete imagery details can be re-obtained after digital decompression. Although the technique may not be applicable in all scenarios, it is essential in instances where quality and accuracy are a priority [80,83]. On the other hand, the lossy compression technique produces an almost negligible loss in image quality [82]. The loss is usually minute and very hard to identify. The technique may not have any major impact on photographs but could have dire consequences if applied in detailed imagery such as satellite images. This is a result of the sensitivity of the detailed information contained in satellite images.

2.1 Image Compression Techniques

An image is usually a two-dimensional signal that has been processed by the human visual system. Images are often represented by analog impulses. However, they are changed from analog to digital form for processing, transmission, and storage in digital computers computer programs. A digital image is nothing more than a two-dimensional arrangement of pixels. Images account for a large portion of data, notably in video conferencing applications, remote sensing, healthcare. As human dependence on information and computers intensifies, so does the need for efficient ways to store and share enormous volumes of data.

Image compression seeks to reduce the data size necessary to represent an image on a digital modern media such as magnetic hard disk, optical drives, or solid-state media. It is a method that produces a solid rendering of a file, lowering image transmission and storage needs. Compression is accomplished by removing one or all of the following redundancies:

- Inter-pixel
- Psychovisual or,
- Coding redundancy.

There are two main image compression techniques: lossless and lossy approaches. The classification of the compression technique depends on how the compressed version of the digital image is compared to the original file. If there is a deviation in the quality of data representation, the technique is called lossy, otherwise, lossless.

2.2 Lossy Compression Techniques

Lossy techniques outperform lossless approaches when it comes to compression ratios. Lossy techniques are extensively employed because the decompressed image quality is suitable for most scenarios. The reconstructed image is not the same as the original image, but the difference is hard to tell or observe.

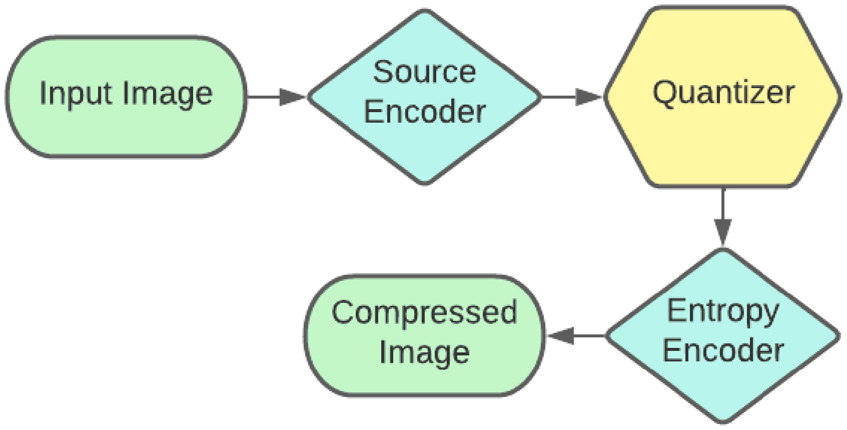


Figure 1. Lossy Image compression technique [Source: Author]

The outline of Lossy compression algorithms is illustrated above. The steps of the prediction – transformation – deconstruction process can be successfully reversed. The quantization process causes minor loss of information. The entropy coding performed after the quantization, on the other hand, does not trigger any losses. Decoding is a backward process that reproduces the original. The compression begins by decoding the entropy on the compression image to obtain the quantized data. The inverse transformation of the image is preceded by dequantization in order to decompress the image. The performance of the Lossy compression technique is influenced by the signal-to-noise ratio, compression ratio as well as encoding and decoding speed. The different Lossy compression schemes are discussed below.

2.2.1 Transformation coding

Discrete Fourier Transform, also known as DFT and Discrete Cosine Transform or, DCT are used to compress images. The schemes transform the pixels of an image to transform coefficient, also known as frequency-domain coefficients [11,88]. These coefficients possess several beneficial qualities, such as energy compaction property. The property ensures just a few compression coefficients are applied in compressing the entire compression while achieving the desired results [15, 84, 88]. This means the most important compression coefficients are kept and used, while the rest are discarded. The coefficients are then used in the decompression process.

2.2.2 Vector quantization

The main concept behind this method is to create a dictionary of fixed-size vectors known as code vectors. A vector is often comprised of several pixel values. An image is usually divided into non-overlapping vectors or blocks known as image vectors. The index of each image vector is used to compress the original image [90]. It implies that every image is replaced by indices that can further be encoded and compressed.

2.2.3 Fractal coding

The technique breaks down an image using a convectional processing approach such as edge detection, texture and spectrum analysis, and color separation [94]. A fractal library contains details of all image sections. The library also contains Iterated Function System (IFS) codes containing compacted integers [100]. The schematic approach is employed to determine the codes for a given image such that, with the application of the IFS codes, the resulting image is a close approximation of the original image. The technique is best suited for images with a high self-similarity index [99].

2.2.4 Block Truncation Coding

The technique segments an image into different pixel blocks. The reconstruction and threshold values for each block are usually specified. The mean value for the values in each pixel block is the threshold for that particular block. To encode the image, the values of the block that are more than the threshold are replaced with one, while those that are less than or equal to the threshold are replaced with a zero. The result is called a bitmap and can be reconstructed by reversing the process.

2.2.5 Subband coding

The image is examined in this technique to yield components comprising frequencies in well-defined bands, known as sub-bands. Consequently, coding and quantization are done to each of the bands. The benefit of this method is that the coding and quantization for each of the sub-bands can be achieved independently.

2.3 Lossless Compression Techniques

The original image may be perfectly retrieved from the compressed (encoded) image using lossless compression techniques. The techniques are also known as noiseless because they do not introduce noise into the transmission of the image. It is also referred to as entropy coding because it eliminates redundancy using decomposition techniques. The compression technique is only employed in a few critical applications such as medicine and intelligence. The techniques include

2.3.1 Run-length encoding

This is a straightforward compression technique for linear data. It comes in handy when dealing with repetitious data. This approach uses shorter symbols to replace runs of similar symbols (pixels). A grayscale image's run-length code is represented by the sequence Vi, Ri, where Vi is the pixel intensity and Ri is the number of consecutive pixels associated with the intensity Vi. The relationship is illustrated in figure 3. When both Vi and Ri are represented by one byte, a span of 12 pixels can be coded in 8 bytes, producing a 1:5 compression ratio.

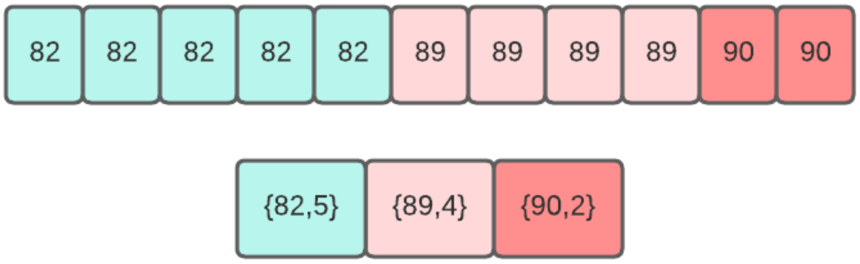


Figure 2. Run-length encoding [Source: Author]

2.3.2 Huffman encoding

This is a method for compressing data symbols with respect to their statistical probabilities. The image's pixels are viewed, treated, and processed as symbols. The number of bits assigned to symbols is inversely proportional to the frequency. That is the higher the frequency, the smaller the number of buts. The prefix code in this case is the Huffman code. This indicates that any symbol's binary code cannot be the prefix of any other symbol's code. Lossy compression techniques are mainly used in the early stages of image encoding, while Huffman coding is used as the last step in most image coding standards.

2.3.3 LZW coding

Lempel-Ziv-Welch (LZW) uses dictionaries to compress image data. The use of dictionaries for coding might be either static or dynamic. The dictionary is fixed during the encoding and decoding stages in static dictionary coding. The dictionary is updated on the fly in dynamic dictionary coding. LZW is a commonly used compression algorithm in the computer industry, and it is implemented as a UNIX compress command.

2.3.4 Area coding

It is a more advanced version of run-length coding that takes into account the two-dimensional nature of pictures. This is a big step forward from the previous lossless approaches. It makes little sense to code a picture as a sequential stream because it is an array of sequences that make up a two-dimensional entity. Area coding methods look for rectangular areas with similar properties. These areas are coded descriptively as a two-pointed element with a specific structure. This sort of coding is very efficient, but it has the drawback of being a nonlinear approach that can't be implemented in hardware. As a result, the compression time performance is not as aggressive as the compression ratio.

2.4 Neural Image Compression

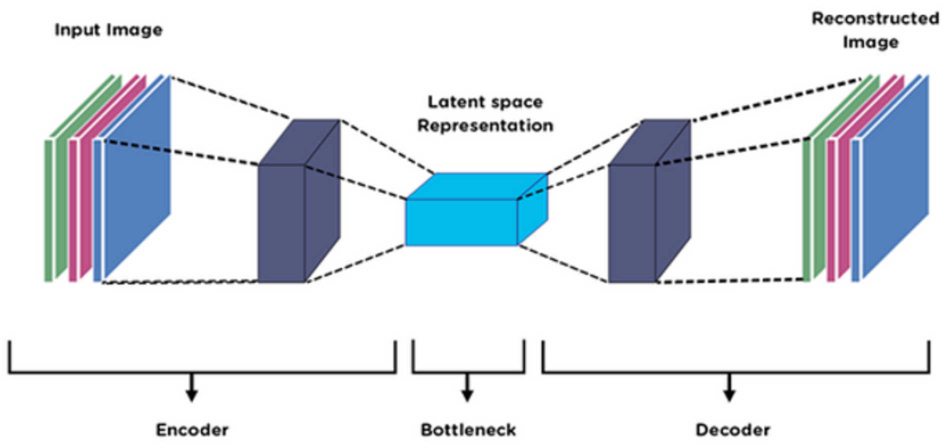


Figure 3. Autoencoder illustration for typical RGB Image compression [Source: Proceedings Press [108]

Compression is the process of shrinking an image's pixels, color components, or dimensions in order to minimize its overall file size. It decreases the amount of data they have to store and process [47, 124]. Advanced algorithms for image optimization can detect the most relevant visual elements while ignoring the less relevant ones. Image compression is generally administered by reducing spatial redundancy in the visual data and consequent reconstruction of the image. In general, an autoencoder refers to a type of neural network, which transforms the input into a code (or 'bottleneck') and, consequently, reconstructs it into a finished product [33]. At present, this framework is utilized in various industries for image compression, image classification, anomaly detection, and many more [1].

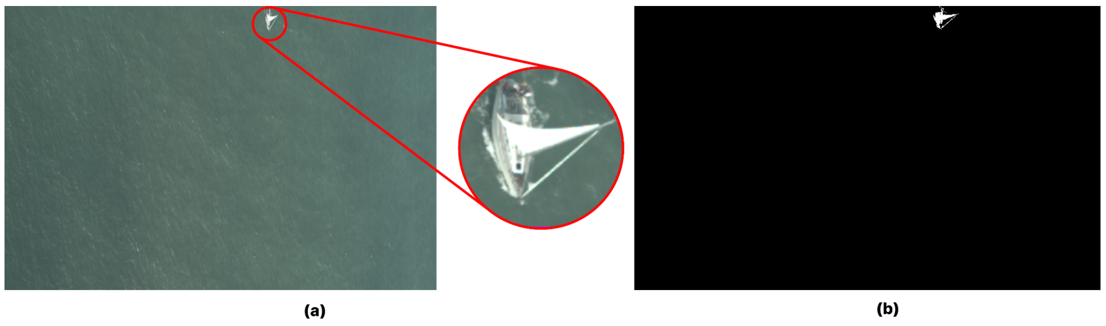


Figure 4. Helgoland island, North Sea, Anomaly detection autoencoder illustration. a) original, b) decoded [Source: DLR, MACS]

Furthermore, autoencoders can be used for both unsupervised (no classified data), semi-supervised and supervised analysis [20]. Another advantage of the framework, compared to traditional algorithms of image compression, is the high accuracy of the outputs and similarity to other computer-vision models [65]. As a result, it is possible to transform the existing networks of anomaly detection or image classification into image compression, which makes it more convenient for satellite communication [70]. Ultimately, compared to traditional algorithms, neural image compression is a highly effective framework and is applicable to satellite imagery.

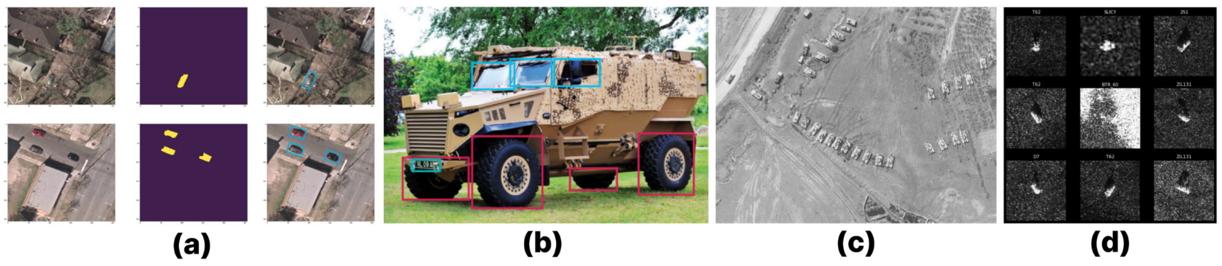


Figure 5. object segmentation (a), detection and decomposition (b&c) and classification (d) illustration [Source: gov.uk, FSA, DARPA [119,129,130]]

Figure 6 illustrate the supervised, semi-supervised and unsupervised learning of the nural networks. Figure (6a) and (6d) illustrate object segmentation and classification trained on datasets such as Moving and Stationary Target Acquisition and Recognition (MSTAR) from the U.S. Defense Advanced Research Projects Agency (DARPA) [129] and SpaceNet [131]. In (6b) the Foxhound vehicle and (6c) military vehicles show object detection and decomposition using neural networks [119,130].

2.4.1 Autoencoder Fundamentals

Autoencoder is commonly used for the superiority in dimensionality reduction, noise removal and anomaly detection, however; autoencoders meant to be designed for data generation. Normally, autoencoders trained in as unsupervised neural networks, nevertheless; they can be trained in supervised and semi-supervised ways also, but less common. Unsupervised, there are no correct labels to compare the results to. Supervised and Semi-supervised is the ability to use correct labels during the training of the autoencoder, but in this case, using the supervised or semi-supervised autoencoder cannot produce metrics such as accuracy and root-mean-square. Supervised and semi-supervised autoencoders has optimal use case when the autoencoder branched to hire other functionality such as classification or object detection [110, 111, 112].

Autoencoders are designed to reproduce their input at the output layer. The key difference between an autoencoder and a typical Multi-Layer Perception (MLP) network is that the number of input neurons is equal to the number of output neurons. Considering a simple autoencoder with one single hidden layer that represents the code as per fig 7, in order to produce the same output as the final layer, the internal hidden layers must learn what features are important. We can see the design of 5 dimensions reduced to 2 dimensions, then expanded back to the original 5 in the output.

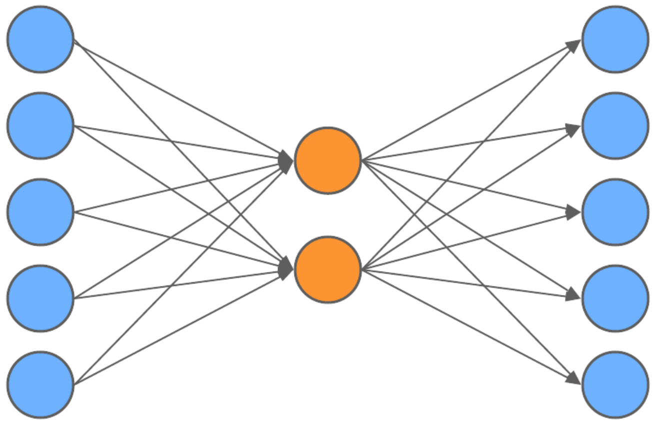


Figure 6. Shallow Autoencoder with single hidden layer [Source: Author]

Unlike feedforward network, autoencoders can be trained with the recirculation concept to reproduce its input at the output [104]. The hidden representation h in the middle attempts to obtain the significant input information, which is normally the unique input features. That, before they are decoded back to the output. This concept is extremely similar to the Principal Component Analysis (PCA) which tries to reduce the dimensionality into a few principal components.

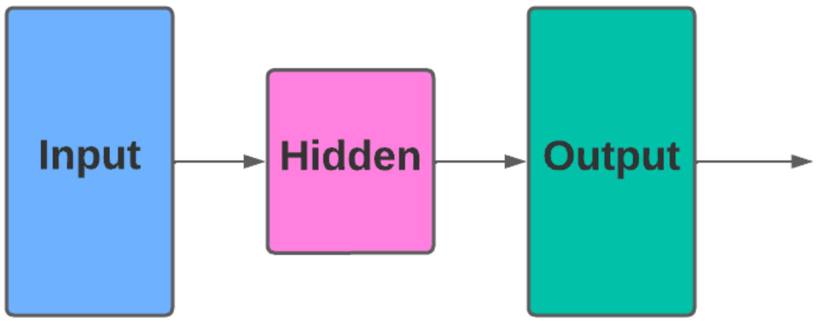


Figure 7. Feedforward network illustration [Source: Author]

The autoencoder network can be described in two parts, the encoder function $h = f(x)$ and the decoder that reconstructs the input features function $r = g(h)$. In case of the autoencoders successes to reproduce $g(f(x)) = x$ perfectly, then they are not significantly useful. Instead, autoencoders are implemented to be able to learn patterns not to only copy. Thus, they are controlled to allow them to copy only approximately, and to copy only input that resemble the data used during the training process. Further, the model is designed to prioritize features of as often learns useful properties of the input date.

Modern autoencoders have generalized the idea of an encoder and a decoder beyond deterministic functions to stochastic mappings, which are encoding distributions $p_{encoder}(h \mid x)$ and decoding distributions $p_{decoder}(x \mid h)$. Any latent variable model $p_{model}(h \mid x)$ defines a stochastic encoder

$$p_{encoder}(h \mid x) = p_{model}(h \mid x) \tag{2.1}$$

and stochastic decoder.

$$p_{decoder}(x \mid h) = p_{model}(x \mid h) \tag{2.2}$$

Generally, the encoder and the decoder distributions are not necessarily conditional compatible with a unique joint distribution $p_{model}(h \mid x)$, but asymptotically compatible [103].

2.4.2 Deep autoencoders

Considering the basic autoencoder structure, stacked autoencoders are commonly used when more hidden layers are needed to optimize and enhance the autoencoder calculations. Stacked autoencoders are also called deep autoencoders. Hidden layers are not only sub-selecting certain features but calculating combinations of the original features representing the original data in a reduced dimensional space.

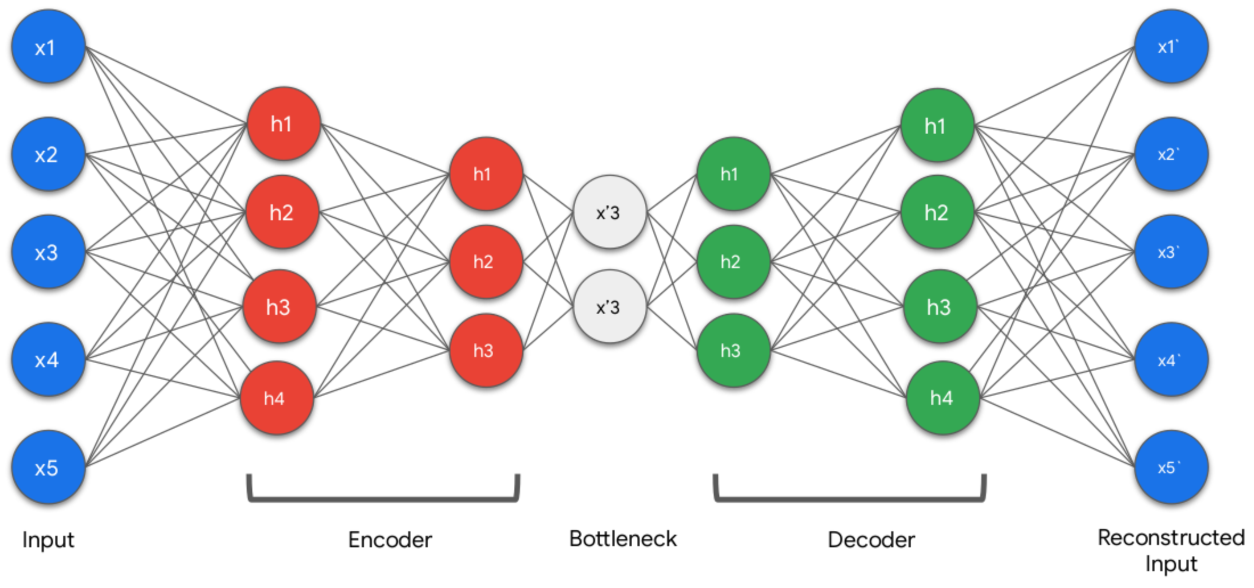


Figure 8. Stacked Autoencoder Illustration [Source DeepLearning AI]

The single hidden layer (Shallow) autoencoders and the stacked autoencoder have significant impact on the model losses and accuracy. The more hidden layers the autoencoder has, the better the output results. The following figures demonstrate the difference between shallow and stacked autoencoders. Modified National Institute of Standards and Technology database (MNIST) used in this demonstration [114]. MNIST consist of grayscale color band with images of 28 x 28 pixels. MNIST is commonly one of the best datasets to assess the performance of neural networks.

The MNIST handwritten digits dataset used to train and evaluate both shallow and stacked autoencoders. The results are elaborated where the autoencoder trained to encode the input images in lower resolution latent representation. The autoencoder learns to reduce the input vector to 4 x 8, then reconstruct the input back to its 28 x 28 original dimensions.

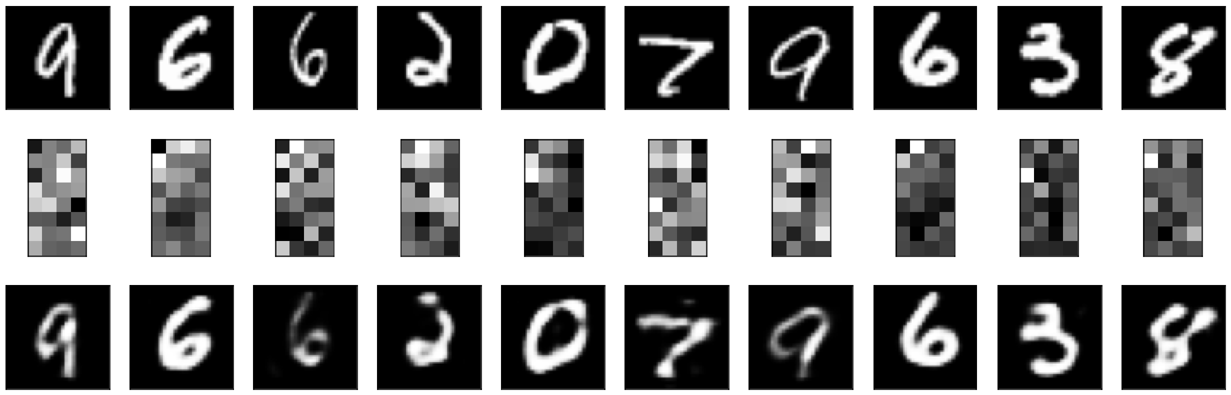


Figure 9. Shallow autoencoder output, from top to bottom, input, encoded and decoded

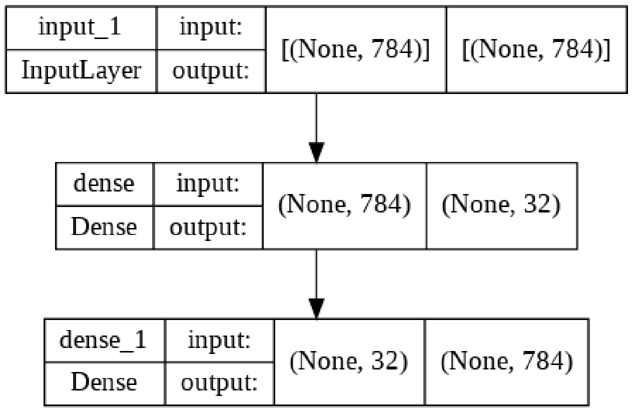


Figure 10. Autoencoder architecture

The shallow autoencoder consists of 1 hidden dense layer, In other hands, the deep autoencoder consist of 5 dense hidden layers. There is a noticeable improvement in the decoded output of the images, with lower losses from the training process, that as per fig 9 and fig 11. Furthermore, both autoencoder architectures are illustrated on the graph of fig 10 and fig 12. None in the input shape means that the autoencoder can be trained on unlimited number of images as input, in MNIST case 60000 images used for training and

10000 images used for validation. Also 784 is the resultant of 28 x 28 pixels after flattening the image to fit in the dense layer. Samples of MNIST are selected randomly as the dataset set to shuffle itself before each training process. Moreover, the code source developed to produce these figures is listed in appendix A.

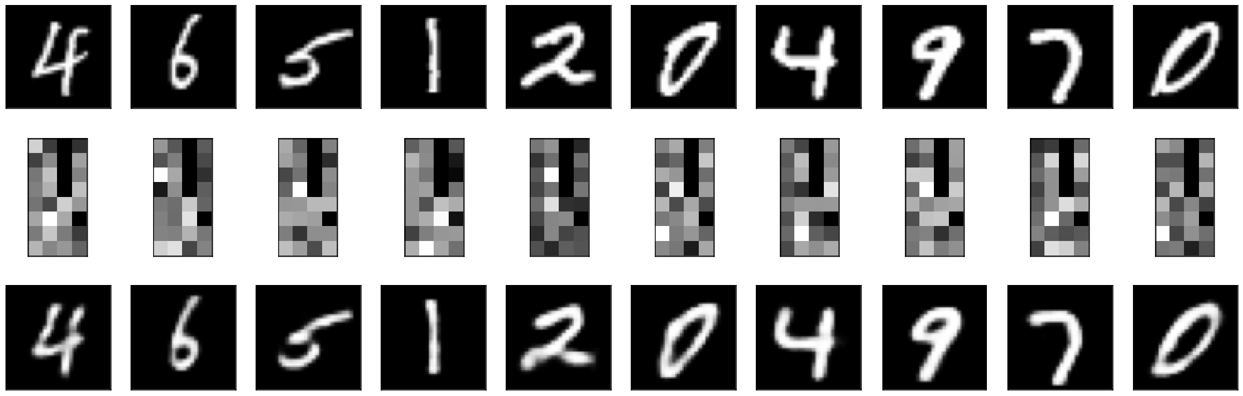


Figure 11. Stacked autoencoder output, from top to bottom, input, encoded and decoded

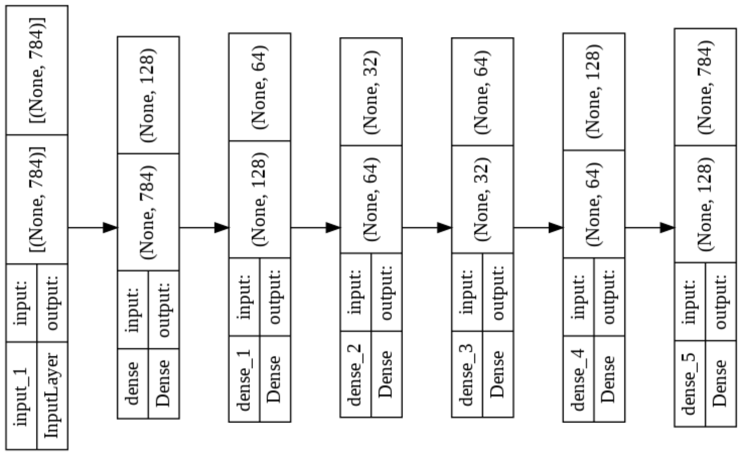


Figure 12. Autoencoder architecture

2.4.3 Autoencoder layers

Autoencoders can use various layer types when it comes to computer vision. Convolutional layers are superior in feature extraction compared to dense layers. Thus, creating the latent representation of an image reflect enhanced results during the decoding phase. When the autoencoder uses convolutional layers, it requires all image dimensions. Thus, a tensor of shape 28 x 28 x 1 is required for the MNIST images, which is the image width and height and the single color channel.

Convolutional kernels are the main advantage of a Convolutional Neural Network (CNN). They filter through the images to extract feature maps [115]. The spatial extents of the kernels are 3x3. The subsequent two figures show the difference between the Dense Neural Network (CNN) and CNN for building autoencoders.

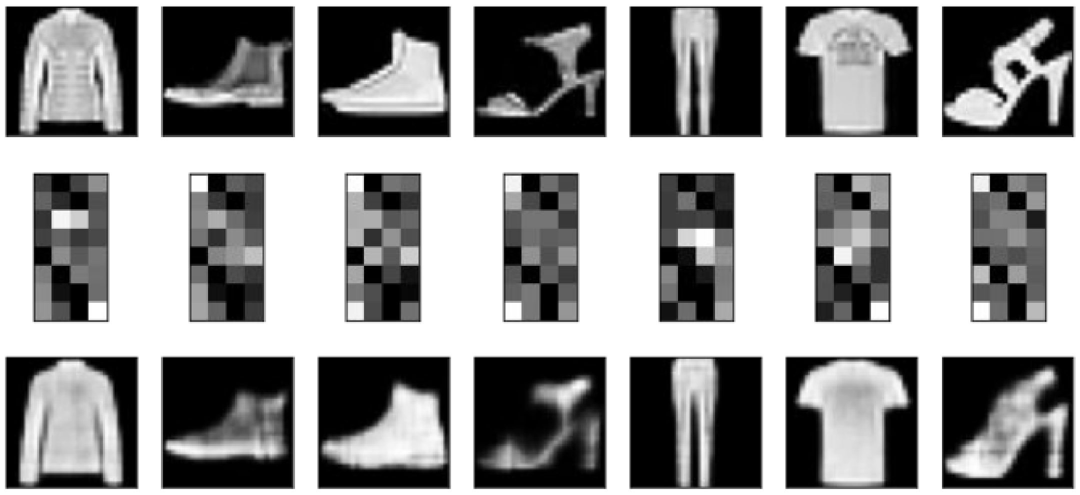


Figure 13. Autoencoder using DNN, from top to bottom, input, encoded and decoded

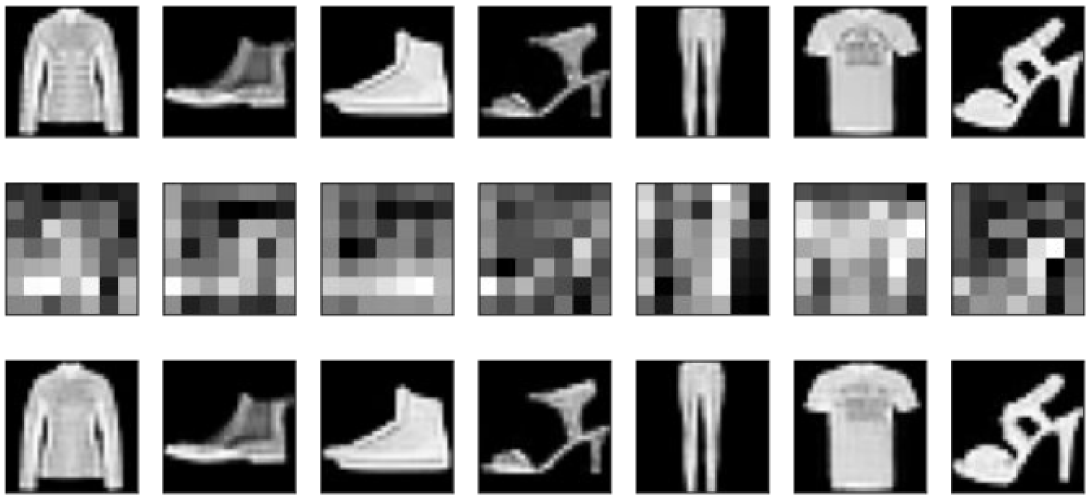


Figure 14. Autoencoder using CNN, from top to bottom, input, encoded and decoded

2.4.4 Denoising Autoencoder

The autoencoders, known also by its ability to reduce noise from its data, is tested. The fashion MNIST with same 28 x 28 grayscale images used, but with random noise applied to the training dataset. The denoising demonstration showed a superior ability of denoising the images as illustrated on the figure below. Further, the code developed to produce this demonstration is listed in appendix A. This denosing autoencoder is based on CNN not a DNN as the previous examples.

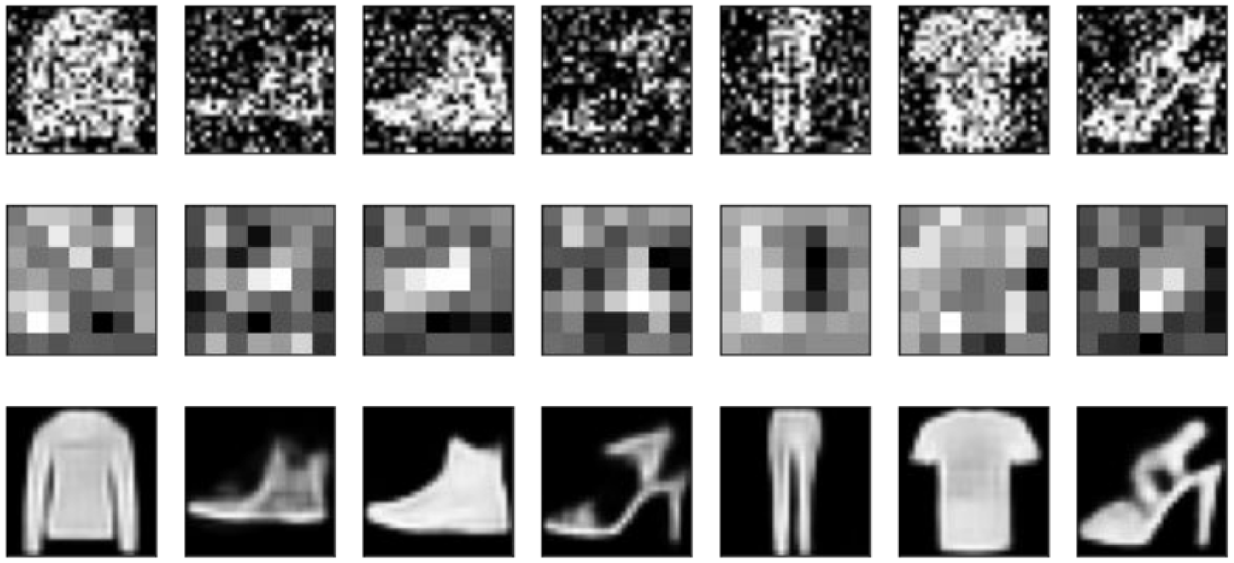


Figure 15. Denoting deep autoencoder, from top to bottom, input, encoded and decoded

2.4.5 Classical and Variational Encoders

Classical and variational autoencoders are the two prominent frameworks that are utilized for neural image compression. An instance of the former type is MLP and refers to the standard autoencoder architecture depicted in Fig. 6 [27]. Variational autoencoders utilize an additional layer of encoding, which is specifically prepared, depending on the objectives [37]. As a result, variational autoencoders demonstrate positive results on both image compression and change detection [47,53,57]. For instance, the improved variational autoencoder shows better accuracy than most traditional algorithms and standard convolutional autoencoders for desertification detection based on satellite imagery [65,67]. Therefore, the implementation of variational autoencoders might be highly beneficial for various uses in the industry. The encoded latent variable z for 1-D case can be presented using as probability equals to the normal distribution of the input mean and standard deviation, such that $z \sim q_{\mu,\sigma}(z) = \mathcal{N}(\mu, \sigma^2)$, first a sampling of $\epsilon \sim \mathcal{N}(0,1)$ done, than z is produced by $\epsilon \sim \mathcal{N}(0,1)$. For a multidimensional vector, the calculation follows $\epsilon \sim \mathcal{N}(0, I)$, The decoding process can be described as sampling process from the latent space. Further illustration about latent calculation represented by the pseudo matrix operation as shown on equation (2.3), where Z has fewer dimensions than the right-hand side. For example, with weights matrix W in shape of (2, 5) multiplied with flattened input vector X with shape (5, 1) then subtract the bias vector which in shape of (2, 1), the resultant Z come in shape of (2, 1). That explain the dimensionality reduction mathematically.

$$[W]@[X] + [Bias] = [Z] \tag{2.3}$$

2.4.6 Activation functions

Activation functions are supporting in increasing the neural network learnability and ability to solve nonlinear problems. One of the main advantages of the Rectified Linear Unit (ReLU) function is increasing the sparsity of the model, that by increasing the number of zeros during the multiplications. Thus, ReLU is used commonly as an activation function in deep learning. Further, the Sigmoid mathematical function is also used to add non-linearity to the model values. The ReLU and Sigmoid mathematical representation are shown in equation (2.4) and (2.5) respectively. The ReLU function mainly map the negative input values to 0. Furthermore, the Sigmoid can be used to map the input values be between 0 and 1, or to -1 and 1. One of the most used Sigmoid functions in deep learning is the logistic function, which maps any real input value to the range between 0 and 1. The characteristics of Sigmoid graph looks as a S-shape, which reflect the name from the Greek letter sigma. Both ReLU and Sigmoid used to map real input values to one that can be interpreted as a probability [144].

$$ReLU(x) \; = \; max(0, x) \tag{2.4}$$

$$h_\theta(x) = \frac{1}{1 + e^{-}\theta^T x} \tag{2.5}$$

2.4.7 Multi-Layer Perceptrons

Multi-layer perceptrons have two main layers: the input and output neurons. There is, however, a hidden layer between the two (input and output). Theoretically, the more hidden layers, the more effective the algorithm in dimension reduction and data compression [15,18]. MLP modifies the spatial data unitarily. In 1988, the first MLP algorithm for image compression was released. The novel algorithm comprised the conventional techniques including binary coding, spatial domain transformation, and quantization, all operating as one single file compression tool [28,35]. The approach employed neural networks to establish the most suitable combination of binary codes, but could not adjust the parameters to adjustable compression ratio [61,75]. The program has been advanced by introducing predictive algorithms to predict all pixel values with respect to their neighbors. The program also includes backpropagation to minimize the mean square error between original and projected pixel values [18, 68].

2.4.8 Specific Models

Most contemporary neural networks are based on the classical and variational models of autoencoders. Consequently, a derivative from these types - a recurrent neural network (RNN) - is the basis for numerous frameworks [87]. The experts primarily adjust the quantization and entropy coding of neural image compression to achieve the best possible accuracy of reconstruction and storage efficiency [97]. Furthermore, while such autoencoders generally perform adequately based on specialized hardware, some models demonstrate positive results even on resource-constrained edge systems [17]. As a result, regardless of the conditions, it is possible to adjust the frameworks according to the objectives. The technological advancement in this area would allow closing the research gaps in the satellite industry. For instance, comprehensive image compression methods might be used to improve the global agricultural monitoring systems. From these considerations, the industry might significantly benefit from the innovative framework of satellite imagery compression.

3 Literature Review

3.1 Introduction

The proliferate growth in the number and size of satellite images captured in recent years has spurred the need to develop new, better, and more efficient compression techniques. The increase in size is a result of the improvement in the satellite sensor capabilities, which capture electromagnetic radiation reflected by the surface of the earth with higher precision than older generations. This research explores different image compression and decompression techniques employed in the storage and processing of satellite images. As of date, satellite images are by far the most effective way of representing landscape. The information captured by satellites is essentially used by urban planners, weather and climate change analysis, soil science, Internet of Things (IoT), military, and intelligence agencies to track different progressive activities on the surface of the Earth [10]. This chapter presents a review of existing literature from articles published in the last five years (since 2018) with respect to the capture, storage, compression and decompression, and processing of satellite images. The chapter begins by exploring the need, history, advancement in quality and storage, as well as the different techniques used to compress images. The chapter also explores the advantages and disadvantages of different image compression techniques when applied to satellite images.

3.2 History of Satellite Images

The history of satellite imagery dates back to the advent and proliferation of the space race between the Soviet Union and the United States of America [20]. Back then, all images were recorded in black and white, and the quality was extremely low. Since detailed imagery was not a major concern, the space rivalry was more concerned about conquering space before one another. As a result, the nations spent huge amounts of resources to develop spacecraft which could reach further than their rivals. However, the success of the missions would not be recognized until there was proof of success: that is, images captured from or about spatial objects [22]. Back in the day, satellite images devices were solely owned by government organizations such as the National Aeronautics and Space Administration (NASA) of the US and the Russian Space Agency, Roscosmos.

As the space race heightened, NASA and Roscosmos developed better, more effective, and more sophisticated imaging tools which were used to image the earth, moon, and other terrestrial bodies [24]. The US was the first to records satellite images of the earth from space in 1959 while the Soviet Union captures a satellite image of the moon later that year [19,26]. The most popular satellite image of the earth was captured in 1972 by the Voyagers and remains the most outspoken image of the earth to date [28]. The space race ended when NASA landed the first man on the moon in 1969 onboard the Apollo 17 [30]. The groundbreaking achievement was followed by the development of earth imaging programs that used satellites to capture and record the earth's surface. The Landsat program, launched by the US in 1972 was meant to captured satellite imagery of the earth [32,33,34]. The information is stored in the NASA Earth Observatory database which is accessible to the public for free [38]. To date, Landsat remains the largest earth imagery acquisition program ever launched by a government or private entity

3.3 Advancement in Satellite Imagery

The application of satellite imagery technologies gained popularity among and are now used in agriculture, meteorology, agriculture, oceanography, cartography, regional planning, education, intelligence, and warfare. The applications also extend to biodiversity conservation, geology, landscape, and agriculture. Although some satellite imagery enthusiasts use the images for minor applications such as hunting or other unexplained applications, their essence remains unparalleled as time goes by. The application of satellite images depends on the clarity, quality, and geographical data presented on the images such as forestry, landscape, soil types, infrastructure, and natural features.

As governments and private entities launched earth imagery programs, the amount of earth information contained in satellite images grew. The quality and number of sensors installed on earth-orbiting satellites also improved, resulting in clearer and more accurate images. The data transmission rates of satellite images also increased, allowing systems to capture and transmit more images per unit time. Satellite signals travel at the speed of light (300,000 kilometers per second) [40]. The current data storage media, mainly magnetic disks and solid-state disks dwarf the floppy disks used to store satellite images in the early stages of the earth imagery programs [42,43]. Computer processing speed has literally doubled every two years as predicted by Moore [41,44]. In other words, the development of computer microprocessors has always obeyed Moore's law, and modern computing power has grown billions of times. During the launch of the Landsat, the overall capturing, transmission, storage, and processing of satellite images were low, a factor that has greatly changed over the years. The low and high earth orbit is comprised of satellite systems launched by governments and the private sector as explained below.

3.4 Public satellite imagery system

The earth's orbit is open to public and private investors to launch satellites for whatever reason. However, launching satellites in space should be done in a manner that one's satellites do not endanger other satellites in space. The earth imaging satellites have captured a lot of data that has been freely available to the public for scientific use. Below is a detailed list of earth imaging programs owned and managed by different governments or unions around the globe.

3.4.1 CORONA.

The CORONA program was launched by the US central intelligence Agency with the help of the US Air force. The Directorate of Science and Technology within the CIA spearheaded the program and used the wet film panoramic technology [48]. The satellites used two cameras for capturing earth images.

3.4.2 Landsat

Landsat stands as the oldest earth imagery program ever launched by humanity. The satellites recorded images with 30 meters accuracy and resolution from the early 1980s [50]. The program has been updated over the years and renamed according to the generation of satellites launched in orbit. As from Landsat 5, the system began capturing earth images using thermal infrared instead optical sensors to capture data. The current generation of satellites in orbit for this program are Landstats 7 to 9.

3.4.3 MODIS

The program was launched in 2000 and used 36 spectral bands to capture earth imagery on an almost daily basis. The sensors are installed on Aqua and NASA Terra satellites of the US.

3.4.4 Sentinel

The Sentinel is a constellation of satellites planned by the European Space Agency which will be launched in seven missions. Each mission is designed to perform a specific function ranging from land surface imaging using decametre optical imaging, to land and water imaging using thermal and hectometer optical imaging sensors. Currently, three missions, Sentinel 1 to 3 have been launched and are already in use.

3.4.5 ASTER

The program was launched in 1999 by NASA as an Earth-observing System. The Japan Space Systems, as well as the Ministry of Economy, Trade and Industry, were also involved. The system is designed to capture detailed images and maps on eland surface elevation, reflectance, and temperature [52]. The program contributes immensely to NASA's division of Science Missions and Earth Science. It has contributed immensely to understanding and forecasting volcanoes, surface climatology, hazard motioning, hydrology, land cover, and change as well as surface and ecosystem change [45, 53, 55]

3.4.6 Meteosat

It is a weather monitoring Earth-imaging system in operation since 1981. The sensors are designed to detect weather factors such as water vapor, water bodies, clouds, and other weather-related elements. Since 1987, the Meteosat is operated by Eumetstat. Different generations of Meterostat are in operation including the Metesostat Visible and Infrared Imager which is a three-channel system using the first generation Meteosat [56]. The Spinning Enhances Visible and Infrared Imager has provided continuous data on climate change for the past decades [54, 57,58]. The next generation Meteosat, the Flexible Combined Imager will encompass the technologies used in the first and second generations.

3.5 Private satellite imagery systems

They are owned and operated by private corporations. Although most of the systems have not been launched, they are in development phases and will capture and provide essential data for the scientific and industrial communities. The systems are listed and described below:

3.5.1 GeoEye

The satellite has been in operation since 2008 and captures earth images in high resolutions. Black and white images have a resolution of 16 inches while colored images have a resolution of 64 inches [36, 60]. It is owned and operated by the GeoEye Company.

3.5.2 Maxar

Maxar owns the WorldView-2 commercial satellite which captures high-resolution images with 0.46 meters. The satellite-only uses panchromatic mode only and can distinguish objects that are at least 46 centimeters apart. The company also owns the QuickBird satellite with a spatial resolution of 60 centimeters. The WorldView-3 satellite has the highest spatial resolutions at 31 centimeters. The satellite includes both atmospheric and infrared sensors on board.

3.5.3 Airbus Intelligence

The Airbus Intelligence owns the Pleiades constellation that comprises two sets of high-resolution earth imaging satellites (105 and 201 meters). The two satellites are Pleiades-HR 1A and 1B. The satellites image the surface of the earth and operate as civil and military satellites. They are designed with the European defense standards in mind [62]. The Pleiades Neo constellation comprises four satellites with 0.3m spatial resolution.

3.5.4 Spot Image

SPOT Image has three high-resolution satellites orbiting the earth. The satellites provide a 1.5 panchromatic channel and 6-meter multi-spectral resolutions. The satellites were launched in 2011 and 2012 and are also used by the Taiwanese Formostrat-2 and South Korean Kompstat-2

3.5.5 Planet’s RapidEye

The RapidEye Constellation was launched in 2008 by BlackBridge, which was later acquired by Planet in 2015. The constellation contains identical, calibrated sensors, ensuring all images collected by all satellites are equal in size. The feature enables the constellation to capture up to four million square kilometers in a day. The imagery captured by the constellation is applied in agriculture, disaster management, cartography, and environmental management [63,67]. The constellation was, however, retired in 2020.

3.5.6 ImageSat International

The network is comprised of the smallest high-resolution earth mapping satellites orbiting at low altitudes. The satellites are designed to move fast between the target objects. The network is also called Earth resource Observation Satellites, or at times, EROS. The satellites operate near the poles in a circular sun-synchronous manner, orbiting at 210 meters. Although the satellites are mainly used for military and intelligence purposes [64], they are also used in infrastructure planning, border control, land mapping, and disaster response.

3.5.7 China Siwei

China Survey and Mapping technology company owns and operates four satellites called the Superview. The satellites orbit the earth at 530km and operate in the same orbit [66,69]. The satellites are of high resolution, up to 2m and 0.5 m multispectral and panchromatic resolutions respectively [16,71,73]. The satellites are also called Gaojing-1 with the labels 1,2,3 and 4.

3.6 Pros And Cons of Satellite Image Compression

For a variety of reasons, many of the images available on the Internet today have been compressed. Users can benefit from image compression since images load quicker and take up less space on a storage device. Image compression does not lower the actual size of an image; rather, it reduces the actual data components of the image. One of the most significant advantages of satellite imagery is the capacity to examine huge areas of the Earth fast. At the same time, the present satellite data's coverage restrictions are visible [98]. Many natural phenomena with higher spatial and temporal heterogeneity are not adequately caught by polar-orbiting imagers in Low Earth Orbit (LEO), which attain global coverage in a minimum of one day (but usually two or more days). This constraint is addressed by high-orbit geostationary observations (GEO), which provide many daily views of the same target. There is, however, a trade-off between satellite picture resolution and spatial coverage (typically, larger coverage results in lower resolution). For many applications, obtaining observations with both large geographic-temporal coverage and high spatial resolution is required, but it is also quite difficult. As a result, new inventions, supplemental data, and synergies of complementary observations may be called for in the design of satellite sensors to tackle specific objects or issues. In the following part, we'll go over this in more detail.

Although satellite observations have demonstrated their excellent capabilities, the data currently delivered by our satellite equipment has low information richness for many applications. As a result, it is time to deploy new sensors with expanded capabilities. For example, it is well known that Multi-Angular Polarimeters (MAPs) provide the best data for characterizing detailed columnar properties of atmospheric aerosol and cloud [97].

Several advanced parametric missions, including Hyper-Angular Rainbow Polarimeter (HARP), Multi-View Multi-Channel Multi-Polarization Imaging mission (3MI) on MetOp-SG satellite [95], Spectropolarimeter for Planetary Exploration (Spex), and Multi-Angle Imager for Aerosols (MAIA) instrument [96] as part of NASA PACE mission. More so, the China National Space Administration (CNSA) has invested heavily in polarimetric sensors [94]. The MAI/TG-2, CAPI/TanSat, DPC/GF-5, and SMAC/GFDM are among the polarimetric remote sensing instruments that CNSA has recently released, with the POSP, PCF, and DPC-Lidar to follow in the future years [89]. The principles of these sensors, their technological designs, and algorithm development have all been extensively explored and tested using aerial prototypes [91]. Below are some major impacts of image compression and how they relate to satellite imagery.

3.7 Reduction in Size

The most important benefit of picture compression is the reduction in file size. You can keep compressing the image until it is the size you want it to be, depending on the file type you're dealing with. Unless you adjust the image's physical size with an image editor, this implies the image occupies less space on the storage media while maintaining the same physical size. This file size decrease is ideal for the Internet, since it allows webmasters to produce image-rich sites without consuming a lot of storage space or bandwidth.

3.8 Slow Devices

Large, uncompressed images may take a long time to load on most electronic devices, for example, digital cameras and computers. Compact disks drives, for instance, only read data at a certain rate and cannot show huge graphics in real-time. Compressed pictures are also required for a fully working website on some web hosts that send data slowly. Uncompressed data will also take a long time to load on other storage devices, such as hard disks. Image compression helps data to load more quickly on slower devices.

3.9 Degradation

When compressing images, one may see image deterioration, which means the image's quality has deteriorated. In most image formats such as GIF or PNG, the image data is preserved even though the image quality has deteriorated. A slight degradation in satellite images could have devastating consequences in analyzing the images [87]. If you need to show someone a high-resolution image, whether large or little, image compression will be a disadvantage.

3.10 Data loss

Compressing some image formats reduces the file size which means a part of the file is permanently deleted. As a result, it is important to keep backup copies of the uncompressed images which nullifies the importance of compressing the images in the first place [85, 93]. Instead of saving storage space, keeping a backup copy occupies more storage space.

3.11 Problems with satellite images

Satellite databases are big and image processing (generating meaningful pictures from raw data) is time-consuming since the total amount of land on Earth is so large and resolution is so high [82.87] Image de-striping, for example, is frequently necessary as part of the preprocessing process. Weather conditions might impact image quality depending on the sensor used: for instance, it is difficult to get photographs for regions with regular cloud cover, such as mountaintops. Third companies usually process satellite image datasets that are publicly available for visual or scientific commercial usage for these reasons [96]. Commercial satellite businesses do not release their images to the public or sell it; instead, a license is required to utilize such imagery. As a result, the ability to use commercial satellite images to create derivative works is limited. Some people have expressed worries about their privacy since they do not want their property to be visible from above. In its Frequently Asked Questions (FAQ) section, Google Maps answers to similar issues with the disclaimer: "We recognize your privacy concerns... The visuals displayed by Google Maps are identical to those viewed by anyone flying over or driving by a given geographic place.

3.13 Development of the State-of-Art Data Processing Approaches of the Next Generation Satellite Images

An important factor that influences the final product's quality is the quality of a remote-sensing capturing method used. In reality, the quality of the generated remote data cannot be significantly enhanced once the equipment has been installed, although retrieval techniques are constantly improved. The eventual remote sensing output may change significantly, not just as a result of ingesting data from many types of equipment, but also as a result of improved retrieval ideas. In this context, in the last decade, a new generation of satellite image processing algorithms has made substantial progress. New techniques, for example, are capable of extracting a huge number of parameters and rely on rapid and precise atmospheric modeling (rather than precomputed Look-Up-Tables, or LUT). Furthermore, simultaneous retrieval of aerosol characteristics is possible. In addition, retrieval of aerosol characteristics in conjunction with land surface and/or cloud properties has been introduced [79, 86]. Finally, the combined retrieval of CO2 and aerosol characteristics, as stated above in the context of the CO2M EU/Copernicus project, is a potential technique for lowering the influence of aerosol pollution on the resultant CO2 product.

For reliable cloud remote sensing from space, certain computational hurdles remain. It is necessary to have an efficient and accurate radiative transfer model. While independent column approximation is commonly used to retrieve optical depth and cloud droplet size, cloud top roughness causes 3-D radiative transfer (RT) effects which can cause retrieval biases [79, 81]. Starting with linking their retrieval into a joint framework, the 3-D character of clouds becomes more of a concern for exploring the interlinkage between aerosols and clouds, for example, near cloud boundaries. In this context, a pressing need exists for the development of an inversion-targeted quick but accurate 3D RT model for optically and geometrically multiplex media, with the inclusion of the spectral signature of gas absorption and correct adoption of the cloud particle scattering model. For proper interpretation of all satellite pictures, the development of credible 3D radiative models is also required to account for horizontal variability of the land surface [77]. Another important unsolved problem is generating 3D cloud fields to represent 3D radiation fields, which might be solved by combining active and passive sensors [23].

Cirrus clouds have an important role in weather and climate processes, according to several observational and modeling studies [73,75]. Cirrus clouds, despite their visual thinness [51], have a worldwide presence, control Earth's radiation, and play a vital role in the study of climatic systems. Cirrus particles have very irregular forms, and their single-scattering characteristics, such as single scattering albedo and scattering phase functions, differ dramatically from spherical particles [59.73]. These irregular forms can produce significant biases in the cloud and aerosol retrievals if an algorithm does not recognize them [55,59]. As a result, identifying a realistic cirrus particle model and incorporating it into aerosol retrievals is a viable path to pursue. Furthermore, advancement in the global chemical and climate transport models is highly linked to the utilization of satellite data (CTM). When observations are unavailable, for example, trustworthy aerosol retrievals can be included in Chemical Transport Models (CTMs) to give precise aerosol loadings [50]. On the same hand, spectral and polarimetric data have a lot of sensitivity when it comes to constraining aerosol type [43,47,49,51], and satellite data can help improve the study of transport models emitting atmospheric components [33]. As a result, combining the processing of satellite data with existing modeled data is another interesting study area for satellite remote sensing advancement.

Finally, machine learning techniques are now being employed more often to identify patterns and insights from geospatial and remote sensing data [27, 39]. Because it presents techniques that can "learn" from data, find patterns, and make judgments with minimum human interaction, this area of artificial intelligence is particularly suited and appealing for the study and exposition of Earth observation data. Deep neural networks, in particular, have lately been employed in remote sensing investigations, particularly for the processing and interpretation of large volumes of data. Such methods demonstrate the possibility for automatically extracting Spatio-temporal linkages and gaining additional knowledge useful for enhancing predictions and modeling of observable physical processes over various timeframes. These approaches are particularly promising for satellite data interpretation, especially when data-driven machine learning is combined with physical process models [83].

3.14 Deep Learning Models for Image Compression

AI-Ready microchips have evolved during the last decade, creating multiple divisions of Canonical architecture based on various use cases. Canonical Artificial Intelligence (AI) architecture consists of sensors, data conditioning, algorithms, modern computing, robust AI, human-machine teaming, and users (missions) [101]. Each technology evolution step is critical in developing end-to-end AI applications and systems, as illustrated below. Nowadays, Modern AI-Ready chips such as Xilinx Zynq Ultrascale, Phytech, and Google Coral Edge TPU (Tensor Processing Unit) employ the use of the modern AI-optimised development frameworks. Frameworks such as Vitis AI, HLS4ML on the Python productivity for boards (PYNQ), the Open Visual Inference and Neural Network Optimization (OpenVINO), and the Open Neural Network Exchange (ONNX). These frameworks are used to accelerate the AI application life-cycle from development to prototyping and deployment to production, cross-platform deployment in particular use cases. Although, some of these frameworks works optimally in association with their commercial boards only, they have common application life-cycle. Furthermore, they are utilized to develop deep neural networks and machine learning models, which applied in image compression [77]. Canonical AI architecture illustrated on figure 17 from the Lincoln Laboratory Supercomputing Center of Massachusetts Institute of Technology (MIT).

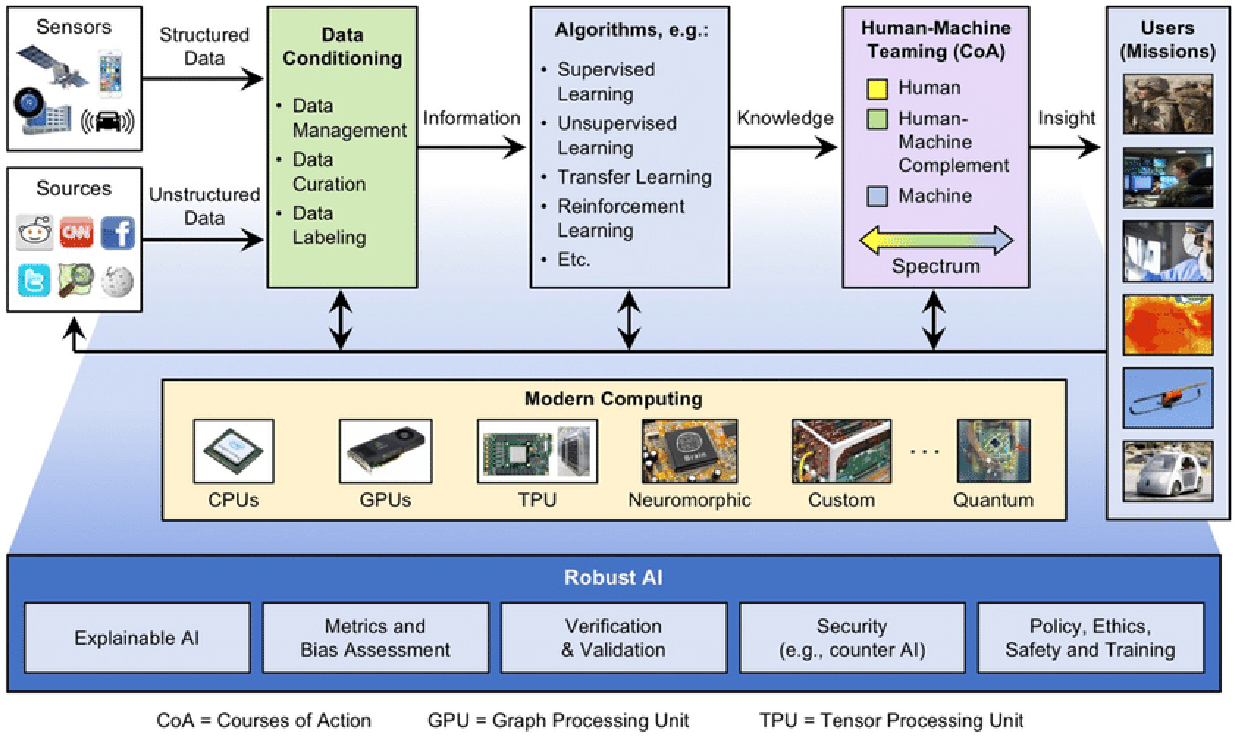


Figure 17. Canonical AI architecture [Source: MIT, LLSC [101]]

4 Methodology

The methodology plays an important role in exhibiting the validity of the research. Usually, research methodology defines data collection and processing tools, data sources and sample sizes, as well as research requirements [90]. The methodology can be compared to a formula in mathematical expressions. In this case, it defines how the research will be carried out, the variables needed and how the projected outcome will be presented. Since the research relies on secondary data, it is more qualitative than quantitative one. As a result, the research was approached as a qualitative study seeking to evaluate the best autoencoder architecture for satellite image compression. That with considering new space trends such CubeSats and constrained resources.

In this research, autoencoders selected as the main compression method discussed. Further, the implementation and evaluation of different autoencoder neural networks are discussed in the analysis and results chapter.

4.1 Technical objectives

This thesis is an execution of one objective of the ScOSA project, which is to evaluate, if neural networks can be an alternative to classical compression algorithm. Thus, the main objective of this research is to study the adaption of neural image compression technologies in satellite imagery and the use cases of deep autoencoders. The following table list the main technical objectives of the thesis.

Table 1. Technical objectives

OBJ-ID	Technical Objective
OBJ-01	Evaluate, if neural networks can be an alternative to classical compression algorithm.
OBJ-02	Identify different neural image compression techniques.
OBJ-03	Evaluate neural image compression state-of-the-art on satellite images.
OBJ-04	Implement an auto-encoder for satellite image compression.
OBJ-05	Validate the produced neural network model on random satellite images

4.2 Requirements

Current ScOSA advancements requires 20W for 4 nodes, the neural image compression required to run on 2 nodes maximum. The evaluation and implementation processes shall comply with the following system requirements.

Table 2. Requirements

REQ-ID	Requirements
REQ-01	The autoencoder architecture shall fit on a space-qualified FPGA.
REQ-02	The autoencoder architecture should fit on two Zynq-7020 maximum.
REQ-03	The autoencoder implementation should fulfill a real-time condition of 1 Hz frame rate.
REQ-04	The autoencoder architecture should be distributed on two heterogeneous nodes.
REQ-05	The overall power budget for image compression shall not exceed 10W.
REQ-06	The autoencoder hyperparameters shall be dynamically reconfigurable.
REQ-07	The autoencoder trained model shall be split in encoder and decoder.

4.3 Mission scenarios

This project aims to conduct a higher Technical Readiness Level (TRL) by preparing the in-orbit demonstration of the developed ScOSA on-board computer. The implemented autoencoder performance shall be demonstrated on future DLR-CubeSat mission.

4.4 Concept trade-offs

In addition to the trade-off between reliability and performance, which is the DLR mainly investigating in the ScOSA project, there is also a trade-off between image quality and compression factor. Image quality, measured based on image similarity structure indices and the mean square error between raw and reconstructed image.

4.5 Development tools

A few development frameworks required to implement and evaluate the autoencoders, as well as to analyze the results. The deep learning framework for this research is TensorFlow. The fast machine learning HLS4ML based on Python PYNQ overlays and Vitis Ai development platforms selected for implementation and analysis on Xilinx FPGAs. The following figure shows the abstraction of the PYNQ framework and the workflow.

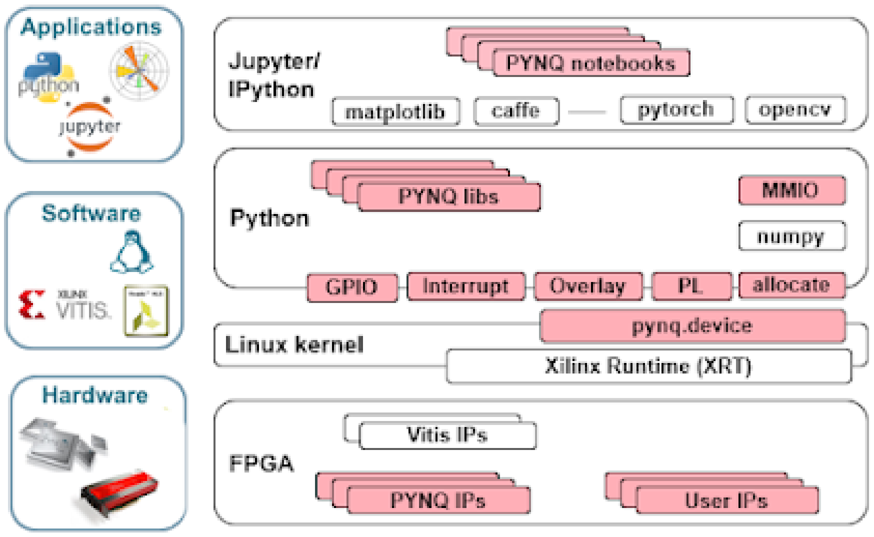


Figure 18. PYNQ Python based FPGA development framework diagram from Xilinx [102]

The TensorFlow Keras Functional API shall be used instead of the classical sequential method for building neural networks [162]. The Functional API will give the architecture implementation process more high resilience to adapt to the design needs, such as model breakdown. For example, with functional API the autoencoder can be split in 3 parts, that will open doors to deploy the compressor (encoder) only to production platform like satellites with constrained computing resources as CubeSats. Further, keep the decompressor (decoder) part of the autoencoder on the ground segment to be used for image reconstruction.

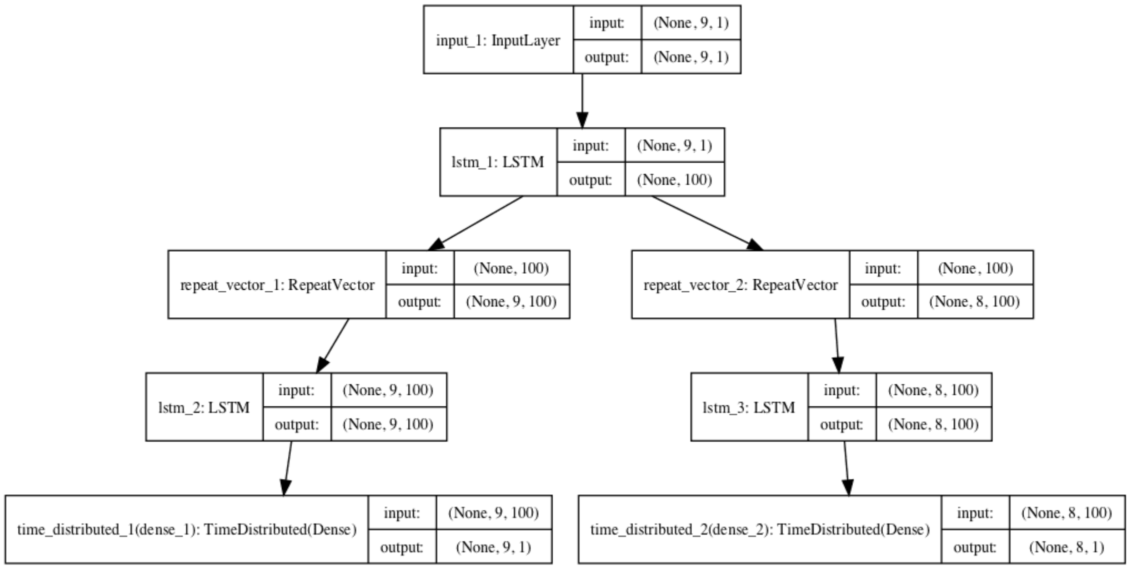


Figure 19. Illustration of branched model using Functional API [Source: MachineLearningMastery [143]]

4.6 Data collections

Once the aircraft lands, the data collected during airborne remote sensing and missions may be downloaded. After then, it may be processed and transmitted to the intended recipient. However, because the satellite remains in orbit throughout its lifetime, data collected from satellite platforms must be electronically sent to Earth. If the data is immediately needed on the ground, the technology developed to do this can also be utilized by an airborne platform.

For transferring data from satellites to ground stations, there are three basic ways. The choice of transmission depends on the position of the satellite and the ground station. First, if the ground station is visible to the satellite, the data is sent directly. In the second option, the satellite can store the data and wait until the ground station appears in its line of sight. The third option is applicable for urgent purposes. If the satellite capturing data is not in the line of sight with the ground station, the data can be shared with another satellite that forwards the signal to the ground station [31].

Several earth imaging programs, mostly owned by governments, are available to the public for scientific, academic, and exploratory use. The datasets exist in large sizes and are best accessed through cloud computing services such as Amazon, Azure, Google Colab, and

Kaggle. The services provide high computing and storage services that cannot be accessed on personal computers. There are several ground pickup stations around the world to facilitate the collection and processing of satellite signals. Usually, satellites send raw digital data which must be processed to the appropriate geometric, atmospheric, and systematic distortions. The data is also converted into the right format for storage in the various satellite imagery datasets. The transformed data is stored in conventional storage devices such as magnetic hard disks, compact Disks, or solid-state media such as thumb drives [21]. Visible Satellite Images (VIS) are captured by reflecting the sun's light rays. The images represent what is visible to the human eye in all colors as they appear. However, the images are presented in 2D and are hence static. Most water bodies appear in blue while forest covers appear in green. VIS images are easier to analyze from a human point of view [4]. The images are also easier to compress using both lossy and lossless techniques.

4.6.1 Satellite and instrument datasets (IR and VIS images)

Infrared satellite imagery acts as a temperature map in that weather satellites detect heat energy in the infrared spectrum [62]. Object visible to the human eye such as water, land surfaces, and clouds are displayed on the satellite image depending on their temperature. Dark colors represent warm temperatures while light colors represent low temperatures [52, 94]. A temperature scale usually accompanies the satellite image for clarification and interpretation purposes. The public IR satellite image dataset can be found on the Sentinel Open Access Hub. In addition to the public data, data collected from DLR such as the Bispectral InfraRed Optical System (BIROS) satellite, which is the second satellite from the FireBird mission [113]. Further data collected from the DLR Earth Sensing Imaging Spectrometer (DESIS) which is a hyperspectral Earth observation instrument on the Multiple User System for Earth Sensing (MUSES) platform on the International Space Station (ISS). Further, data collected from the DLR Modular Aerial Camera System (MACS).

4.7 Data Preprocessing

As a typical deep learning process, datasets require preprocessing before using them in the training or inference cycles. Typical preprocessing for training is splitting the dataset into training and test datasets. Training and test may follow the ratio of 80 / 20, respectively. The test dataset must not be used during the training; otherwise, the training validation fails logically. Further preprocessing is reshaping the data input to a single dimension vector, or in other words, called data flattening. Furthermore, data shall be mapped to values between 0 and 1 to validate with the matrix multiplication and neural network calculations. In this research case, using an autoencoder, most of the preprocessing shall be reversed after the decoding phase to reach the optimal reconstructed data.

The data processing shall also include a patching method for high-resolution images. The data patching method is slicing the data into small patches to fit and accelerate the training and inference processes, then rearranging the sliced patches during the reconstruction phase. This research will use the data patching method to fit BIROS images to constrained dimension autoencoder and infer on an embedded system like the Zynq-7000 FPGA. Further, data patching also will be applied to DESIS data during the hyperspectral compression experiment. The following figure shows an example of patched images from, the GeoEye satellite.

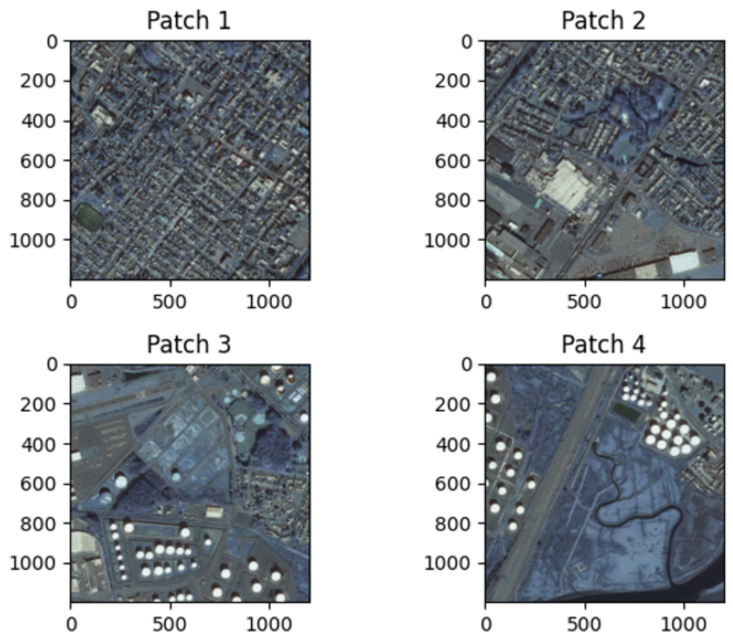


Figure 20. Example of the patch preprocessing [Source: GeoEye, IKONOS]

4.8 Analysis Technique(s)

Based on the collected data, autoencoder implementation trained and evaluated on VIS, IR, and Hyperspectral Images (HSI). Analysis shall consider the different compression ratios, resolution of reconstructed images, Mean Square Error (MSE) of the images, the Structural Similarity Index (SSIM) and the Multiscale Structural Similarity Index (MS-SSIM) between original and reconstructed images. Further, analysis shall discuss the procedures to deploy autoencoders on AI-Ready boards. Finally, the analysis should conclude with a comparison between the different autoencoder types used for data and image compression. The MSE described in equation (4.1), where N is the width and M is the height of the compared images. High MSE value refer to high losses between the original g and compared ĝ image.

$$\epsilon_{MSE} = \frac{1}{MN}\sum_{n=1}^{M}\sum_{m=1}^{N}[\hat{g}(n,m) - g(n,m)]^2 \tag{4.1}$$

The Structural Similarity Index described in equation (4.2), where x and y are the two in;put images for the comparison. The SSIM is based on three comparison measurements, luminance (l), contrast (c) and structure (s). $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are the comparison weights. The SSIM measure the symmetry properties, which make it more accurate than the MSE.

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha} \cdot [c(\mathbf{x},\mathbf{y})]^{\beta} \cdot [s(\mathbf{x},\mathbf{y})]^{\gamma} \tag{4.2}$$

Furthermore, the Multi-Scale Structure Similarity Index described in equation (4.3), where it is an extension of the SSIM to include multiple channels of the image. MS-SSIM offers better accuracy than the SSIM, but at the higher computing cost.

$$MSSIM(\mathbf{X},\mathbf{Y}) = \frac{1}{M}\sum_{j=1}^{M} SSIM(\mathbf{x}_j,\mathbf{y}_j) \tag{4.3}$$

Moreover, the MSE and Binary Cross Entropy (BCE) loss functions will be used to measure the errors during the training and validation processes of an autoencoder. The MSE loss function described in equation (4.4), where y and ŷ are the ground truth and model prediction respectively and N is the mean of the total dataset samples.

$$\mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{4.4}$$

The BCE described in equation (4.5), where y is the prediction limits based on the color scale, for grayscale, 1 for white points and 0 for black points. Further, p(y) is the predicted probability of the point being white for all N points. The formula shows, that for each white point (y=1), it adds log(p(y)) to the loss, where that is, the logarithmic probability of the point being white. Conversely, the equation adds log(1-p(y)), such is, the log probability of the point being black, for each black point (y=0).

$$\sum_{y,x,d} -c \log_2(\hat{P}(c \mid T)) - (1-c) \log_2(1 - \hat{P}(c \mid T)) \tag{4.5}$$

5 Results and analysis

Experiments 5.1, 5.2, 5.3 and 5.4 focus on implementing and evaluating the autoencoders as software in the loop (SIL). Further, experiment 5.5 elaborate on the implementation and evaluation of autoencoders SIL and the hardware in the loop (HIL) for reliable embedded systems.

5.1 Experiment SIMPA: Simple but Powerful Autoencoder

Regarding image compression, dimensionality reduction is one of the main advantages of autoencoders. Dimensionality reduction is demonstrated using a random sample of data points as per fig 21. The autoencoder is also called a shallow autoencoder when there is only one hidden layer for the encoder and the decoder. Simple autoencoder implemented for visualization purposes of the 3D to 2D dimensionality reduction. Further, the reconstruction of original 3D data from the latent representation is visualized, in this case, the 2D compressed data.

Latent representation elaborated as minimal high-priority features from the original data. For example, if there are two datasets, one is random but consists of only five figures and another dataset that is double in size compared to the first dataset but follows a pattern of squares. The larger dataset is easier to memorize as it follows a defined pattern than the random dataset. By that, the latent space concept is visualized.

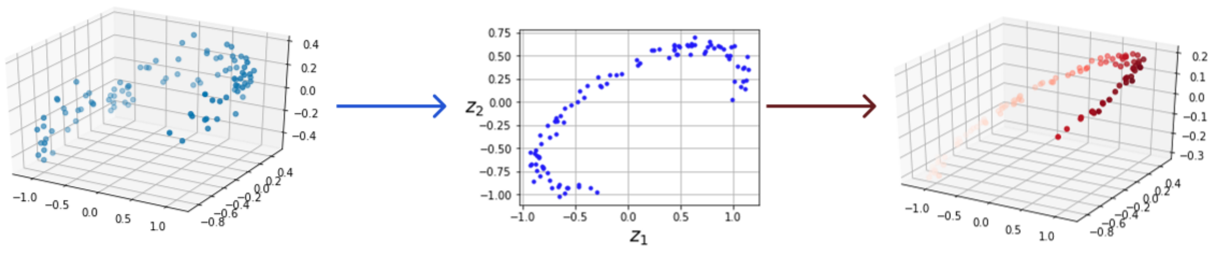
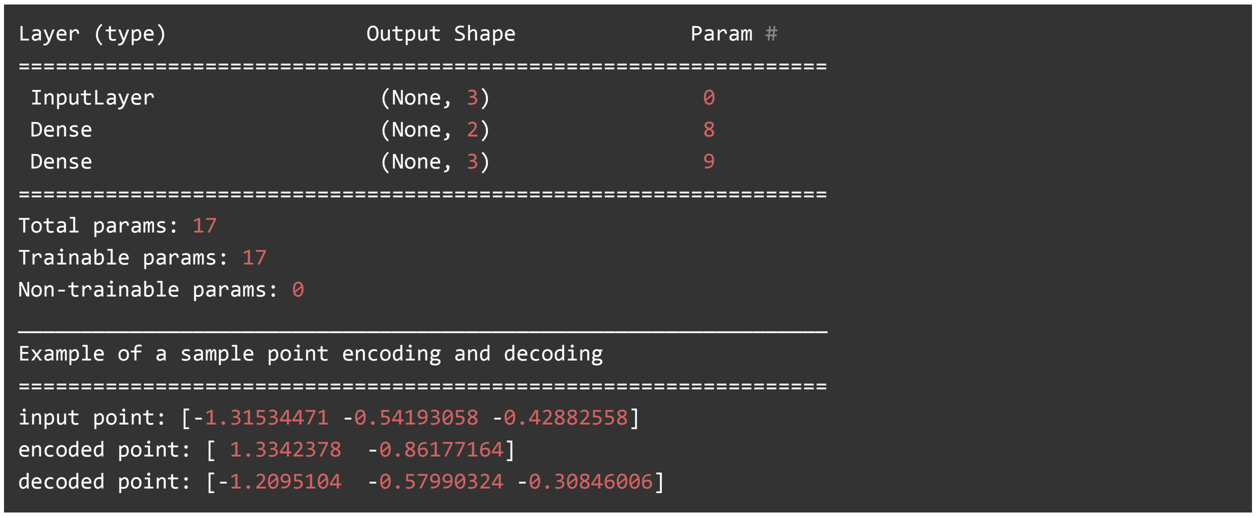


Figure 21. Dimensionality reduction and reconstruction illustration [Source Author]

From this experiment, autoencoders demonstrated significant improvements to be used in lossy compression. From the reconstructed dataset shown on the output image on the right in fig 21, it is clear that the autoencoder has eliminated the noise from the original data input on the left side of fig 21. Thus, in practice, an autoencoder can be used to reduce noise from the dataset and compress the data size.

The autoencoder architecture for this experiment consists of only one hidden layer, with two neurons representing the latent space. The training processes are based on unsupervised learning where the neural network model compiles and fits, trying to reproduce the three neurons' input data back on the output layer. The training and evaluation of the model hired the MSE to assess the accuracy and losses. Further, stochastic gradient descent (SGD) with a learning rate of 1.5 was used to optimize the model weights during the training. The neural network was trained for 200 epochs. Figure 23 shows the detailed architecture of the model, and table 3 shows the parameters.

Table 3. Training parameters



```
Layer (type)                 Output Shape              Param #
=================================================================
 InputLayer                  (None, 3)                 0
 Dense                       (None, 2)                 8
 Dense                       (None, 3)                 9
=================================================================
Total params: 17
Trainable params: 17
Non-trainable params: 0
_________________________________________________________________
Example of a sample point encoding and decoding
=================================================================
input point: [-1.31534471 -0.54193058 -0.42882558]
encoded point: [ 1.3342378  -0.86177164]
decoded point: [-1.2095104  -0.57990324 -0.30846006]
```

In the shape column, the term None appears, which mean the neural network can be trained on variable batches of data. In this experiment, the data input are 3 columns of data with 100 rows each. The columns here represent the 3 dimensions and the 100 rows are the date batches which represented by the term None. Further, None here refers that the model architecture still valid if the data batches increased, for example 300 instead of 100. Figure 22 shows the plotted training and validation losses.

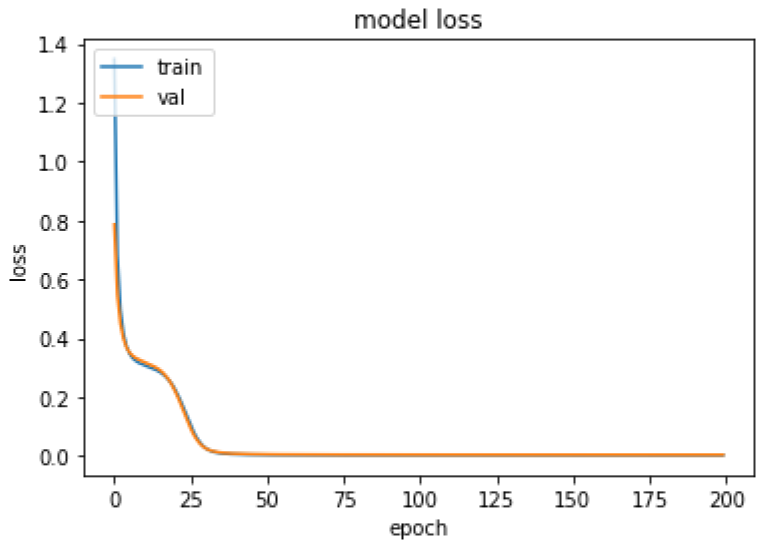


Figure 22. Losses during the training and validation of the simple autoencoder

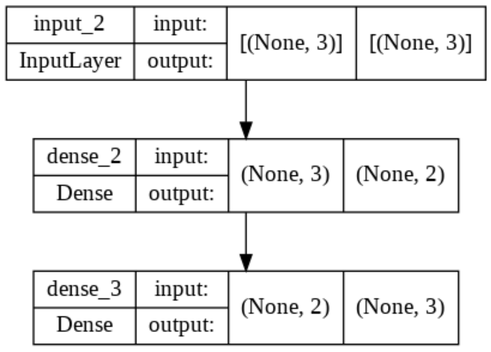


Figure 23. Simple autoencoder architecture

This experiment's results led to an evaluation of a simple but powerful autoencoder. That is because this autoencoder can quickly decode data points, which could represent telecommunication signals as an example. With such power, generic random encryption shall be reached to the data, mitigating the spoofing and Man in The Middle (MITM) issues [109].

5.2 Experiment BIRIA: BIROS images autoencoder

This experiment discusses two high-resolution BIROS compression autoencoders. The first autoencoder reaches a compression factor of 16:1, and the second reaches a 64:1 compression ratio. Both autoencoders hire the patching pre-processing and post-processed to accelerate the compression and decompressing. Patches samples and decompressed images are illustrated in the following figure.

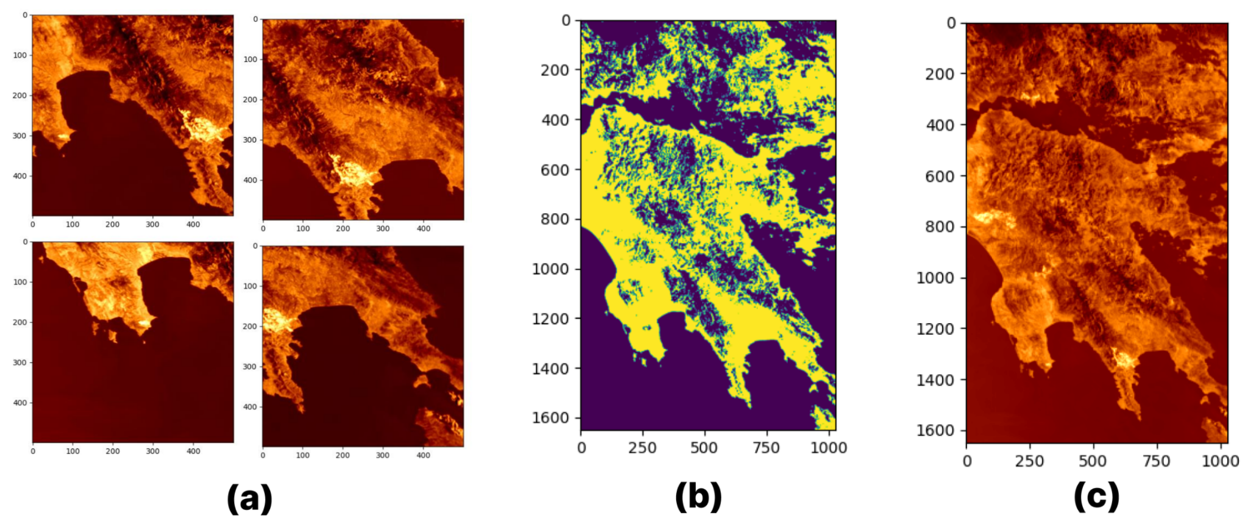


Figure 24. Visual results from BIROS image, location is Greece, a) sample of patches during the preprocessing, b) image with binary mask, c) reconstructed image. [Source: DLR, BIROS]

The figure above showed images a and c, which are infrared images, a red-heat color mapping used to highlight the visual illustration. The experiment focused on using Medium Wave InfraRed (MWIR) images obtained from the L1B level of processing of the BIROS satellite sensors. Typical DLR sensor data processing levels are illustrated in the following figure.

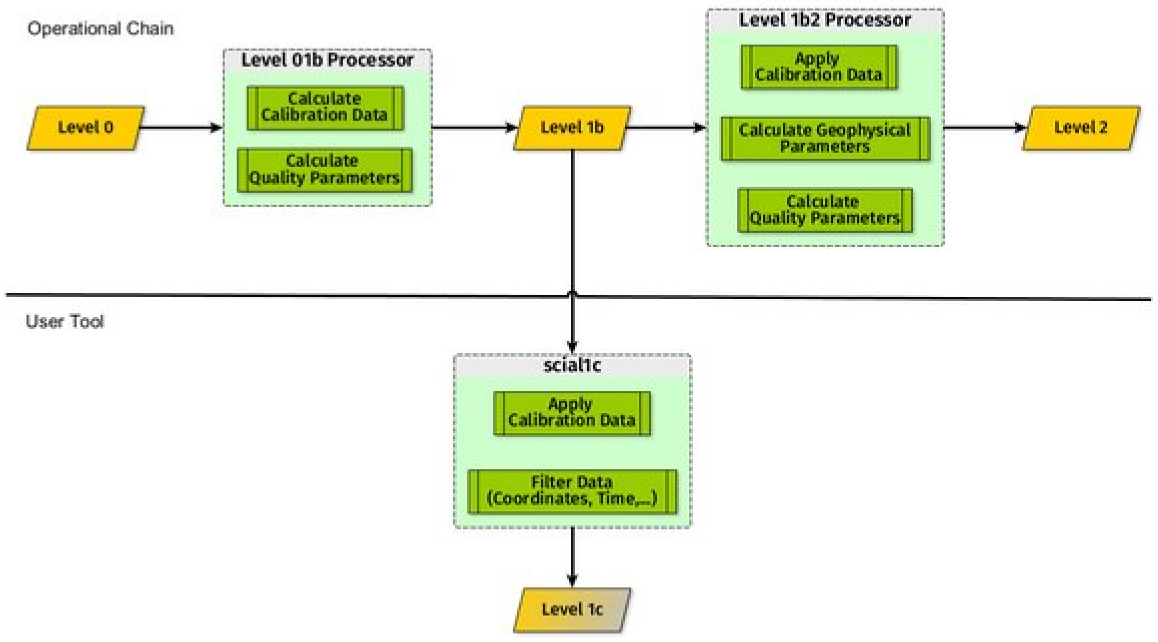

Operational Chain
Level 0
Level 01b Processor
Calculate Calibration Data
Calculate Quality Parameters
Level 1b
Level 1b2 Processor
Apply Calibration Data
Calculate Geophysical Parameters
Calculate Quality Parameters
Level 2
User Tool
scial1c
Apply Calibration Data
Filter Data (Coordinates, Time,...)
Level 1c

Figure 25. Illustration of typical sensor processing levels [122]. [Source: DLR, IMF-ATP]

Both BIRIA experiments use max-pooling and up-sampling layers among the convolutional layers to achieve the results. Both max-pooling and up-sampling layers are layers with no weights. Max-pooling is an input drive operation that learns to half the input dimensions [154]. Thus, the output from the max-pooling layers is smaller than the input. The max-pooling layer moves over the input, similar to the convolutional layer, with a predefined stride [155]. The following figure shows an illustration of the max-pooling and up-sampling processes.

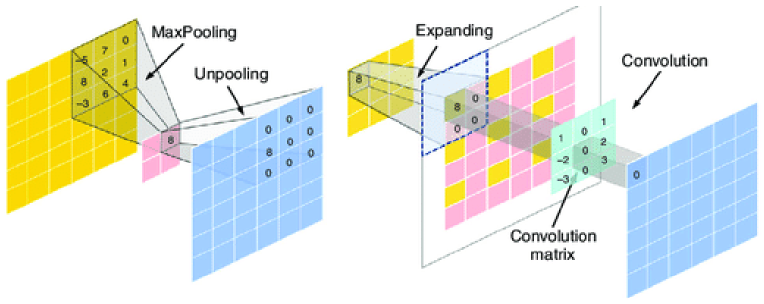


Figure 26. Illustration of the convolution, up-sampling and max-pooling. [Source: ResearchGate [153]]

Further, max-pooling is meant to select the maximum values within the stride and collect it to the output. The up-sampling layer does the opposite of the max-pooling layer; it works on sampling from the convolutional output and decomposes the values to a larger shape as an output from its stride [156]. The up-sampling layer doubles the input dimensions and is followed by a convolutional layer. Furthermore, the up-sampling layer combined with the convolutional layer is commonly known as deconvolution, inverse convolution, or transpose convolution [157, 159]. The transpose convolutional layer does both up-sampling the input and learning how to fill in details during the model training process [158]. Further illustrations about the max-pooling and the up-sampling are listed with code examples in appendix B.

5.2.1 Autoencoder for BIROS image compression

This part of the BIRIA experiment focused on implementing an autoencoder based on 10 hidden layers. The autoencoder succeeded in compressing the BIROS images with a 16:1 compression ratio factor. Further, the architecture of this autoencoder is built as a convolutional autoencoder. The encoder consists of 3 convolutional layers and 3 max-pooling layers, leading to a bottleneck that is 16 times smaller than the input size. The decoder consists of 3 convolutional transpose layers and 2 up-sampling layers. The 10-layers-autoencoder is trained to accept an unlimited number of inputs. Moreover, 97 images were pre-processed and prepared as a dataset for this model. The dataset is split into 80% for training and 20% for testing and evaluation. This model accepts input shapes of 500 × 500 pixels; thus, the full BIROS images are patched to fit the required input shape. The following figure shows the training versus validation losses, where the curve exponentially converges and saturates under 5e-4.

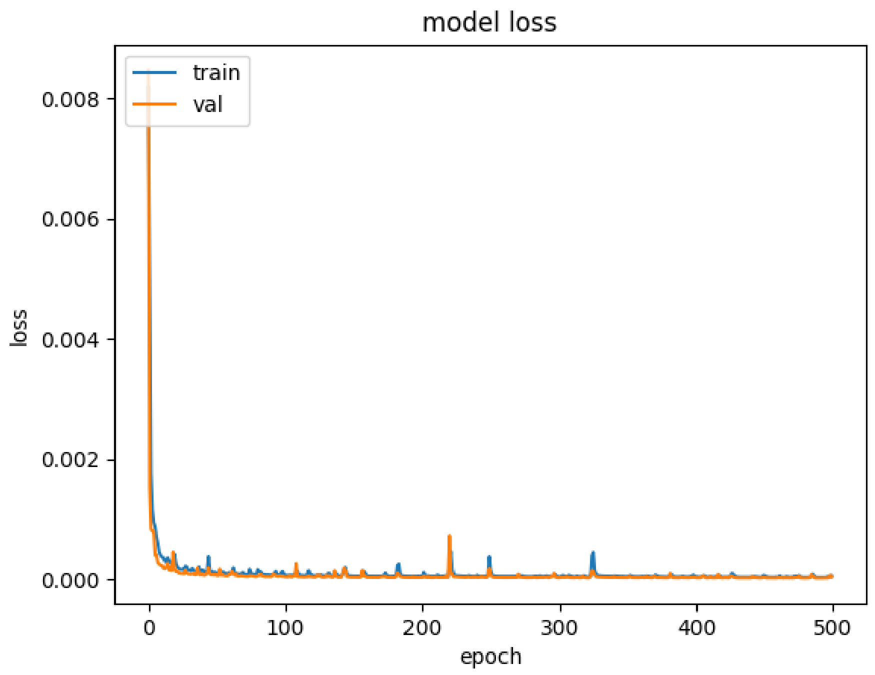


Figure 27. Losses during the training and validation of the 10-layers-autoencoder. [Source: Author]

The following figure demonstrates the encoding and decoding processes using 10 samples of the test dataset. The encoded latent space visualization is demonstrated in the shape of 125 × 125, then reconstructed to the 500 × 500 original shape.

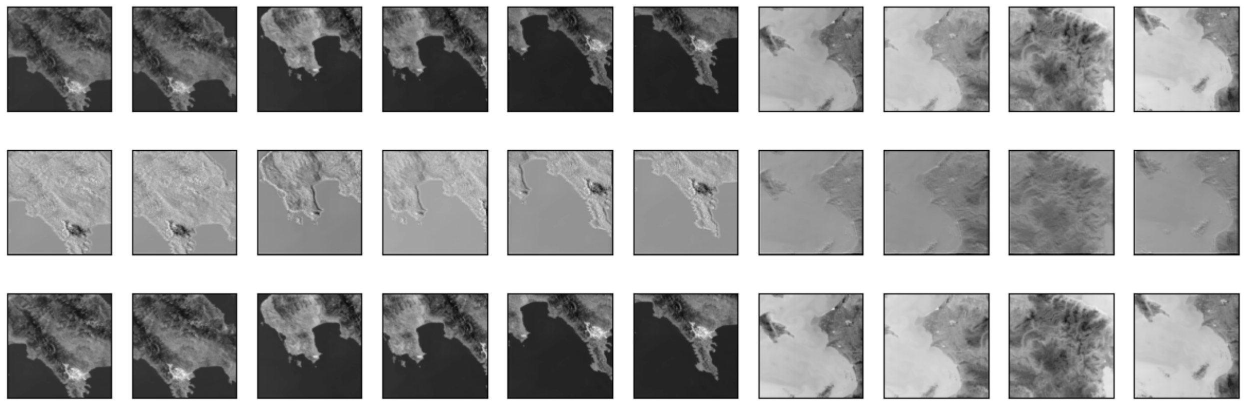
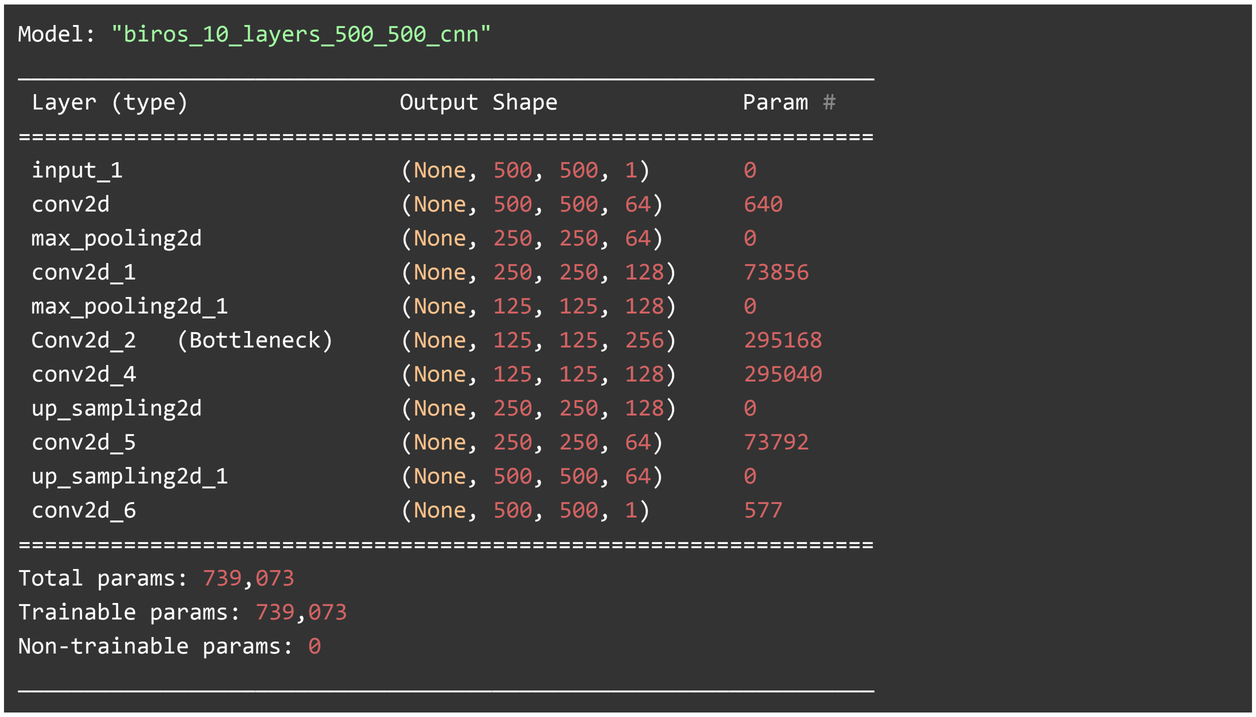


Figure 28. The 10-layers-autoencoder results, from top to bottom, input, encoded, decoded. [Source: Author]

Further, the adaptive moment estimation (Adam) optimizer algorithm was hired to dynamically optimize and tune the model weights during the training process [151]. The autoencoder trained for 500 epochs based on the newly prepared BIROS dataset. Figure 29 shows the detailed architecture of the model, and table 4 shows the parameters. The None term in the parameter table indicates that the autoencoder architecture can be trained and tested on an indefinite batch of input images as long as the input images are in the accepted input shape in the first layer. Furthermore, the ReLU activation function is used in the hidden layers, and the Sigmoid function is used before the decoder output.

Table 4. Parameters of the 10-layers-autoencoder



```
Model: "biros_10_layers_500_500_cnn"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1                     (None, 500, 500, 1)       0
 conv2d                      (None, 500, 500, 64)      640
 max_pooling2d               (None, 250, 250, 64)      0
 conv2d_1                    (None, 250, 250, 128)     73856
 max_pooling2d_1             (None, 125, 125, 128)     0
 Conv2d_2   (Bottleneck)     (None, 125, 125, 256)     295168
 conv2d_4                    (None, 125, 125, 128)     295040
 up_sampling2d               (None, 250, 250, 128)     0
 conv2d_5                    (None, 250, 250, 64)      73792
 up_sampling2d_1             (None, 500, 500, 64)      0
 conv2d_6                    (None, 500, 500, 1)       577
=================================================================
Total params: 739,073
Trainable params: 739,073
Non-trainable params: 0

_________________________________________________________________
```

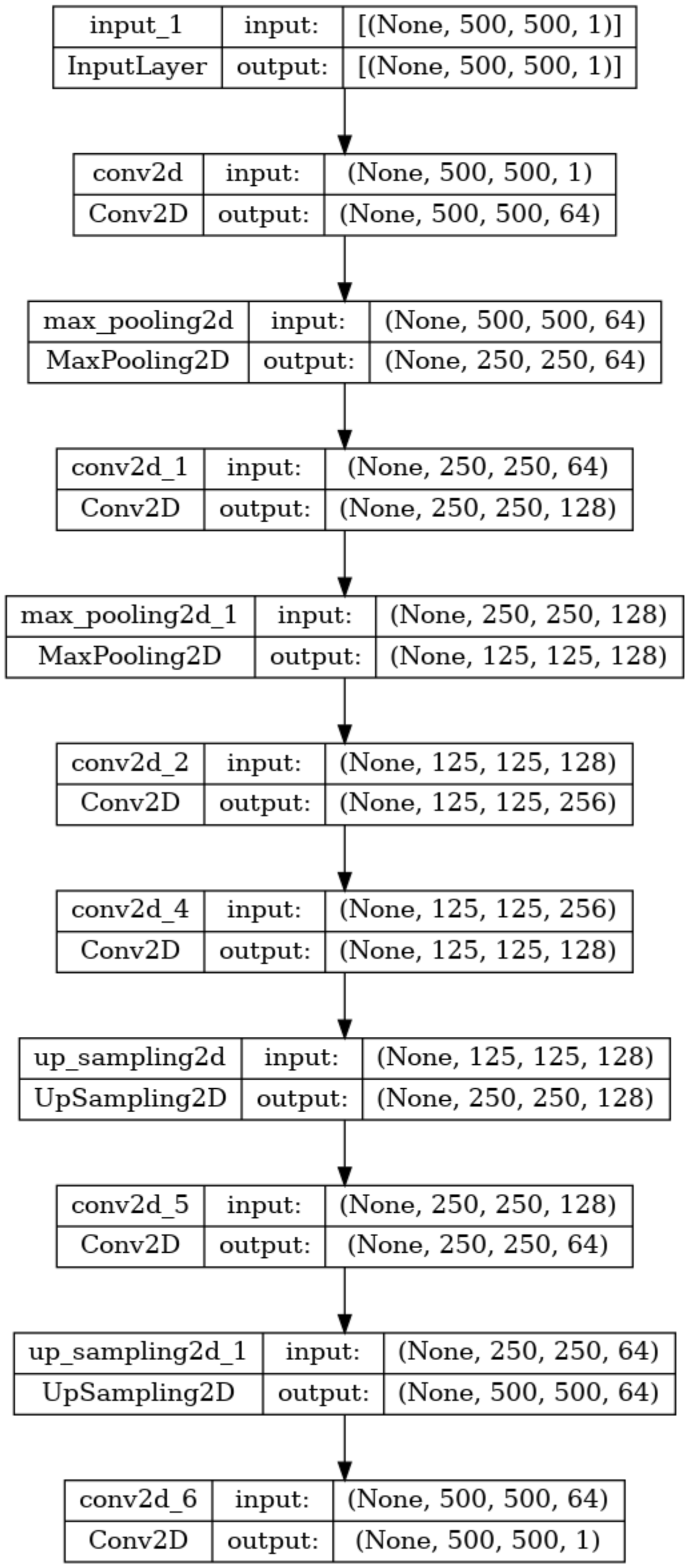

input_1 | InputLayer | input: [(None, 500, 500, 1)] | output: [(None, 500, 500, 1)]
conv2d | Conv2D | input: (None, 500, 500, 1) | output: (None, 500, 500, 64)
max_pooling2d | MaxPooling2D | input: (None, 500, 500, 64) | output: (None, 250, 250, 64)
conv2d_1 | Conv2D | input: (None, 250, 250, 64) | output: (None, 250, 250, 128)
max_pooling2d_1 | MaxPooling2D | input: (None, 250, 250, 128) | output: (None, 125, 125, 128)
conv2d_2 | Conv2D | input: (None, 125, 125, 128) | output: (None, 125, 125, 256)
conv2d_4 | Conv2D | input: (None, 125, 125, 256) | output: (None, 125, 125, 128)
up_sampling2d | UpSampling2D | input: (None, 125, 125, 128) | output: (None, 250, 250, 128)
conv2d_5 | Conv2D | input: (None, 250, 250, 128) | output: (None, 250, 250, 64)
up_sampling2d_1 | UpSampling2D | input: (None, 250, 250, 64) | output: (None, 500, 500, 64)
conv2d_6 | Conv2D | input: (None, 500, 500, 64) | output: (None, 500, 500, 1)

Figure 29. The 10-layers-autoencoder architecture. [Source: Author]

Table 5 describes the pseudocode used to build the convolutional autoencoder. All convolutional kernels are set to 3 × 3 shape, and the padding stays the same through the whole input. The max-pooling process is applied to half the size for each layer during encoding after each convolutional layer. The up-sampling layer is set to double the previous input size, then passes the output to the convolutional transpose layer.

Table 5. Pseudocode of the 10-layers-autoencoder

```
Encoder function (inputs){
  conv_1 = Conv2D(filters=64, kernel_size=(3,3), activation='relu',
                  padding='same')(inputs)
  max_pool_1 = MaxPooling2D(pool_size=(2,2))(conv_1)
  conv_2 = Conv2D(filters=128, kernel_size=(3,3), activation='relu',
                  padding='same')(max_pool_1)
  max_pool_2 = MaxPooling2D(pool_size=(2,2))(conv_2)
  return max_pool_2
}

Bottleneck function(inputs){
  encoder_output = encoder(inputs)
  bottleneck = Conv2D(filters=256, kernel_size=(3,3), activation='relu',
                       padding='same')(encoder_output)

  return bottleneck
}

Decoder function (inputs){
  conv_1 = Conv2D(filters=128, kernel_size=(3,3),
                  activation='relu', padding='same')(inputs)
  up_sample_1 = UpSampling2D(size=(2,2))(conv_1)
  conv_2 = Conv2D(filters=64, kernel_size=(3,3),
                  activation='relu', padding='same')(up_sample_1)
  up_sample_2 = UpSampling2D(size=(2,2))(conv_2)
  conv_3 = Conv2D(filters=1, kernel_size=(3,3), activation='sigmoid',
                  padding='same')(up_sample_2)
  return conv_3
}

Autoencoder function {
  inputs = Input(shape=(500, 500, 1,))
  bottleneck_output = bottleneck(inputs)
  decoder_output = decoder(bottleneck)

  model = Model(inputs =inputs, outputs=decoder_output)
  encoder_model = Model(inputs=inputs, outputs=bottleneck_output)
  decoder_model = Model(inputs=bottleneck_output, outputs=decoder_output)
  return model, encoder_model, decoder_model
}
```

5.2.2 Deep Autoencoder for BIROS extreme image compression

The second part of the BIRIA experiment focused on implementing an autoencoder based on 14 hidden layers. The autoencoder succeeded in compressing the BIROS images with a 64:1 compression ratio. Further, the architecture of this autoencoder is built as a convolutional autoencoder as well. The encoder consists of 4 convolutional layers and 3 max-pooling layers, leading to a bottleneck that is 64 times smaller than the input size, with MS-SSIM higher than 91%. The decoder consists of 4 convolutional transpose layers and 3 up-sampling layers. The 14-layers-autoencoder is trained to accept an unlimited number of inputs.

Moreover, 396 images were pre-processed and prepared as a dataset for this model. The dataset is split into 80% for training and 20% for testing and evaluation. This model accepts input shapes of 256 × 256 pixels; thus, the full BIROS images are patched to fit the required input shape. The following figure shows the training versus validation losses, where the curve exponentially converges and saturates under 7e-5.

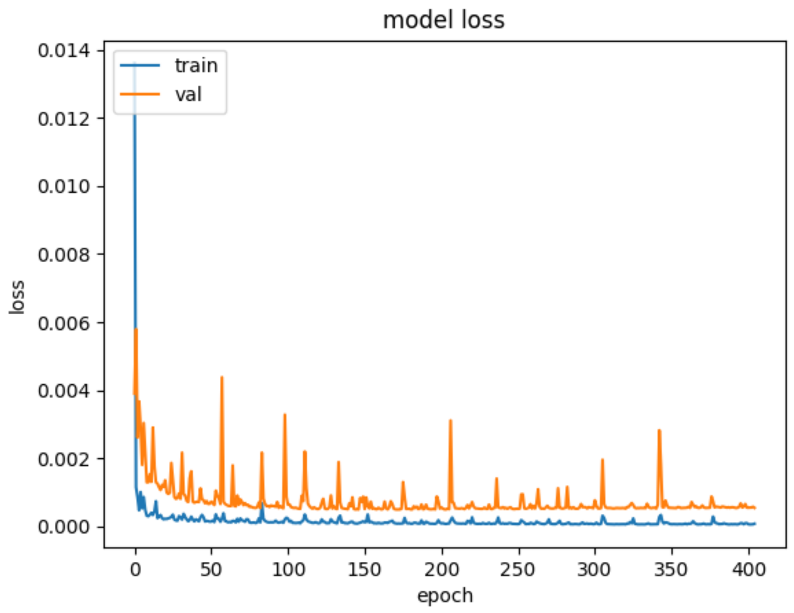


Figure 30. Losses during the training and validation of the 14-layers-autoencoder [Source: Author]

The spikes in the losses graph reflect limitations from the dataset images; this can be smoothed by increasing the dataset size with various images and increasing the training epochs. The following figure demonstrates the encoding and decoding processes using 10 samples of the test dataset. The encoded latent space visualization is demonstrated in the shape of 32 × 32, then reconstructed to the 256 × 256 original shape.

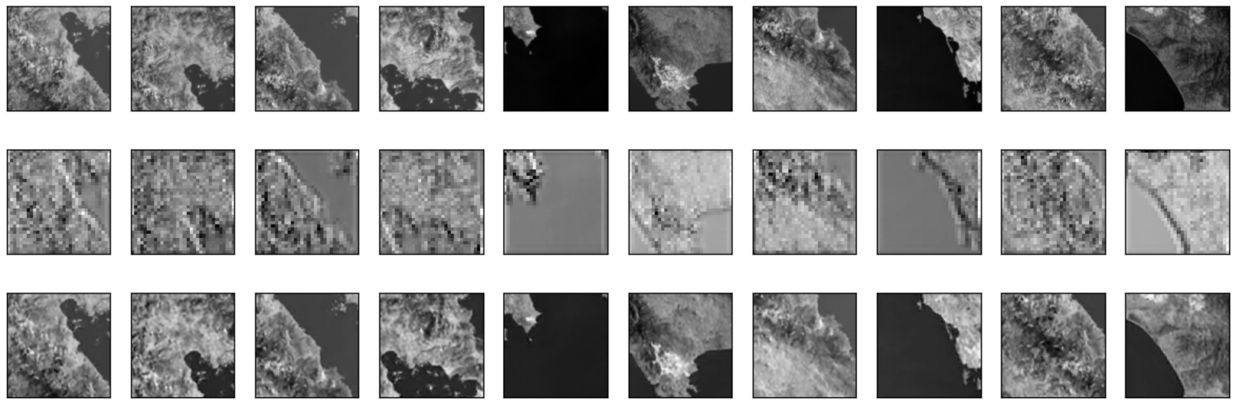
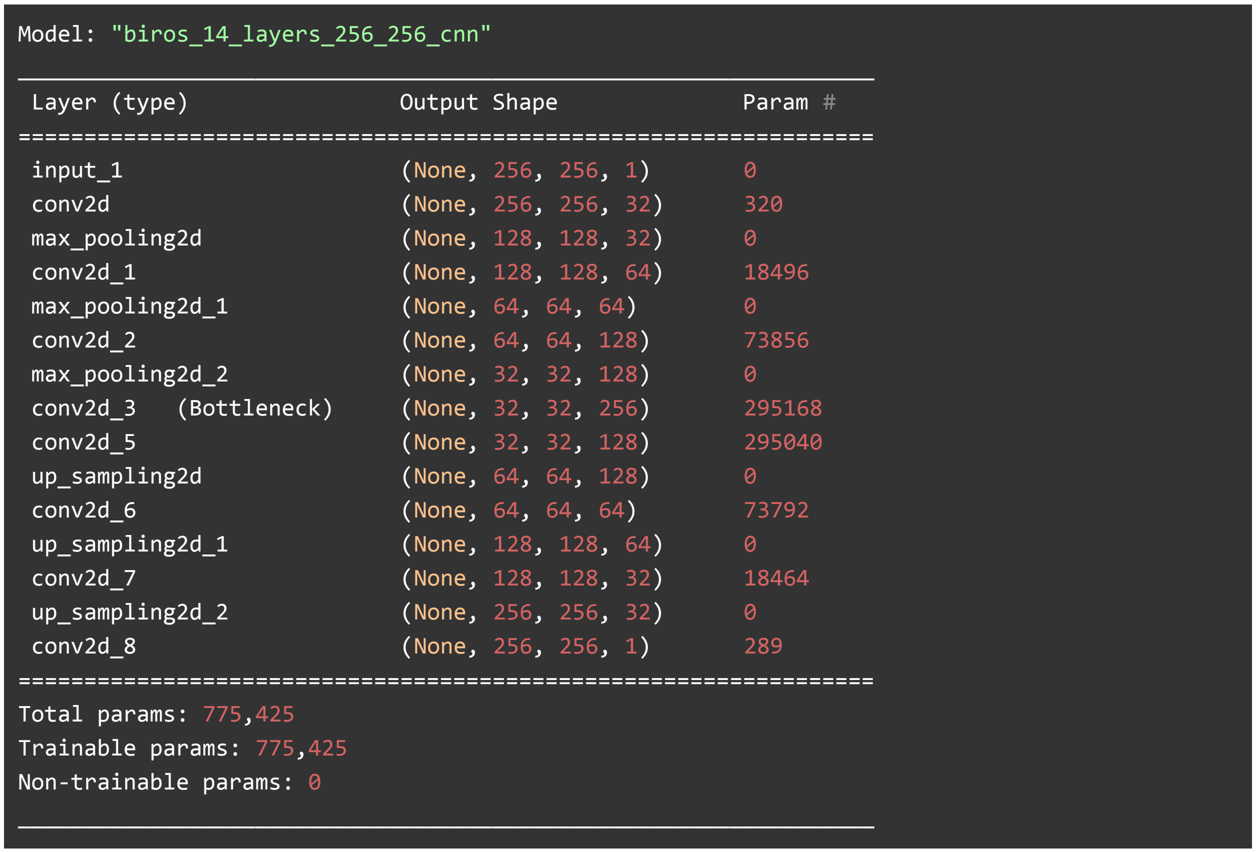


Figure 31. The 14-layers-autoencoder results, from top to bottom, input, encoded, decoded [Source: Author]

Further, the Adam optimizer was used to dynamically optimize and tune the model weights during the training process. The autoencoder trained for 500 epochs based on the newly prepared BIROS dataset. The None term in the parameter table indicates that the autoencoder architecture can be trained and tested on an indefinite batch of input images as long as the input images are in the accepted input shape in the first layer. Furthermore, the ReLU activation function is used in the hidden layers, and the Sigmoid function is used before the decoder output.

Table 6. Parameters of the 14-layers-autoencoder



```
Model: "biros_14_layers_256_256_cnn"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1                     (None, 256, 256, 1)       0
 conv2d                      (None, 256, 256, 32)      320
 max_pooling2d               (None, 128, 128, 32)      0
 conv2d_1                    (None, 128, 128, 64)      18496
 max_pooling2d_1             (None, 64, 64, 64)        0
 conv2d_2                    (None, 64, 64, 128)       73856
 max_pooling2d_2             (None, 32, 32, 128)       0
 conv2d_3   (Bottleneck)     (None, 32, 32, 256)       295168
 conv2d_5                    (None, 32, 32, 128)       295040
 up_sampling2d               (None, 64, 64, 128)       0
 conv2d_6                    (None, 64, 64, 64)        73792
 up_sampling2d_1             (None, 128, 128, 64)      0
 conv2d_7                    (None, 128, 128, 32)      18464
 up_sampling2d_2             (None, 256, 256, 32)      0
 conv2d_8                    (None, 256, 256, 1)       289
=================================================================
Total params: 775,425
Trainable params: 775,425
Non-trainable params: 0

_________________________________________________________________
```

Table 7 describes the pseudocode used to build the convolutional autoencoder. All convolutional kernels are set to 3 × 3 shape, and the padding stays the same through the whole input. The max-pooling process is applied to half the size for each layer during encoding after each convolutional layer. The up-sampling layer is set to double the previous input size, then passes the output to the convolutional transpose layer.

Table 7. Pseudocode of the 14-layers-autoencoder

```
Encoder function (inputs){
  conv_0 = Conv2D(filters=32, kernel_size=(3,3), activation='relu', padding='same')(inputs)
  max_pool_0 = MaxPooling2D(pool_size=(2,2))(conv_0)

  conv_1 = Conv2D(filters=64, kernel_size=(3,3), activation='relu', padding='same')(max_pool_0)
  max_pool_1 = MaxPooling2D(pool_size=(2,2))(conv_1)

  conv_2 = Conv2D(filters=128, kernel_size=(3,3), activation='relu', padding='same')(max_pool_1)
  max_pool_2 = MaxPooling2D(pool_size=(2,2))(conv_2)
  return max_pool_2
}

Bottleneck function(inputs){
  encoder_output = encoder(inputs)
  bottleneck = Conv2D(filters=256, kernel_size=(3,3), activation='relu',
                      padding='same')(encoder_output)
  return bottleneck
}

Decoder function (inputs){
  conv = Conv2D(filters=128, kernel_size=(3,3), activation='relu',
                                          padding='same')(inputs)
  up_sample = UpSampling2D(size=(2,2))(conv)
  conv_0 = Conv2D(filters=64, kernel_size=(3,3), activation='relu',
                                          padding='same')(up_sample)
  up_sample_0 = UpSampling2D(size=(2,2))(conv_0)
  conv_1 = Conv2D(filters=32, kernel_size=(3,3), activation='relu',
                                          padding='same')(up_sample_0)
  up_sample_1 = UpSampling2D(size=(2,2))(conv_1)
  conv_3 = Conv2D(filters=1, kernel_size=(3,3), activation='sigmoid',
                                          padding='same')(up_sample_1)
  return conv_3
}

Autoencoder function {
  inputs = Input(shape=(256, 256, 1,))
  bottleneck_output = bottleneck(inputs)
  decoder_output = decoder(bottleneck)

  model = Model(inputs =inputs, outputs=decoder_output)
  encoder_model = Model(inputs=inputs, outputs=bottleneck_output)
  decoder_model = Model(inputs=bottleneck_output, outputs=decoder_output)
  return model, encoder_model, decoder_model
}
```

The BIROS dataset preparation process is described in figure 32. After collecting BIROS images, they passed to the patch maker module to slice them into 256 × 256 patches and store them in a dataset stack of images for training or testing. The dataset must be split into training and testing datasets before the autoencoder training. The implemented patch maker module is a module class using native and open-source code. The open-source code used is optimized to generically adapt to any shape of the raw input images from BIROS [162].

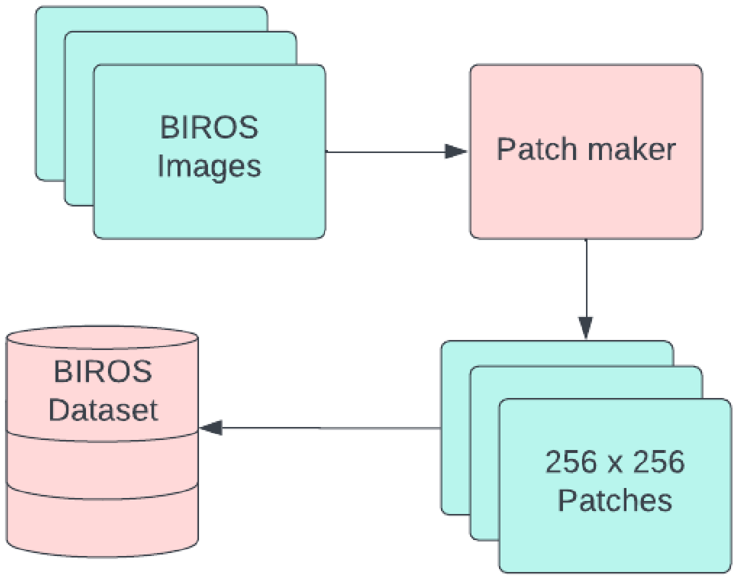


Figure 32. A Block diagram about BIROS dataset preparation process. [Source: Author]

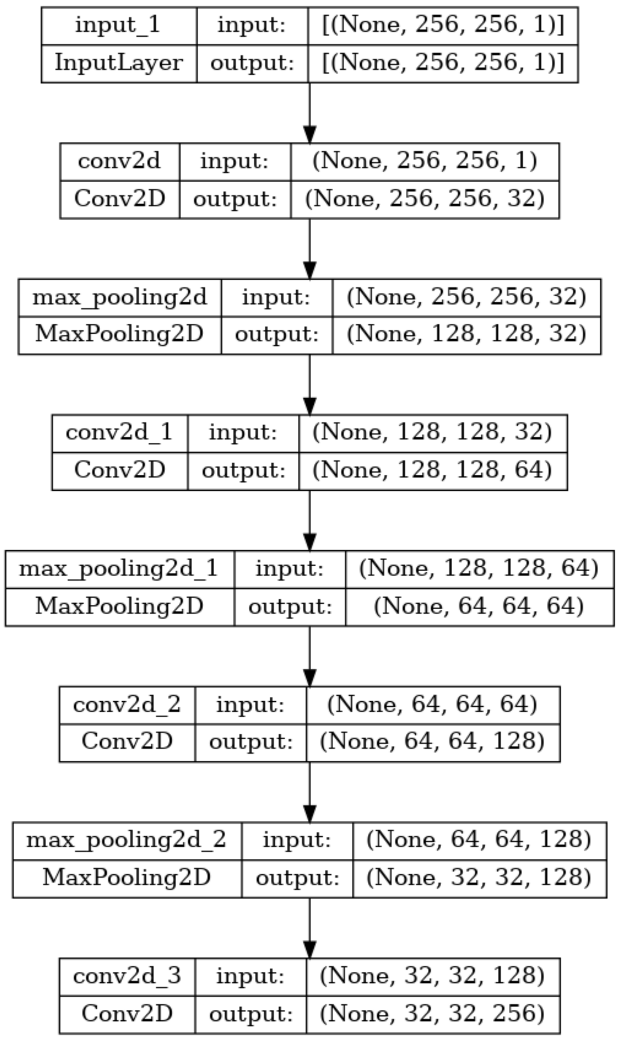

input_1 | input: | [(None, 256, 256, 1)]
InputLayer | output: | [(None, 256, 256, 1)]
conv2d | input: | (None, 256, 256, 1)
Conv2D | output: | (None, 256, 256, 32)
max_pooling2d | input: | (None, 256, 256, 32)
MaxPooling2D | output: | (None, 128, 128, 32)
conv2d_1 | input: | (None, 128, 128, 32)
Conv2D | output: | (None, 128, 128, 64)
max_pooling2d_1 | input: | (None, 128, 128, 64)
MaxPooling2D | output: | (None, 64, 64, 64)
conv2d_2 | input: | (None, 64, 64, 64)
Conv2D | output: | (None, 64, 64, 128)
max_pooling2d_2 | input: | (None, 64, 64, 128)
MaxPooling2D | output: | (None, 32, 32, 128)
conv2d_3 | input: | (None, 32, 32, 128)
Conv2D | output: | (None, 32, 32, 256)

Figure 33. The 14-layers-autoencoder modular encoder architecture. [Source: Author]

The autoencoder is designed such that it can be decomposed and used in a modular form. So, the encoder and the decoder can be used separately.

Figure 34 shows the compression processes using the 14-layers-autoencoder. The input image with its original shape is passed to the patch maker module to be sliced into equally produced 256 × 256 image patches. Suppose the input image does not fit into equally sliced patches. In that case, the patch maker is designed to repeat a few patches dynamically to ensure the reconstruction succeeds by applying equation (5.1) to predict the minimal number of patches. In equation (5.1) ϕ is the input image width, β is the patch width and θ is the stride's step size. After producing the patches out of the input image, they get passed to the encoder or multiple encoders in case of heterogeneous programming. Following the encoder, each encoded patch in the latent space is stored in an N-dimensional stack of the compressed images. The stack of compressed images stores the latent representation of all images in the exact position of their location in the N-dimensional stack before encoding them. The N-dimensional stack size is dynamically defined based on the input image size and the patching processes in the patch maker module.

$$(\phi - \beta) \% (\theta) = 0 \tag{5.1}$$

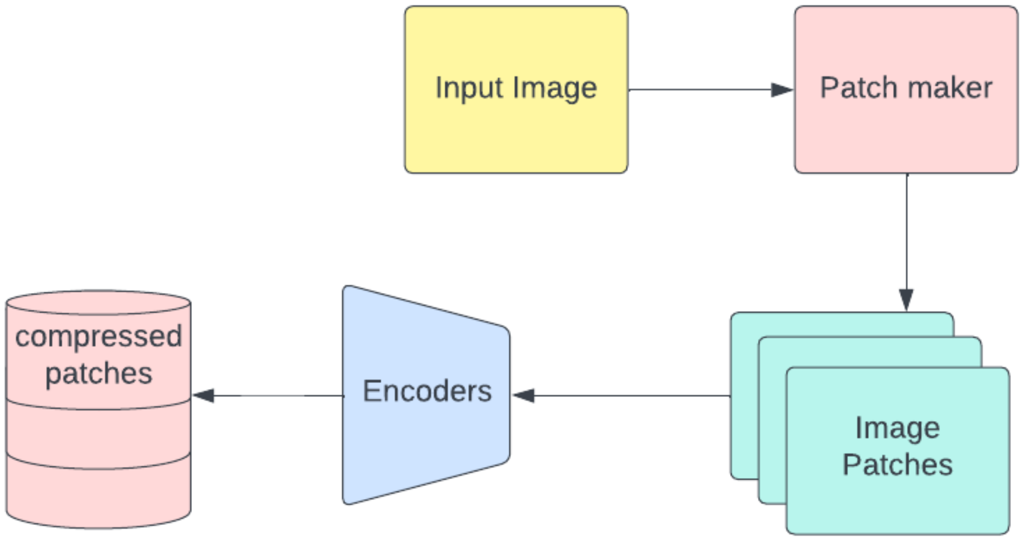
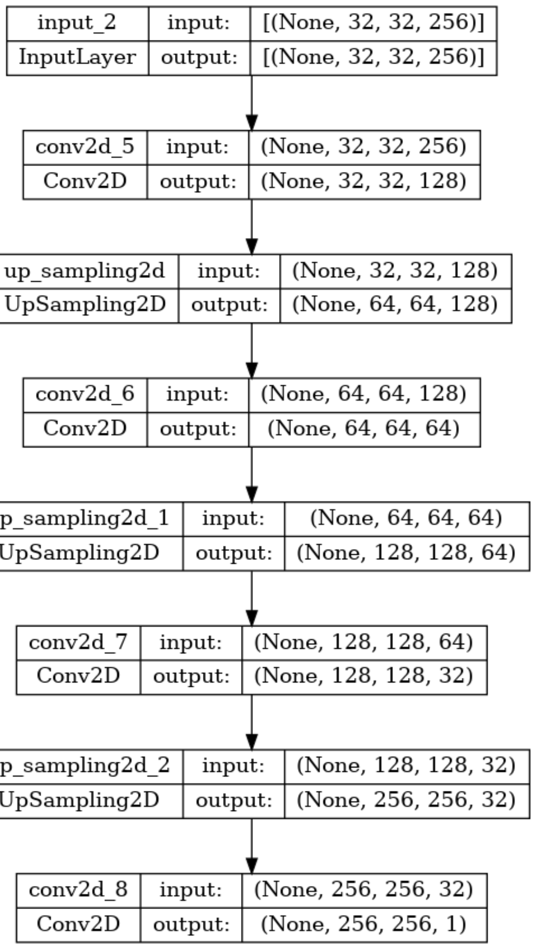


Figure 34. A block diagram illustrates the compression phase. [Source: Author]



input_2 | input: | [(None, 32, 32, 256)]
InputLayer | output: | [(None, 32, 32, 256)]
conv2d_5 | input: | (None, 32, 32, 256)
Conv2D | output: | (None, 32, 32, 128)
up_sampling2d | input: | (None, 32, 32, 128)
UpSampling2D | output: | (None, 64, 64, 128)
conv2d_6 | input: | (None, 64, 64, 128)
Conv2D | output: | (None, 64, 64, 64)
up_sampling2d_1 | input: | (None, 64, 64, 64)
UpSampling2D | output: | (None, 128, 128, 64)
conv2d_7 | input: | (None, 128, 128, 64)
Conv2D | output: | (None, 128, 128, 32)
up_sampling2d_2 | input: | (None, 128, 128, 32)
UpSampling2D | output: | (None, 256, 256, 32)
conv2d_8 | input: | (None, 256, 256, 32)
Conv2D | output: | (None, 256, 256, 1)

Figure 35. The 14-layers-autoencoder modular decoder architecture. [Source: Author]

Figure 35. shows the decoder architecture separated based on the 14-layers-autoencoder. Figure 36 shows the decoding and reconstruction process of the image, which is a reverse process compared to figure 34.

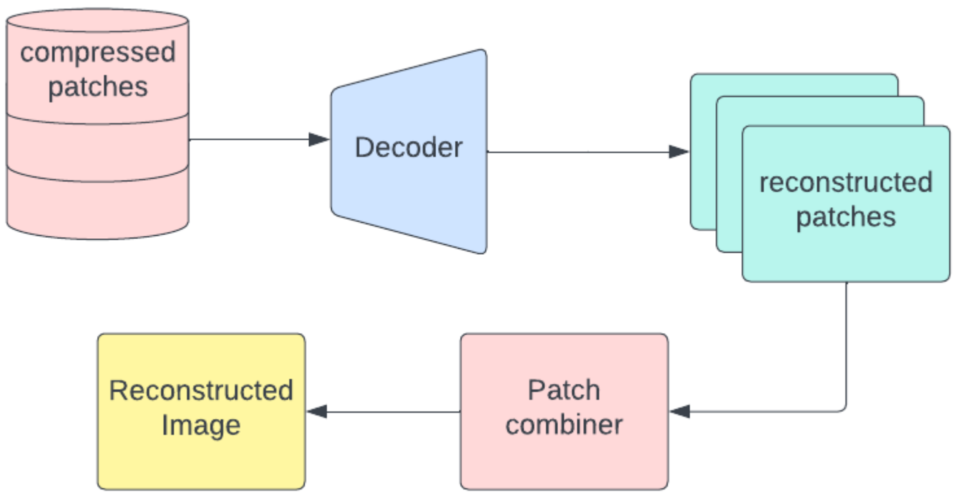


Figure 36. A block diagram for the decoding and reconstruction phases. [Source: Author]

The BIRIA experiment showed significant results for the autoencoders. Results can be improved by extending the autoencoder training with larger datasets. Further, the autoencoder architecture can be extended and generalized for 2D and 3D images. The standard three-color channels and hyperspectral color channels as explained in the following experiments.

5.3 Experiment MPSIA: MACS and public satellite images Autoencoders

MPSIA experiment on MACS images and public satellite images. Further, instead of implementing an autoencoder from scratch, pre-trained models will be adapted to the satellite and remote image compression without fully retraining the autoencoders. In addition to the advancement from the previous experiment, two autoencoder architectures evaluated in this experiment, the first is based on RNN and Long Short-Term Memory (RNN) and the second is based on Generative Adversarial Network (GAN). The LSTM model is based on the residual Gated Recurrent Unit (GRU), which use RNN-based autoencoder to compress and reconstruct the images with different compression rates using the same model [116]. The GAN model is about combining Generative Adversarial Networks with learned compression to obtain a generative compression system [117].

5.3.1 Transfer learning: Modern deep learning paradigm

Considering the modern deep learning paradigm, training neural networks from scratch is not the optimal solutions for complex applications. Instead, transfer learning is the solution [123]. Transfer learning and fine-tuning pre-trained models is optimal to accelerate application customization, especially when the pre-trained models are significantly efficient and trained on large datasets. The neural networks can rely on pre-trained models, with small modification applied on their architecture. For example, Residual Neural Network is one of the advanced and commonly used neural networks, retraining such a neural network would require long time and hyper computing power. Otherwise, developing new neural network on top of it is the optimal solution, that by retraining only partial part of it, like unfreezing the trained model and retrain only one or a couple of layers from. The partial training approach accelerates the processes of producing new effective neural networks.

5.3.2 LSTM: Long Short-Term Memory based Compression Autoencoder

The RNN architecture permits controlling the amount of information learned from the data: it increases at every shot, and a well-observed trade-off is found between the more substantial compression and a better reconstruction, tuning the reconstruction bit rate. The model was initially evaluated on Kodak uncompressed dataset images [118]. The residual GRU is a one-shot additive reconstruction network. BCE loss function was used to assess the model accuracy. The model trained on 1 million training steps.

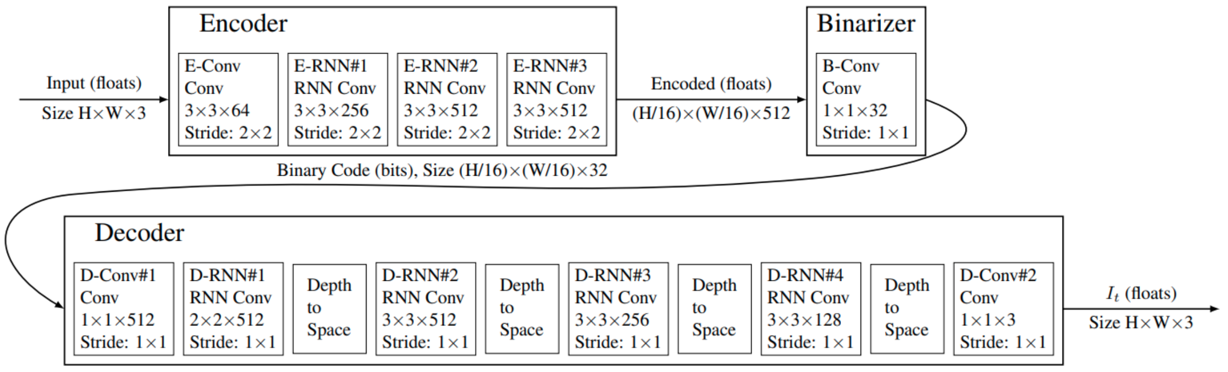

Encoder
Input (floats)
Size H×W×3
E-Conv Conv 3×3×64 Stride: 2×2
E-RNN#1 RNN Conv 3×3×256 Stride: 2×2
E-RNN#2 RNN Conv 3×3×512 Stride: 2×2
E-RNN#3 RNN Conv 3×3×512 Stride: 2×2
Encoded (floats)
(H/16)×(W/16)×512
Binarizer
B-Conv Conv 1×1×32 Stride: 1×1
Binary Code (bits), Size (H/16)×(W/16)×32
Decoder
D-Conv#1 Conv 1×1×512 Stride: 1×1
D-RNN#1 RNN Conv 2×2×512 Stride: 1×1
Depth to Space
D-RNN#2 RNN Conv 3×3×512 Stride: 1×1
Depth to Space
D-RNN#3 RNN Conv 3×3×256 Stride: 1×1
Depth to Space
D-RNN#4 RNN Conv 3×3×128 Stride: 1×1
Depth to Space
D-Conv#2 Conv 1×1×3 Stride: 1×1
I_t (floats)
Size H×W×3

Figure 37. Single shot iteration of the RNN Autoencoder [Source: ResearchGate [116]]

Figure 37 illustrates that the encoder (E) and the decoder (D) are RNNs optimized with residual error. The single-shot strategy of the autoencoder, including the binarizer (B), which is Entropy coding binary, can be described by equation (5.2). The encoder is followed by the binarizer, which is similar to the BCE function (equation 4.5), to do Entropy coding on the pixels, thus, storing the input data with a minimal number of bits in latent representation. Later, the decoder can reconstruct the original image from the binarized probabilistic latent representation.

$$\hat{x}_t = D_t(B(E_t(x - \hat{x}_{t-1}) \quad (5.2)$$

This is a progressive reconstruction of the original image $\boldsymbol{x}$. Each iteration populates more bits generated from the encoder and leads to a significant improvement during the reconstruction. Thus, each successful iteration adds to the previous estimation of the original image. The following equation can describe the additive reconstruction.

$$\hat{x}_t = D_t(B(E_t(x - \hat{x}_{t-1}) + x_{t-1} \tag{5.3}$$

Table 8 shows a snapshot of the RNN-LSTM-based model comparison with JPEG compression on compression based on the image SSIM. Comprehensive qualitative comparison between LSTM- and GAN-based compression autoencoder and the Consultative Committee for Space Data Systems (CCSDS) standards elaborated by the end of this experiment.

Table 8. Comparison be LSTM image compression and JPEG

Compression method	SSIM
LSTM	0.8091
JPEG	0.7748

Table 9 compares the BPP, the compression ratio, and its evolution over the additive reconstruction.

Table 9. Adaptive reconstruction iteration samples

Iteration	Bits Per Pixel (BPP)	Compression Ratio
0	0.125	192:1
5	0.750	32:1
15	2.000	12:1

The evaluation of the adaptive reconstruction learning on satellite and aerial images is shown in figure 38.

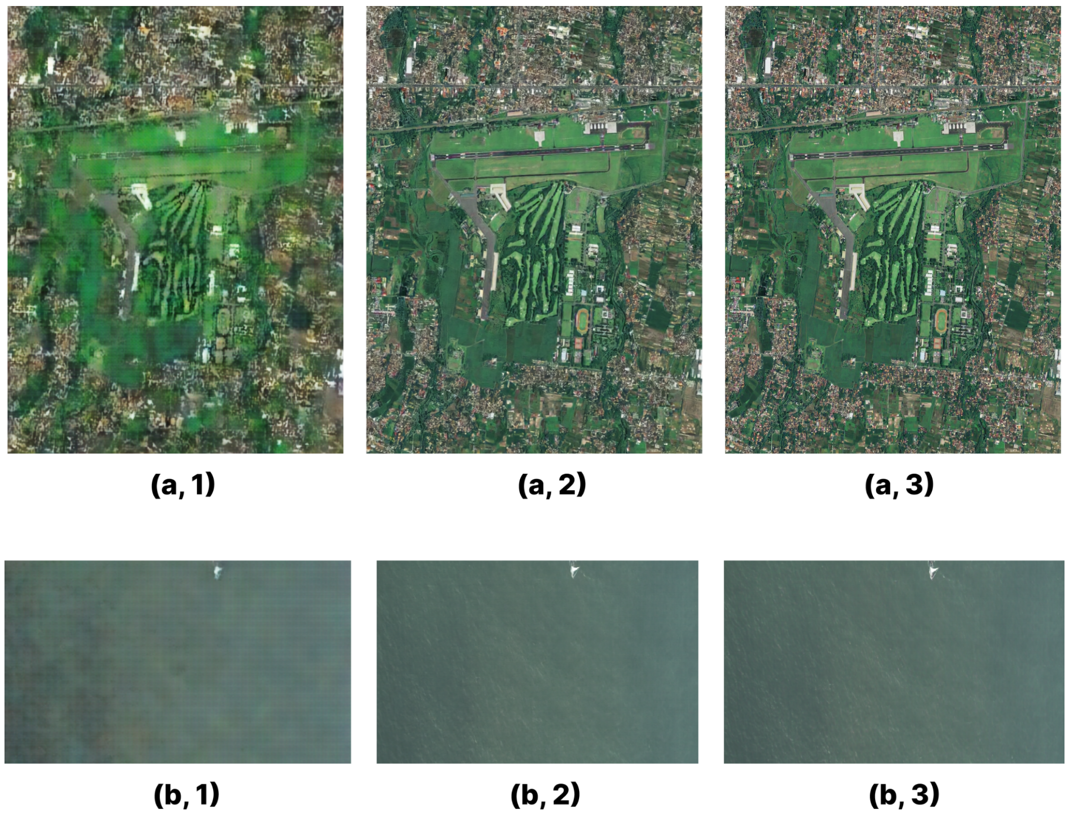


Figure 38. Visual illustration of the iterative reconstruction. a) Jogjakarta, Indonesia [Source: Airbus, Pleiades], b) Helgoland island, North Sea.

5.3.3 GAN: High-Fidelity Generative Image Compression Autoencoder

The High-Fidelity Generative Image Compression (HiFiC) combines GAN and learned image compression. The HIFIC model arguments conditional GAN with learned lossy compression. In figure 39 HiFiC is the model trained using the MSE loss function and the Learned Perceptual Image Patch Similarity (LPIPS) combined with GAN [117, 146]. The M&S is the deep-learning-based Mean and Scale Hyperprior optimized for mean squared error [149]. The Better Portable Graphics (BPG) is a non-learned codec based on H.265 the High-Efficiency Video Coding that commonly reaches significant PSNR [147, 148]. No GAN is a benchmark, using the same architecture as the HiFiC model, but with combining with the GAN. The following graph shows the average bits per pixel (bpp) and the learned components, which done on the images of the evaluation study.

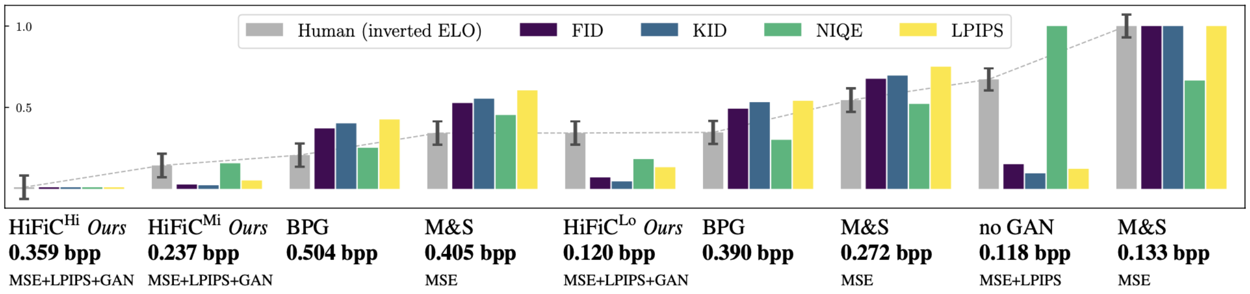


Figure 39. Comparison between the HIFIC and some state-of-the-are compression models. [Source: HIFIC [117]]

The compression performance comparison above shows that training the model with GAN improves the image decoding reconstruction. The trained GAN model significantly outperforms other compression methods with an intensive bit per rate compression. Further, the HIFIC model utilized Leaky ReLU (LReLU) and ReLU activation functions. The advantage of using the LReLU is to tune the saturation losses of the signal to either zero or the dominance of random noise from digital rounding, which reflects a significant optimization of the training process [150]. Figure 40 shows the GAN-based autoencoder architecture.

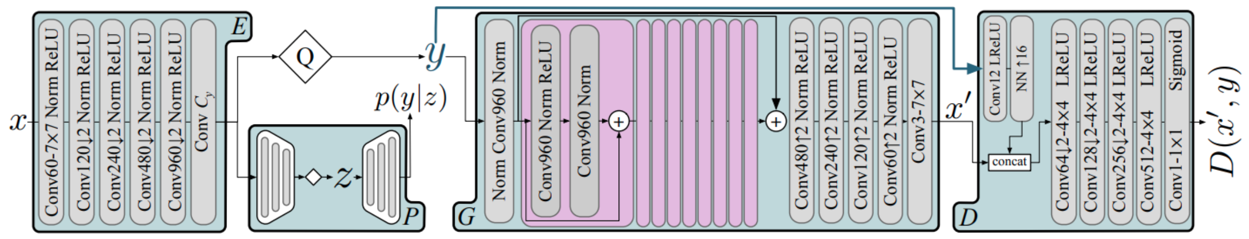


Figure 40. HIFIC autoencoder architecture [Source: HIFIC [117]].

Considering the transfer learning paradigm, the HIFIC model was retrained to combine 2D satellite images with 3-color-channels. Due to the high computational cost of the HIFIC model, the model has been optimized and retrained twice up to 50,000 epochs. The first time model was trained using MSE and LPIPS only as loss functions up to 50,000 epochs, then saved a training checkpoint. The second time the model used the trained model weights using the MSE and LPIPS checkpoint 50,000 but applied the GAN loss on it for another 50,000 epochs. The training outcome is saved as a checkpoint, which can be directly evaluated or extended until the 1 million epochs per the original model. The model checkpoint was evaluated on, 7006 random images mixed between images from the COCO dataset and satellite images. There are noticed 20% increase in losses and drops in the PSNR than the model trained for 1 million epochs. Further, the pre-trained version and the two trained checkpoints of the model are included in the source code base as listed in appendix A.

The reconstructed results using the HIFIC trained on 1 million COCO images are significantly impressive; thus, a difference image is calculated to check the losses between the original and reconstructed images. Figure 41 shows that the losses are mainly the noise eliminated from the encoded image, which highlights the advantage of the autoencoder's ability to denoise the data.

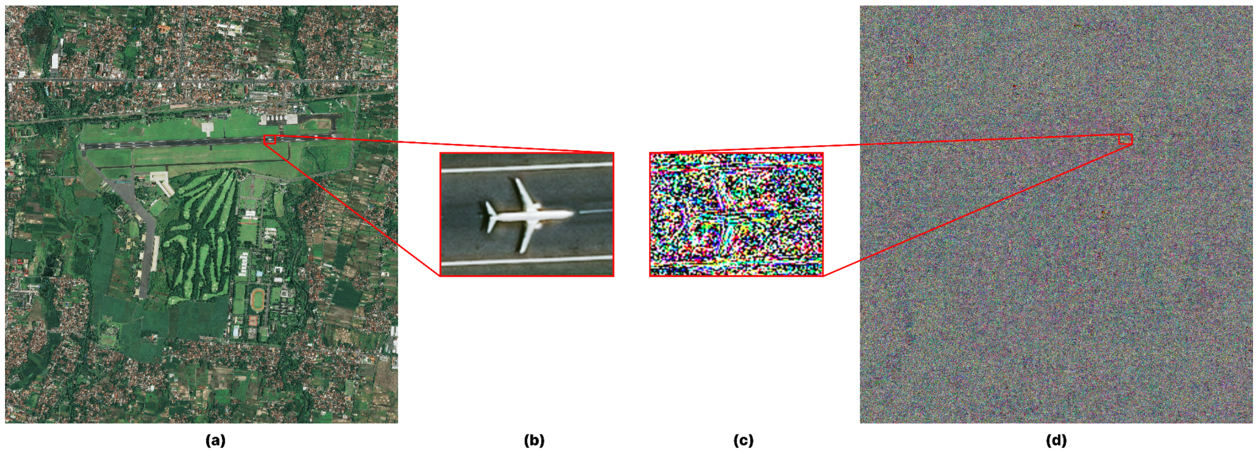


Figure 41. Difference image calculated between original and reconstructed image. a) and b) original, c and d) reconstructed.

Table 10 shows that the HIFIC-GAN model outperforms the LSTM-RNN-based compression autoencoder. Since LSTM-RNN already outperforms the JPEG compression method, then no further comparison with HIFIC-GAN is required.

Table 10. Comparison on the quality of the reconstructed images suing LSTM and GAN models

	LSTM-RNN	HIFIC-GAN
MSE	116.1020	54.3756
SSIM	0.9083	0.9049
MS-SSIM	0.9625	0.9544
Compression ratio	12:1	26:1
Bits Per Pixel	2.0000	0.3590

The HIFIC results show dramatic improvement in the MSE than the LSTM-RNN model. A significant difference in the compression ratio and BPP between the two models, where the HIFIC-GAN model wins over. The LSTM-RNN model showed a slightly better SSIM and MS-SSIM index than the HIFIC-GAN model; however, the slight difference can be ignored considering the advanced compression ratio. Finally, with the near-to-1 MS-SSIM index results, the autoencoder compression using the selected architectures can be described as near-lossless compression, contrary to JPEG compression.

5.4 Experiment DEHIA: DESIS and Hyperspectral images Autoencoder

The DESIS and HSI images Autoencoder (DEHIA) experiment show the results of developing an autoencoder to compress and reconstruct Hyperspectral images. Reducing the spectrum channels dimensionality of HSIs is crucial during the decision-making process on the raw data and the analysis. By utilizing the advantages of autoencoder, quick analysis can be reached and eliminate the dead-weight data on the host satellites. The DESIS instrument has 235 spectral bands with a wavelength range from VIS (400 nm) to near-infrared (1000 nm), reflecting a 2.5 nm spectral sampling and a ground sampling of 30 m from ~430 km orbit of the ISS [120]. Figure 42 shows a visualization of the spectral channels of the DESIS instrument (a), referenced with an HSI from the PaviaU dataset [126].

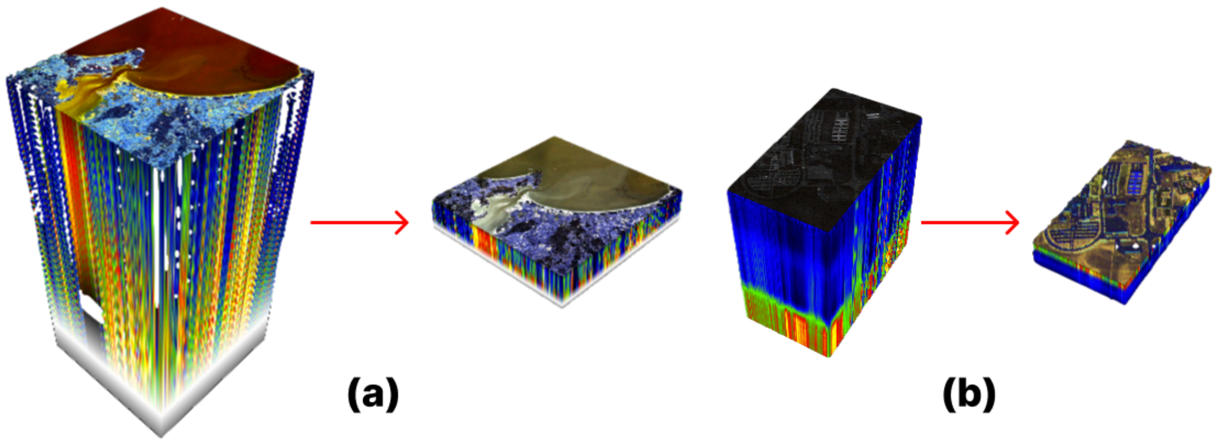


Figure 42. Compression visualization on a) 235 bands HSI [Source: DLR, DESIS], b) 103 bands HSI [Source: Pavia University].

During the preprocessing phase of the DESIS HSI, spectral stacking was required to prepare the training and evaluation data for the autoencoder. The visualization in figure 43 shows different wavelength channels of the HSI, a) and b) VIS mixed with NIR, b) and c) are VIS, and e) is the gray ground truth of the HSI.

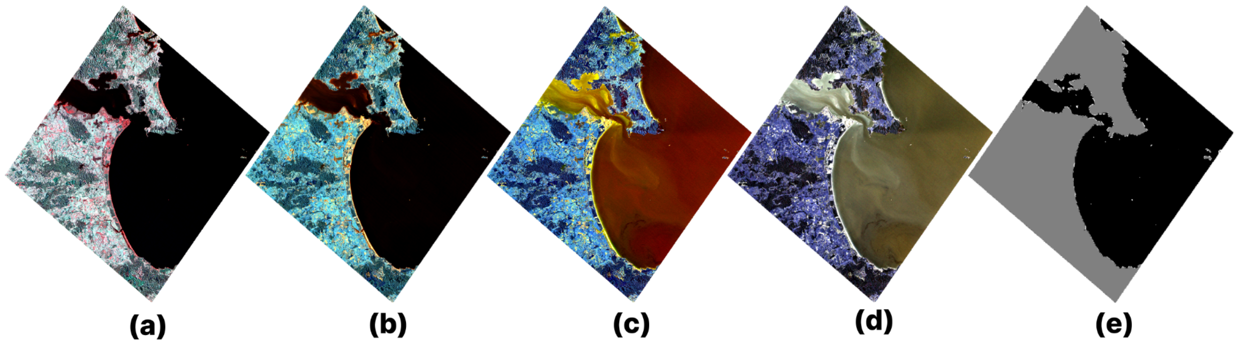


Figure 43. Different spectral visualization on DESIS HSI, from Paparoa, New Zealand.

Two hyperspectral autoencoder architectures were implemented in this experiment. The first is a linear MLP autoencoder, and the second is a CNN autoencoder. Since the PaviaU dataset provides latent classification labels, the autoencoders extended to classify the extracted features from the latent space [127]. The cosine spectral angle (CSA) loss function is primarily used to train and optimize the spectral autoencoders [128]. The cosine spectral loss function is described in equation (5.3). Where K is the original hyperspectral dimensions, L is the output layer index, yk is an element of the input data y, f is the sigmoid activation function, $f(z_k^{(L)})$ is an element of the reconstructed input $f(z^{(L)})$.

$$E_{SA} = 1 - \frac{\sum_{k \in K} f\left(z_k^{(L)}\right) y_k}{\left|f\left(\mathbf{z}^{(L)}\right)\right| |\mathbf{y}|} \tag{5.4}$$

The result of feature extraction classification on the latent space representation of the PaviaU dataset using the linear MLP autoencoder is shown in figure 44.

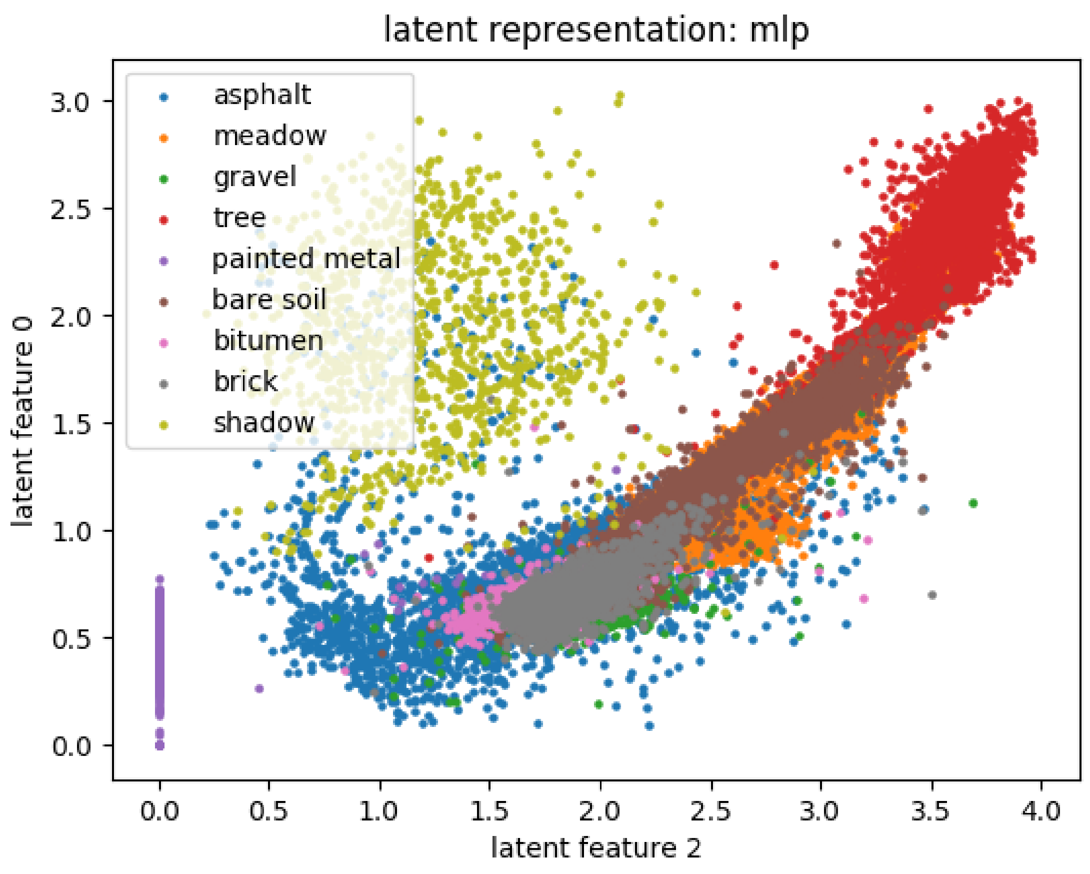


Figure 44. Latent space classes with MLP autoencoder. [Source: DeepHype [127]]

The feature extraction classification results in the same dataset's latent space using the CNN autoencoder shown in figure 45. It is clear from figure 45 that convolutional autoencoders are more advanced than the dense MLP in feature extraction.

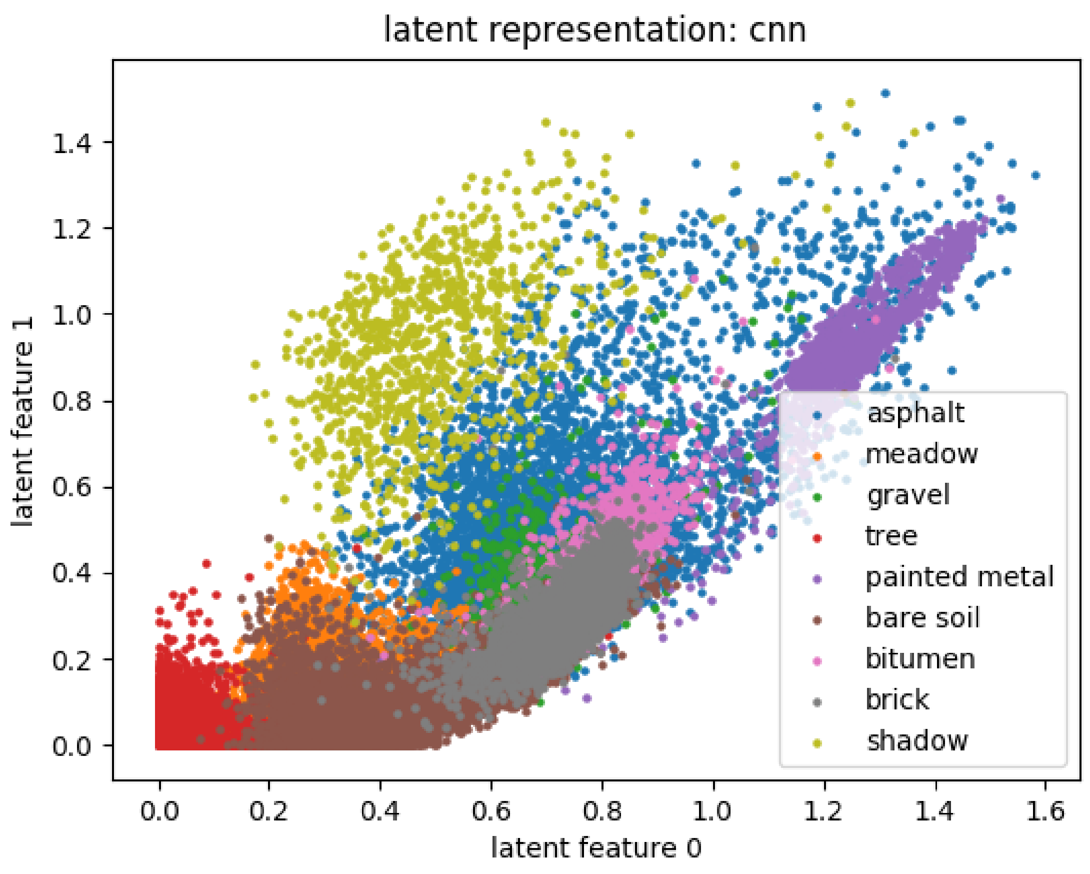


Figure 45. latent space with CNN autoencoder. [Source: DeepHype [127]]

A latent representation of the total spectral of DESIS HSI visualized on an iterative 3D scale using OpenGL and the code used to produce the latent plots above is listed on the code source as per Appendix A. Finally, DEHIA showed successful compression ratio of 21:1, which evaluated on DESIS and PaviaU dataset. The results of HSI compression using autoencoders surpass JPEG 2000 by ~1.5 dB at rates under 0.15 bps and outperform the CCSDS)-122.1-B-1 by up to 5 dB overall rates [135, 136].

5.5 Experiment PEMDE: Procedures for Embedded Deployment

This experiment aims to apply the required procedures to deploy a neural network to an embedded system. Quantization and compilation are required to convert the trained model to be compatible with the limited resources of the embedded host. Xilinx FPGA Ultrascale system on chip (SoC) Zynq-7020 and Ultrascale+ multiprocessor system on a chip (MPSoC) Zynq ZU7EV were used to evaluate the PEMDE experiment. Both FPGAs are evaluated using the PYNQ-Z2 for the Zynq-7020 SoC and ZCU104 containing the ZU7EV MPSoC.

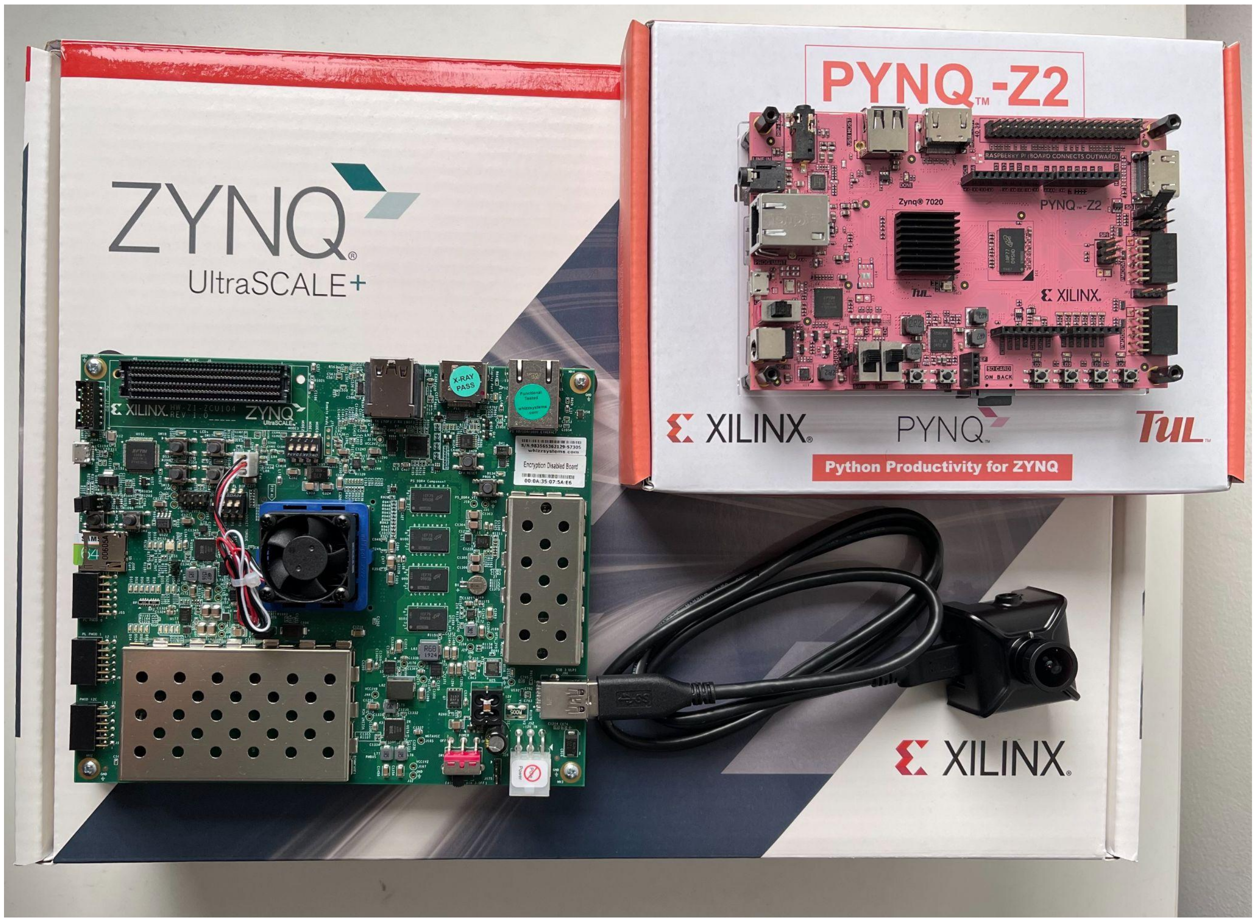


Figure 46. On the left side the ZCU104 evaluation board, and on the right top, the PYNQ-Z2 evaluation board. [Source: Author]

5.5.1 Quantization and pruning

Quantization processes also include pruning, which compresses the model weights to fit the fixed point architecture of the embedded device. There are two types of quantization, post-training quantization (PTQ) and quantization-aware training (QAT). After a thoroughly trained and validated model, the post-training quantization considers quantization and pruning. Quantization-Aware training considers building the neural network layers with quantized Fixed Point Layers from the beginning instead of the Float Point Layers. PTQ is faster than the QAT, which is commonly used. 5.5.2 experiment applies QAT, 5.5.3 experiment applies PTQ.

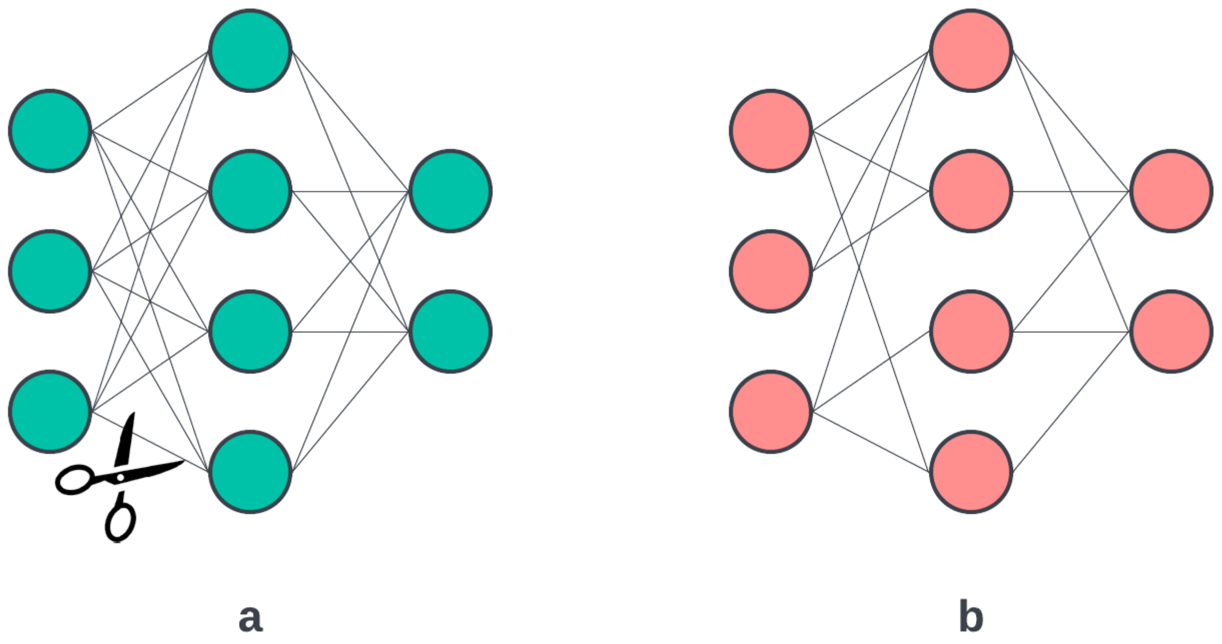


Figure 47. Illustration about the pruning process, (a) before pruning, (b) after pruning.

5.5.2 Deployment of simple neural network to the FPGA

HLS4ML is the primary framework for this part of the experiment. Combined with Vivado software and the sophistication of Python, developing neural networks on FPGA is significantly faster than using Hardware Description Language (HDL). Further, HLS4ML synthesis of the TensorFlow Python implementation into the required HDL to build the Vivado project generates the Bit Stream overlay to accelerate on the FPGA [160].

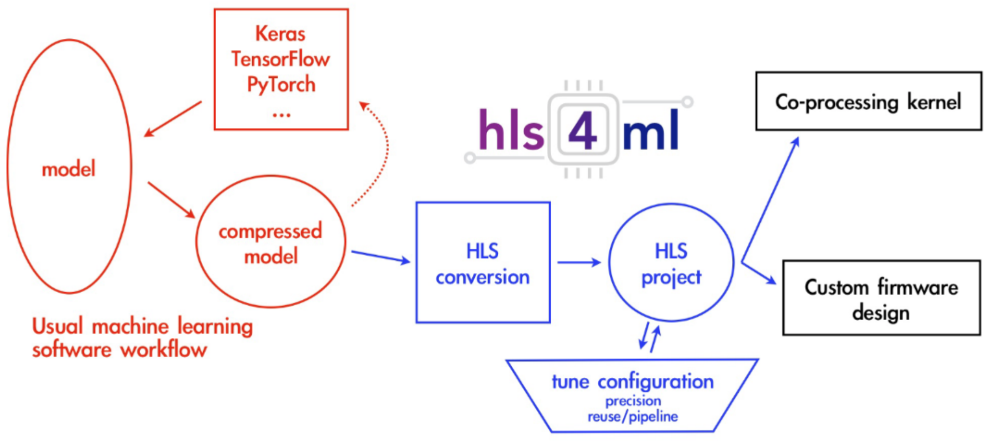


Figure 48. Typical workflow with HLS4ML [Source: Fast Machine Learning Lab [145]]

Since the autoencoders can be used for classification, a classification neural network is considered for this part of the experiment. Based on the MNIST handwritten digits, dataset is used to train an autoencoder for this experiment. This experiment aimed to evaluate the deployment procedures to an embedded system, in this case, Zynq-7020, on a PYNQ-Z2 board. Figure 49 shows the PYNQ-Z2 board.

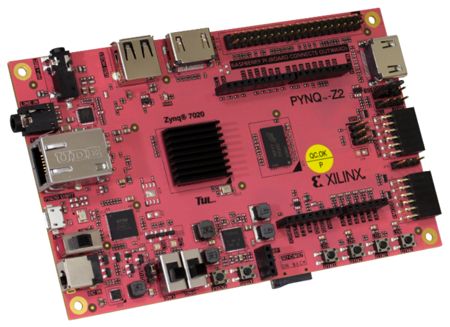
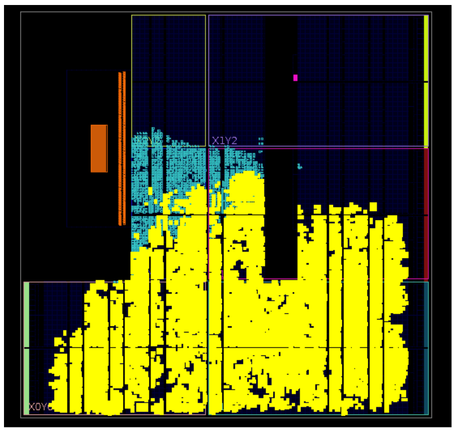
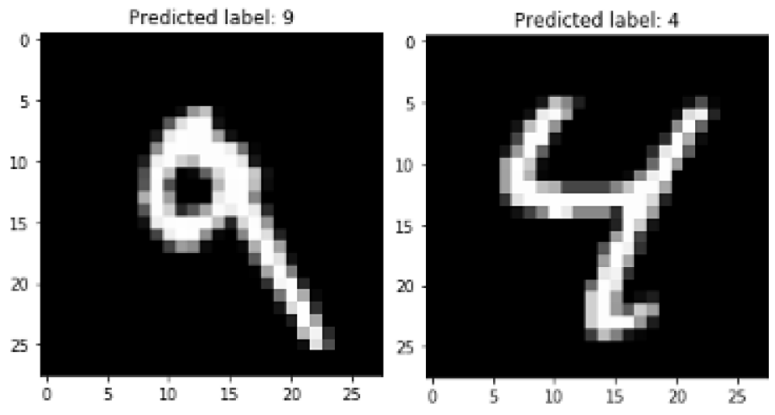


Figure 49. PYNQ-Z2 Zynq Ultrascale SoC [Source: Technology Unlimited]



(a)



(b)

Figure 50. (a) Hardware implementation of the neural network, (b) results accelerated on the SoC [Source: Author, Xilinx]

Figure 50. (a) shows the hardware implementation of the neural network on Zynq-7020 SoC (System on Chip), where the yellow color represents the neural network. The other color highlights the Advanced Microcontroller Bus Architecture (AMBA) interconnections (AXI) intellectual property (IP) core from Advanced Reduced instruction set computer Machine (ARM) on the Programmable logic. The AXIs connect the programmable logic to the soft processor, the ARM core on this board. Although a reuse factor of the input/output stream was defined during the compilation, the Zynq-7020 could not hold a full autoencoder but separately an encoder or decoder.

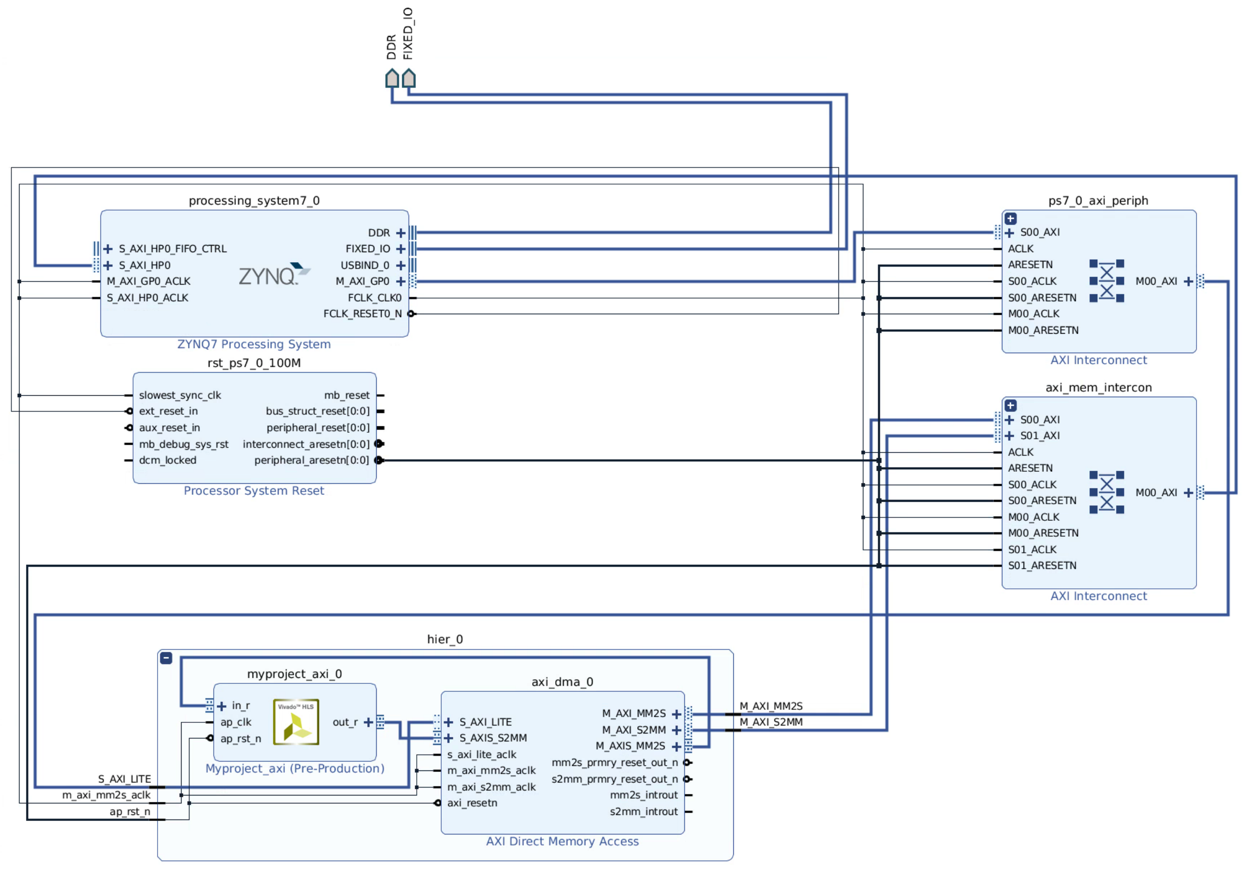
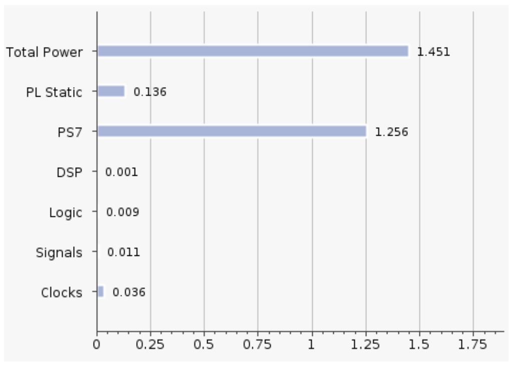


Figure 51. Autoencoder Hardware Architecture with Zynq-7020 [Source: Vivado, Xilinx]

After producing the neural network IP core and the dynamic memory allocator core from the Vivado accelerator program, The system implementation block diagram is built manually, as described in figure 51. Further, figure 52 and 53 shows the plots of the resources allocated from the SoC and the power budget on-chip.



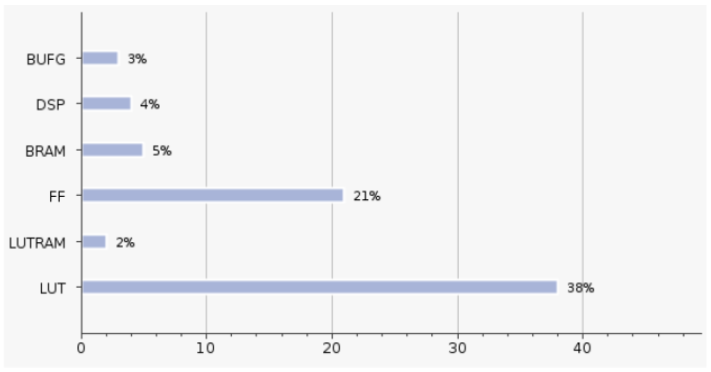
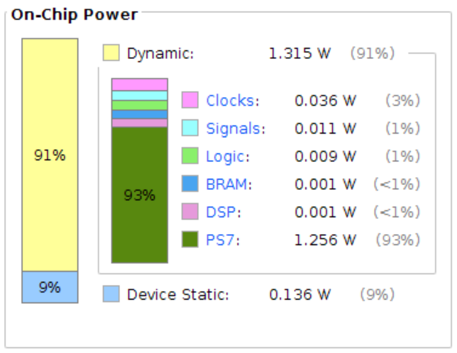


Figure 52. a) Power budget, b) resources allocation. [Source: Vivado, Xilinx]



On-Chip Power

Dynamic:	1.315 W	(91%)	
🟪 Clocks:	0.036 W	(3%)	
🟦 Signals:	0.011 W	(1%)	
🟩 Logic:	0.009 W	(1%)	
🟦 BRAM:	0.001 W	(<1%)	
🟪 DSP:	0.001 W	(<1%)	
🟩 PS7:	1.256 W	(93%)	
Device Static:	0.136 W	(9%)	

Figure 53. On-Chip power budget [Source: Author, Vivado, Xilinx]

Although the total power on-chip results in 1.451W, considering the losses and heat dissipation of the chip temperature on 41°C, the overall power of the chip results in ~3.5W. The optimal throughput is 103 images/s on a 100 MHz clock. The 103 images/s throughputs of the neural network can be extrapolated from 28 x 28 x 1 to 256 x 256 x 1 as per experiment 5.2.2. Thus, the throughput will be 1.232 images/s on a single node. Considering the ScOSA requirements of using maximum 2 nodes for image compression, using Zynq-7020 SoC nodes will 2 encoders can reach a maximum of 2.464 images/s on based on the encoder from 5.2.2.

5.5.3 Deployment of the autoencoder to the FPGA

The Xilinx Deep Processing Unit (DPU) can accelerate the execution of different types of neural network layers; nevertheless, the need to execute models that have fully custom layers is required for complex models. Such a layer is the sampling function of a variational autoencoder (VAE). The DPU fits to accelerate the convolutional encoder and decoder, but not the statistical sampling layer. The statistical sampling layer executed on the soft processor in the ZU7EV is the ARM Core CPU. This experiment uses a CNN-based denoising VAE. The VAE maps the input to a latent space which is a normal distribution. The mean and standard deviation of the learned distribution are passed to the decoder. Vitis AI platform was used for the development and inference of this experiment [161]. The MNIST 28 × 28 figures dataset is used to evaluate the deployment on the ZCU104 board. Figure 54 shows the ZCU104 board.

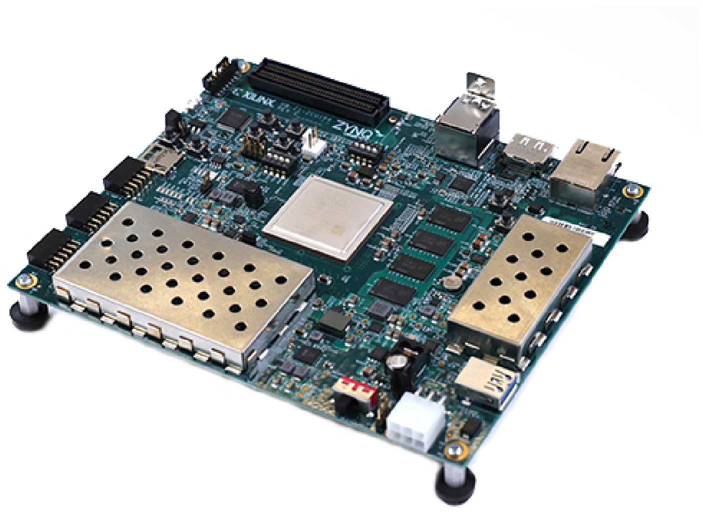


Figure 54. ZCU104 Zynq Ultrascale+ MPSoC [Source: Xilinx]

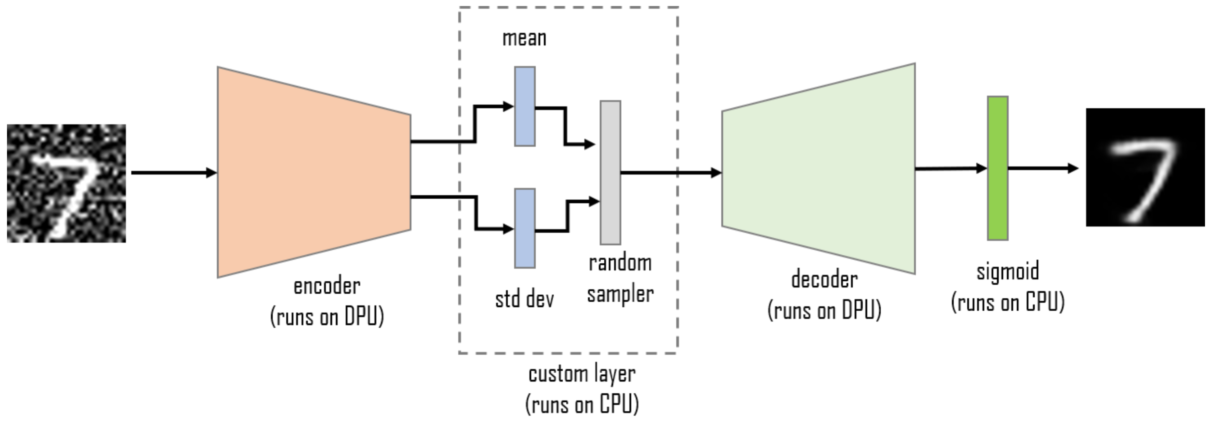


Figure 55. VAE architecture using the DPU and the CPU [Source: Vitis Ai, Xilinx [161]]

Figure 55. shows the VAE abstraction and configuration on the FPGA, where the encoder, decoder, and sigmoid functions are running on the DPU. Further, the standard deviation, mean, and random sampler run on the soft processor (ARM CPU) as a custom layer. The complete autoencoder block diagram of the VAE inference on the ZU7EV MPSoC is listed in appendix C.

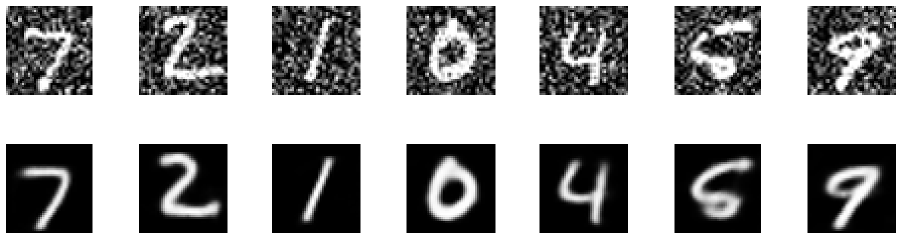


Figure 56. Input and decoded output visualization. [Source: Author, Xilinx, Vitis Ai [161]]

Figure 56. shows the encoded and decoded samples of the VAE. During the pre-processing phase, random gaussian noise was added to all the training and testing MNIST datasets. Figure 57. shows the DPU block diagram, where APU is the Application Processing Unit, PE is the Processing Engine, DPU is the Deep Learning Processing Unit, and RAM is the Random Access Memory.

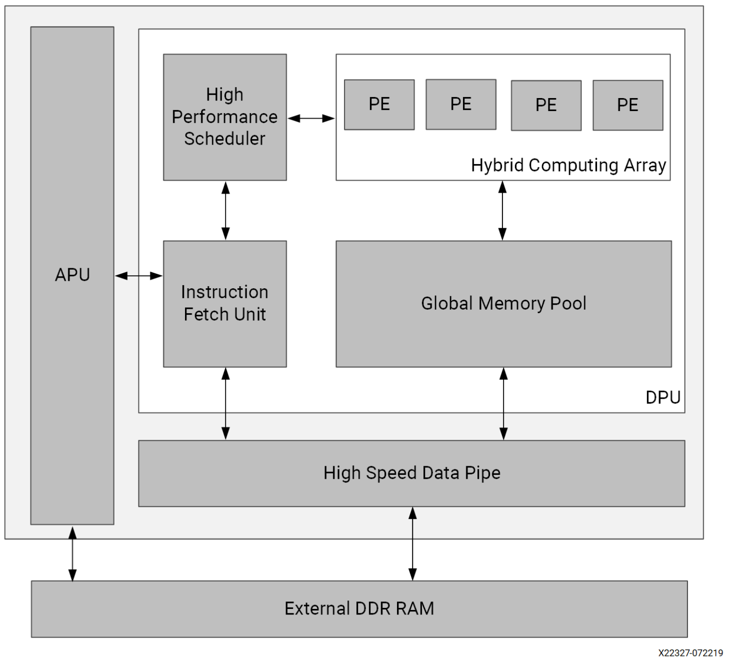


Figure 57. DPU Top-Level Block Diagram [Source: Zynq DPU Product Guide [132]]

Figure 58. shows the Vitis AI stack, where the Vitis AI development platform consists of AI Compiler, AI Quantizer, AI Optimizer, AI Profiler, AI Library, and Xilinx Runtime Library (XRT). The Vitis AI environment is used for AI application inference on Xilinx hardware. Further, Vitis AI consists of optimized IP cores and tools that optimize the targeted hardware's quantization and compilation processes. Furthermore, a detailed block diagram of the zynq architecture is listed in appendix C.

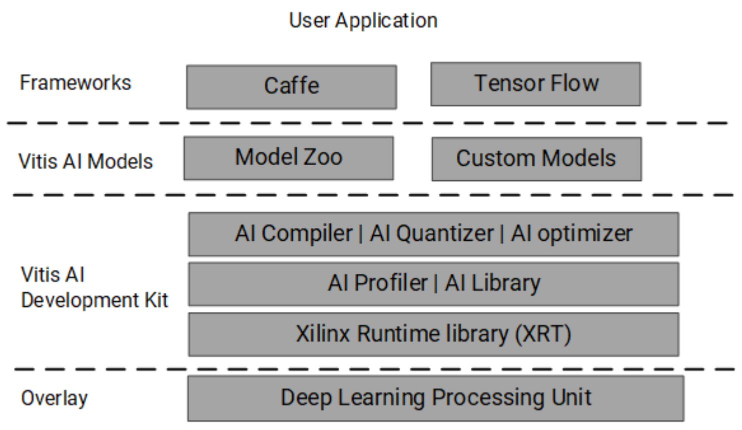


Figure 58. Vitis AI Stack [Source: Zynq DPU Product Guide [132]]

The overall profile power budget is 8.5W, and the clock frequency of 300MHz. The stats outcome measured 2000 frames/s for preprocessing throughput and 1346 images/s for the encoding throughput. The ZCU104 is configured to use both DPUs available; thus, two encoders are running heterogeneously. The results can be extrapolated to 256 x 256 x 1 images based on the BIROS experiment 5.2.2 to yield a throughput of 16.102 images/s. Further, the throughput can reach 1.0063 images/s when a high-resolution camera input of 1K resolution is fed as input to the encoder. The throughput of ~1 image/second is considered in the real-time processing range.

The experiments in chapter 5 are concluded with the evaluation analysis in table 11, which provides the cross-comparison of some of the most prominent frameworks utilized in aerial and satellite imagery compression.

Table 11. Performance comparison of different image compression techniques

Method	Performance	
	Features	*Advantages*
RNN-based Approach [6]	Innovative approach via analysis and synthesis block, based on Generalized Divisive Normalization (GDN)	Outperforms traditional algorithms, such as BPG and JPEG, on most parameters according to the Kodak benchmark.
Supervised Image Compression for Split Computing [7]	Utilizes split computing and knowledge distillation to create a lightweight feature extractor for consequent reconstruction	Can be effectively used for resource-constrained edge computing systems; demonstrates positive results on input and feature compression
Slimmable Compressive Autoencoders [8]	Utilizes only non-parametric operations in compressive autoencoders	Effective for resource-constrained edge computing systems; adjustable to various purposes and settings; low memory footprint, costs, and latency
Generative Adversarial Networks [9]	Utilize contemporary quantization-aware training for GAN architectures	Reveal the efficiency of GAN quantization, which can be used in consequent RNNs and VAEs
Rate-Distortion Optimization-based Network(RDONet) [10]	Implements dynamic block portioning and additional hierarchical levels to maximally optimize rate-distortion component	Outperforms traditional algorithms while saving up to 20% of RD

6 Conclusion and future work

6.1 Summary

The autoencoder neural network and the differences between autoencoder types are studied in this thesis. After learning about the features and advancements of autoencoders, five experiments are conducted. The five experiments are SIMPA, BIRIA, MPSIA, DEHIA, and PEMDE. SIMPA is the very first autoencoder implemented to visualize and evaluate the neural network's performance. SIMPA also showed a promising approach to data encoding and decoding, which can be used in cybersecurity matters like signal encryption and anti-spoofing mitigation [109]. BIRIA experiment focused mainly on the DLR data, where BIROS satellite image compression was discussed. BIRIA showed significant results in compressing satellite sensor data that led to a high compression factor of 64:1. BIRIA also included the development of a new satellite image compression pipeline based on neural networks.

MPSIA experiment discussed the evaluation of the transfer learning paradigm; thus, two state-of-the-art autoencoders used in web and game dev industries are optimized and evaluated on satellite images. Further, MPSIA showed promising results based on the transfer learning approach. DEHIA experiment mainly discussed hyperspectral satellite image compression based on the reduction of spectrum channels. Further, DEHIA discussed that the convolutional autoencoders outperform the classical CCSDS compression algorithms. DEHIA concluded with a success of 21:1 compression factor.

The PEMDE experiment discussed the procedures for neural network deployment to embedded devices. PEMDE targeted two AI-Ready hardware platforms, Zynq-7020 and ZU7EV. Further, the throughput of the encoder evaluated on the Zynq-7020 can be extrapolated to reach 2.464 images/s with 3.5W power on-chip by using two parallel zynq-7020 nodes. The ZU7EV can hire 2 encoders in parallel on the same chip by using the dual DPU available on the MPSoC. PEMDE concluded that the ZU7EV MPSoC could reach real-time processing when the encoder extrapolated to infer satellite sensor radiance with ~1K resolution.

Autoencoders showed significant results compared to different compression algorithms, which shall be recognized as lossy and near-lossless compression as per the BIRIA and MPSIA experiments.

6.2 Future work

Quantization dramatically impacts neural network accuracy when it comes to embedded deployment. The 14-layers-autoencoder developed in the BIRIA experiment shall need an enlargement of its dataset and experimenting with the QAT and PTQ approaches. Future advancements in autoencoder usage adapted with the Consultative Committee for Space Data Systems (CCSDS) compression algorithms are recommended to be studied as per [121, 133, 134]. Further, satellite hyperspectral image compression and in-depth analysis using convolutional autoencoders gain future research interest as per [135, 137]. As an extension to the DEHIA experiment, an autoencoder shall be developed to compress the hyperspectral images based on the image resolution and the spectrum channel, leading to a significant increase in the compression ratio.

Recommendation for integration with ScOSA project focus on the hardware implementation. Considering the distributed on-board computing of ScOSA, which hires multi-nodes of Zynq-7020, an autoencoder neural network would significantly improve the data storage rates and maintain higher quality. Further, future advancement of ScOSA, including nodes of the Zynq ZU7EV or ZU9EV MPSoCs, would increase the throughput to real-time compression using the autoencoder approach. Futhermore, studing the new canonical hardware architecture for compression, such as Neuromorphic Spiking autoencoders and Quantum autoencoders, is highly recommended as per [137, 138, 139]. In conclusion, the autoencoders approach showed promising results to adapt with modern space applications, such as constrained CubeSats and deep space missions.

References

[1] Abboud, Ali J., Ali N. Albu-Rghaif, and Abbood Kirebut Jassim. "Balancing compression and encryption of satellite imagery." International Journal of Electrical and Computer Engineering 8, no. 5 (2018): 3568.1

[2] Afjal, Masud Ibn, Md Al Mamun, and Md Palash Uddin. "Band reordering heuristics for lossless satellite image compression with 3D-CALIC and CCSDS." Journal of Visual Communication and Image Representation 59 (2019): 514-526.

[3] Ahn, Kyohoon, Sung-Hun Lee, In-Kyu Park, and Hwan-Seok Yang. "Simulation of a laser tomography adaptive optics with Rayleigh laser guide stars for the satellite imaging system." Current Optics and Photonics 5, no. 2 (2021): 101-113.

[4] Akshay, S., T. K. Mytravarun, N. Manohar, and M. A. Pranav. "Satellite image classification for detecting unused landscape using CNN." In 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 215-222. IEEE, 2020.

[5] Asokan, Anju, J. Anitha, Monica Ciobanu, Andrei Gabor, Antoanela Naaji, and D. Jude Hemanth. "Image processing techniques for analysis of satellite images for historical maps classification—An overview." Applied Sciences 10, no. 12 (2020): 4207.

[6] Bai, Yuanchao, Xianming Liu, Wangmeng Zuo, Yaowei Wang, and Xiangyang Ji. "Learning scalable ly=-constrained near-lossless image compression via joint lossy image and residual compression." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11946-11955. 2021.

[7] Bass, L. P., Yu A. Plastinin, and I. Yu Skryabysheva. "Machine learning in problems involved in processing satellite images." Measurement Techniques 63, no. 12 (2021): 950-958.

[8] Bauer, Susanne E., Kostas Tsigaridis, Greg Faluvegi, Maxwell Kelley, Ken K. Lo, Ron L. Miller, Larissa Nazarenko, Gavin A. Schmidt, and Jingbo Wu. "Historical (1850–2014) aerosol evolution and role on climate forcing using the GISS ModelE2. 1 contribution to CMIP6." Journal of Advances in Modeling Earth Systems 12, no. 8 (2020): e2019MS001978.

[9] Behrens, Jonathan R., and Bhavya Lal. "Exploring trends in the global small satellite ecosystem." New Space 7, no. 3 (2019): 126-136.

[10] Beyer, Ross A., Oleg Alexandrov, and Scott McMichael. "The Ames Stereo Pipeline: NASA's open-source software for deriving and processing terrain data." Earth and Space Science 5, no. 9 (2018): 537-548.

[11] Blackwell, William J., S. Braun, R. Bennartz, C. Velden, M. DeMaria, R. Atlas, J. Dunion et al. "An overview of the TROPICS NASA earth venture mission." Quarterly Journal of the Royal Meteorological Society 144 (2018): 16-26.

[12] Borra, Surekha, Rohit Thanki, and Nilanjan Dey. Satellite image analysis: clustering and classification. Singapore: Springer, 2019.

[13] Bosch, Marc, Kevin Foster, Gordon Christie, Sean Wang, Gregory D. Hager, and Myron Brown. "Semantic stereo for incidental satellite images." In 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1524-1532. IEEE, 2019.

[14] Bowler, Tim R. "A new space race." In Deep Space Commodities, pp. 13-19. Palgrave Macmillan, Cham, 2018.

[15] Bychkov, I. V., G. M. Ruzhnikov, R. K. Fedorov, A. K. Popova, and Y. V. Avramenko. "Classification of Sentinel-2 satellite images of the Baikal Natural Territory." Computer Optics 46, no. 1 (2022): 90-96.

[16] Canton, Helen. "European Space Agency—ESA." In The Europa Directory of International Organizations 2021, pp. 549-551. Routledge, 2021.

[17] Cawse-Nicholson, Kerry, Philip A. Townsend, David Schimel, Ali M. Assiri, Pamela L. Blake, Maria Fabrizia Buongiorno, Petya Campbell et al. "NASA's surface biology and geology designated observable: A perspective on surface imaging algorithms." Remote Sensing of Environment 257 (2021): 112349.

[18] CENTRES, ESA. "European Space Agency—ESA." Organization (2020).

[19] Chen, Zhenzhong, Ye Hu, and Yingxue Zhang. "Effects of compression on remote sensing image classification based on fractal analysis." IEEE Transactions on Geoscience and Remote sensing 57, no. 7 (2019): 4577-4590.

[20] Cheng, Tianze. "Review of novel energetic polymers and binders–high energy propellant ingredients for the new space race." Designed Monomers and Polymers 22, no. 1 (2019): 54-65.

[21] Duvaux-Béchon, Isabelle. "The European Space Agency (ESA) and the United Nations 2030 SDG Goals." In Embedding Space in African Society, pp. 223-235. Springer, Cham, 2019.

[22] Ekaterina Kalinicheva et al. “Neutral network autoencoder for change detection in satellite image time series,” IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2018, doi:10.1109/icecs.2018.8617850

[23] Erickson, Andrew S. "Revisiting the US-Soviet space race: Comparing two systems in their competition to land a man on the moon." Acta Astronautica 148 (2018): 376-384.

[24] Fabian Brand et al. “Rate-distortion optimized learning-based image compression using an adaptive hierarchical autoencoder with conditional hyperprior”, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.

[25] Fai Yang et al. “Slimmable compressive autoencoders for practical neural image compression”, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.

[26] Gould, Michael. "Facilitating Small Satellite Enterprise for Emerging Space Actors: Legal Obstacles and Opportunities." In Legal Aspects Around Satellite Constellations, pp. 29-46. Springer, Cham, 2021.

[27] Green, Robert O., Natalie Mahowald, Charlene Ung, David R. Thompson, Lori Bator, Matthew Bennet, Michael Bernas et al. "The Earth surface mineral dust source investigation: An Earth science imaging spectroscopy mission." In 2020 IEEE Aerospace Conference, pp. 1-15. IEEE, 2020.

[28] Gunasheela, K. S., and H. S. Prasantha. "Satellite image compression-detailed survey of the algorithms." In Proceedings of International Conference on Cognition and Recognition, pp. 187-198. Springer, Singapore, 2018.

[29] Hagag, Ahmed, Ibrahim Omara, Souleyman Chaib, Guangzhi Ma, and Fathi E. Abd El-Samie. "Dual link distributed source coding scheme for the transmission of satellite hyperspectral imagery." Journal of Visual Communication and Image Representation 78 (2021): 103117.

[30] Haroun, Fathi Mahdi Elsiddig, Siti Noratiqah Mohamad Deros, and Norashidah Md Din. "Detection and Monitoring of Power Line Corridor From Satellite Imagery Using RetinaNet and K-Mean Clustering." IEEE Access 9 (2021): 116720-116730.

[31] Caldwell, Sonja. 2021. “11.0 Ground Data Systems and Mission Operations,” October. https://www.nasa.gov/smallsat-institute/sst-soa/ground-data-systems-and-mission-operations.

[32] Hu, Yueyu, Wenhan Yang, Zhan Ma, and Jiaying Liu. "Learning end-to-end lossy image compression: A benchmark." IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).

[33] Huang, Wei, Yueyun Wang, and Rui Wang. "A high fidelity haze removal algorithm for optical satellite images using progressive transmission estimation based on the dark channel prior." International Journal of Remote Sensing 40, no. 9 (2019): 3486-3503.

[34] Hudec, Rene, Graziella Branduardi-Raymont, Steven Sembay, and Vojtech Simon. "European Space Agency SMILE and Czech participation." Astronomische Nachrichten 341, no. 3 (2020): 341-347.

[35] Ihalainen, Jani. "Computer creativity: Artificial intelligence and copyright." Journal of Intellectual Property Law & Practice (2018).

[36] Iliopoulos, Nikolaos, and Miguel Esteban. "Sustainable space exploration and its relevance to the privatization of space ventures." Acta Astronautica 167 (2020): 85-92.

[37] Indradjad, A., Ali Syahputra Nasution, Hidayat Gunawan, and Ayom Widipaminto. "A comparison of Satellite Image Compression methods in the Wavelet Domain." In IOP Conference Series: Earth and Environmental Science, vol. 280, no. 1, p. 012031. IOP Publishing, 2019.

[38] Islam Khawar et al. "Image compression with recurrent neural network and generalized divisive normalization". In arXiv preprint arXiv: 2109.01999 (2021).

[39] Jamil, Sonain, and Md Piran. "Learning-Driven Lossy Image Compression; A Comprehensive Survey." arXiv preprint arXiv:2201.09240 (2022).

[40] Johnston, Nick, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, and George Toderici. "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4385-4393. 2018.

[41] Kaarna, Arto, G. Ohinata, and A. Dutta. Compression of spectral images. INTECH Open Access Publisher, 2007.

[42] Kganyago, Mahlatse, and Paidamwoyo Mhangara. "The role of african emerging space agencies in earth observation capacity building for facilitating the implementation and monitoring of the African development agenda: the case of african earth observation program." ISPRS International Journal of Geo-Information 8, no. 7 (2019): 292.

[43] Khelifi, Fouad, Tahar Brahimi, Jungong Han, and Xuelong Li. "Secure and privacy-preserving data sharing in the cloud based on lossless image coding." Signal Processing 148 (2018): 91-101.

[44] Khryashchev, Vladimir, and Roman Larionov. "Wildfire segmentation on satellite images using deep learning." In 2020 Moscow Workshop on Electronic and Networking Technologies (MWENT), pp. 1-5. IEEE, 2020.

[45] Khryashchev, Vladimir, Leonid Ivanovsky, Vladimir Pavlov, Anna Ostrovskaya, and Anton Rubtsov. "Comparison of different convolutional neural network architectures for satellite image segmentation." In 2018 23rd conference of open innovations association (FRUCT), pp. 172-179. IEEE, 2018.

[46] Asokan, Anju, J. Anitha, Monica Ciobanu, Andrei Gabor, Antoanela Naaji, and D. Jude Hemanth. "Image processing techniques for analysis of satellite images for historical maps classification—An overview." Applied Sciences 10, no. 12 (2020): 4207.

[47] Zhang, Zhaoxiang, Akira Iwasaki, Guodong Xu, and Jianing Song. "Cloud detection on small satellites based on lightweight U-net and image compression." Journal of Applied Remote Sensing 13, no. 2 (2019): 026502.

[48] Li, Y., Bruno Sixou, and F. Peyrin. "A review of the deep learning methods for medical images super resolution problems." IRBM 42, no. 2 (2021): 120-133.

[49] Liakopoulos, Dimitris. "The future of European Space Agency-EU relationship: critical aspects and perspectives." European Journal of Current Legal Issues 25, no. 2 (2019).

[50] Lindner, Claudia, Andreas Rienow, and Carsten Jürgens. "Augmented Reality applications as digital experiments for education–An example in the Earth-Moon System." Acta Astronautica 161 (2019): 66-74.

[51] Mahdi Elsiddig Haroun, Fathi, Siti Noratiqah Mohamad Deros, and Norashidah Md Din. "A Review of Vegetation Encroachment Detection in Power Transmission Lines using Optical Sensing Satellite Imagery." arXiv e-prints (2020): arXiv-2010.

[52] Mahmoudi, S. Hadi, and Aishin Barabi. "Environmental Principles of Corporate Social Responsibility and Their Application to Satellite Mega-Constellations in Low Earth Orbit." In Legal Aspects Around Satellite Constellations, pp. 143-159. Springer, Cham, 2021.

[53] Matsuda, Ichiro, Tomokazu Ishikawa, Yusuke Kameda, and Susumu Itoh. "A lossless image coding method based on probability model optimization." In 2018 26th European Signal Processing Conference (EUSIPCO), pp. 151-155. IEEE, 2018.

[54] Moorthy, Talari Venkata Krishna, Anil Kumar Budati, Sandeep Kautish, S. B. Goyal, and Kolalapudi Lakshmi Prasad. "Reduction of satellite images size in 5G networks using machine learning algorithms." IET Communications (2022).

[55] Morton, Y. Jade, Frank van Diggelen, James J. Spilker Jr, Bradford W. Parkinson, Sherman Lo, and Grace Gao, eds. Position, navigation, and timing technologies in the 21st century: integrated satellite navigation, sensor systems, and civil applications. John Wiley & Sons, 2021.

[56] MP Garniwa, Pranda, Raden AA Ramadhan, and Hyun-Jin Lee. "Application of semi-empirical models based on satellite images for estimating solar irradiance in Korea." Applied Sciences 11, no. 8 (2021): 3445.

[57] Neufeld, Michael J. Spaceflight: a concise history. MIT Press, 2018.

[58] Nicart, Edgar Bryan B., Tony YT Chan, and Ruji P. Medina. "Enhanced Satellite Imaging Algorithm Classifier using Convolutional Neural modified Support Vector Machine (CNMSVM)." In Proceedings of the 2018 the 2nd International Conference on Video and Image Processing, pp. 222-225. 2018.

[59] Fortov, V. E., O. F. Petrov, A. D. Usachev, A. M. Lipaev, A. V. Zobnin, V. I. Molotkov, M. Yu Pustylnik et al. "Joint ESA-Roscosmos Experiment called “Plasma Cristall-4” aboard the International Space Station."

[60] Pan, Yao, Zhong Ming Chi, Qi Long Rao, Kai Peng Sun, and Yi Nan Liu. "Modeling and simulation of mission planning problem for remote sensing satellite imaging." In MATEC Web of Conferences, vol. 179, p. 03024. EDP Sciences, 2018.

[61] Patuzzo Fabrizio. “A comparison of classical and variational autoencoders for anomaly detection”. In: arXiv preprint arXiv: 2009.13793v1 (2020).

[62] Patwa, Neel, Nilesh Ahuja, Srinivasa Somayazulu, Omesh Tickoo, Srenivas Varadarajan, and Shashidhar Koolagudi. "Semantic-preserving image compression." In 2020 IEEE International Conference on Image Processing (ICIP), pp. 1281-1285. IEEE, 2020.

[63] Pavel Andreev et al. “Quantization of generative adversarial networks for efficient inference: A methodological study”. In: arXiv preprint arXiv: 2108.13996 (2021).

[64] Pekkanen, Saadia M. "Governing the new space race." American Journal of International Law 113 (2019): 92-97.

[65] Pelton, Joseph N. "Space-based solar power satellite systems." In Space 2.0, pp. 103-114. Springer, Cham, 2019.

[66] Peng, Daifeng, Yongjun Zhang, and Haiyan Guan. "End-to-end change detection for high resolution satellite images using improved UNet++." Remote Sensing 11, no. 11 (2019): 1382.

[67] Petrescu, Relly Victoria, Raffaella Aversa, Taher Abu-Lebdeh, Antonio Apicella, and Florian Ion Petrescu. "NASA satellites help us to quickly detect forest fires." American Journal of Engineering and Applied Sciences 11, no. 1 (2018): 288-296.

[68] Polar Geospatial Center. "Imagery processing options". https://www.pgc.umn.edu/guides/commercial-imagery/imagery-processing-options/?print=pdf Retrieved Oct. 21, 2021

[69] Quegan, Shaun, Thuy Le Toan, Jerome Chave, Jorgen Dall, Jean-Francois Exbrayat, Dinh Ho Tong Minh, Mark Lomas et al. "The European Space Agency BIOMASS mission: Measuring forest above-ground biomass from space." Remote Sensing of Environment 227 (2019): 44-60.

[70] Rast, Michael, and Thomas H. Painter. "Earth observation imaging spectroscopy for terrestrial systems: An overview of its history, techniques, and applications of its missions." Surveys in Geophysics 40, no. 3 (2019): 303-331.

[71] Robinson, Douglas KR, and Mariana Mazzucato. "The evolution of mission-oriented policies: Exploring changing market creating policies in the US and European space sector." Research Policy 48, no. 4 (2019): 936-948.

[72] S. Kim and M. Smolin. "Neural mmage compression". https://ml.berkeley.edu/blog/posts/neural-image-compression/#compressive-auto-encoders Retrieved Oct. 21, 2021

[73] Sagath, Daniel, Christopher Vasko, Elco Van Burg, and Christina Giannopapa. "Development of national space governance and policy trends in member states of the European Space Agency." Acta astronautica 165 (2019): 43-53.

[74] Sanjith, S. "Color Component Based Lossless Compression Method for Satellite Images (CCBLC)." Algerian Journal of Engineering and Technology 4 (2021): 54-58.

[75] Santos, Lucana, Ana Gomez, and Roberto Sarmiento. "Implementation of CCSDS standards for lossless multispectral and hyperspectral satellite image compression." IEEE Transactions on Aerospace and Electronic Systems 56, no. 2 (2019): 1120-1138.

[76] Schiopu, Ionut, and Adrian Munteanu. "A study of prediction methods based on machine learning techniques for lossless image coding." In 2020 IEEE International Conference on Image Processing (ICIP), pp. 3324-3328. IEEE, 2020.

[77] Schiopu, Ionut, and Adrian Munteanu. "Deep-learning-based lossless image coding." IEEE Transactions on Circuits and Systems for Video Technology 30, no. 7 (2019): 1829-1842.

[78] Shi, Qian, Mengxi Liu, Xiaoping Liu, Penghua Liu, Pengyuan Zhang, Jinxing Yang, and Xia Li. "Domain adaption for fine-grained urban village extraction from satellite images." IEEE Geoscience and Remote Sensing Letters 17, no. 8 (2019): 1430-1434.

[79] Siaw, Fei Lu, Yaw Yoong Sia, and Mallikarachchi Dilshani. "Development of Cloud Movement Prediction Method for Solar Photovoltaic System [J]." Journal of Harbin Institute of Technology (New Series) 29, no. 1 (2022): 64-69.

[80] Sommariva, Andrea. The Political Economy of the Space Age: How Science and Technology Shape the Evolution of Human Society. Vernon Press, 2018.

[81] Strese, H., and L. Maresi. "Technology developments and status of hyperspectral instruments at the European Space Agency." In Sensors, Systems, and Next-Generation Satellites XXIII, vol. 11151, p. 111510T. International Society for Optics and Photonics, 2019.

[82] Svynchuk, Olga, Oleg Barabash, Joanna Nikodem, Roman Kochan, and Oleksandr Laptiev. "Image compression using fractal functions." Fractal and Fractional 5, no. 2 (2021): 31.

[83] Tashiro, Makoto, Hironori Maejima, Kenichi Toda, Richard Kelley, Lillian Reichenthal, Leslie Hartz, Robert Petre et al. "Status of x-ray imaging and spectroscopy mission (XRISM)." In Space Telescopes and Instrumentation 2020: Ultraviolet to Gamma Ray, vol. 11444, p. 1144422. International Society for Optics and Photonics, 2020.

[84] Tellman, B., J. A. Sullivan, C. Kuhn, A. J. Kettner, C. S. Doyle, G. R. Brakenridge, T. A. Erickson, and D. A. Slayback. "Satellite imaging reveals increased proportion of population exposed to floods." Nature 596, no. 7870 (2021): 80-86.

[85] Trysnyuk, V., V. Okhariev, T. Trysnyuk, O. Zorina, A. Kurylo, and C. Radlowska. "Improving the algorithm of satellite images landscape interpretation." In 18th International Conference on

Geoinformatics-Theoretical and Applied Aspects, vol. 2019, no. 1, pp. 1-5. European Association of Geoscientists & Engineers, 2019.

[86] Ulacha, Grzegorz, Ryszard Stasiński, and Cezary Wernik. "Extended Multi WLS Method for Lossless Image Coding." Entropy 22, no. 9 (2020): 919.

[87] Unno, Kyohei, Yusuke Kameda, Ichiro Matsuda, Susumu Itoh, and Sei Naito. "Lossless Image Coding Exploiting Local and Non-local Information via Probability Model Optimization." In 2019 27th European Signal Processing Conference (EUSIPCO), pp. 1-5. IEEE, 2019.

[88] Vaseghi, Behrouz, Seyedeh Somayeh Hashemi, Saleh Mobayen, and Afef Fekih. "Finite time chaos synchronization in time-delay channel and its application to satellite image encryption in OFDM communication systems." IEEE Access 9 (2021): 21332-21344.

[89] Vasiliev, Konstantin, Vitaly Dementiev, and Nikita Andriyanov. "Representation and processing of multispectral satellite images and sequences." Procedia Computer Science 126 (2018): 49-58.

[90] Venkatesan, Aparna, James Lowenthal, Parvathy Prem, and Monica Vidaurri. "The impact of satellite constellations on space as an ancestral global commons." Nature Astronomy 4, no. 11 (2020): 1043-1048.

[91] Voicu, Andra M., Abhipshito Bhattacharya, and Marina Petrova. "Towards Global and Limitless Connectivity: The Role of Private NGSO Satellite Constellations for Future Space-Terrestrial Networks." arXiv preprint arXiv:2107.10811 (2021).

[92] Walker, C. E. "Explained in 60 Seconds: Impact of Satellite Constellations in Astronomy." Communicating Astronomy with the Public Journal 29 (2021): 4.

[93] Wigner, Anson. Satellite Vision and Atomic Trails. Rochester Institute of Technology, 2021.

[94] Woon, Wee Meng, Anthony Tung Shuen Ho, Tao Yu, Siu Chung Tam, Siong Chai Tan, and Lian Teck Yap. "Achieving high data compression of self-similar satellite images using fractal." In IGARSS 2000. IEEE 2000 International Geoscience and Remote Sensing Symposium. Taking the Pulse of the Planet: The Role of Remote Sensing in Managing the Environment. Proceedings (Cat. No. 00CH37120), vol. 2, pp. 609-611. IEEE, 2000.

[95] Xie, Xufen, Hongda Fan, Anding Wang, Nianyu Zou, and Yuncui Zhang. "Regularized slanted-edge method for measuring the modulation transfer function of imaging systems." Applied optics 57, no. 22 (2018): 6552-6558.

[96] Xin, Gangtao, and Pingyi Fan. "Soft Compression for Lossless Image Coding Based on Shape Recognition." Entropy 23, no. 12 (2021): 1680.

[97] Yacine Zerrouki et al. “Desertification detection using an improved variational autoencoder-based approach through ETM-landsat satellite data,” IEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, 2021.

[98] Yang, Chenghai. "High resolution satellite imaging sensors for precision agriculture." Frontiers of Agricultural Science and Engineering 5, no. 4 (2018): 393-405.

[99] Yoshitomo Matsubara et al. “Supervised compression for resource-constrained edge computing systems”. In: arXiv preprint arXiv: 2108.11898 (2021).

[100] Zabusky, Stacia E. "Boundaries at work: Discourses and practices of belonging in the European Space Agency." In An Anthropology of the European Union, pp. 179-200. Routledge, 2020.

[101] Reuther, Albert, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. 2020. “Survey of Machine Learning Accelerators.” In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–12.

[102] “PYNQ - Python Productivity for Zynq.” n.d. PYNQ - Python Productivity for Zynq. Accessed March 23, 2022. http://www.pynq.io/.

[103] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. Adaptive Computation and Machine Learning Series. London, England: MIT Press. Ch 14, pp. 499-523.

[104] Hinton, Geoffrey E., and James L. McClelland. 1987. “Learning Representations by Recirculation.” In *Proceedings of the 1987 International Conference on Neural Information Processing Systems*, 358–66. NIPS’87. Cambridge, MA, USA: MIT Press.

[105] Treudler, Carl Johann, Heike Benninghoff, Kai Borchers, Bernhard Brunner, Jan Cremer, Michael Dumke, Thomas Gärtner, et al. 2018. “ScOSA - Scalable On-Board Computing for Space Avionics.” In *Proceedings of the International Astronautical Congress, IAC*. https://elib.dlr.de/122492/.

[106] Schwenk, Kurt, Moritz Ulmer, and Ting Peng. 2018. “ScOSA: Application Development for a High-Performance Space Qualified Onboard Computing Platform.” In *High-Performance Computing in Geoscience and Remote Sensing VIII*, 10792:32–44. SPIE.

[107] Lund, Andreas, Zain Alabedin Haj Hammadeh, Patrick Kenny, Vishav Vishav, Andrii Kovalov, Hannes Watolla, Andreas Gerndt, and Daniel Lüdtke. 2022. “ScOSA System Software: The Reliable

and Scalable Middleware for a Heterogeneous and Distributed on-Board Computer Architecture." *CEAS Space Journal* 14 (1): 161–71.

[108] Baldi, Pierre. 2012. "Autoencoders, Unsupervised Learning, and Deep Architectures." In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, edited by Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, 27:37–49. Proceedings of Machine Learning Research. Bellevue, Washington, USA: PMLR.

[109] Le, Joie, Arun Viswanathan, and Yuening Zhang. 2020. "Generating High-Fidelity Cybersecurity Data With Generative Adversarial Networks." In *ASCEND 2020*. ASCEND. American Institute of Aeronautics and Astronautics.

[110] Ye, Haiming, Weiwen Zhang, and Mengna Nie. 2022. "An Improved Semi-Supervised Variational Autoencoder with Gate Mechanism for Text Classification." International Journal of Pattern Recognition and Artificial Intelligence. https://doi.org/10.1142/s0218001422530068.

[111] Socher, Richard, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions." In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 151–61. Edinburgh, Scotland, UK.: Association for Computational Linguistics.

[112] Wang, Baohua, Junlian Huang, Haihong Zheng, and Hui Wu. 2016. "Semi-Supervised Recursive Autoencoders for Social Review Spam Detection." *2016 12th International Conference on Computational Intelligence and Security (CIS)*. https://doi.org/10.1109/cis.2016.0035.

[113] Grau. 2019. *Contributions to the Advance of the Integration Density of CubeSats*. Universitätsverlag der TU Berlin.

[114] "MNIST" n.d. TensorFlow. Accessed August 4, 2022. https://www.tensorflow.org/datasets/catalog/mnist.

[115] Palla, Moloney, and Fanucci. n.d. "Fully Convolutional Denoising Autoencoder for 3D Scene Reconstruction from a Single Depth Image." *2017 4th International*. https://ieeexplore.ieee.org/abstract/document/8248355/.

[116] Toderici, George, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. 2017. "Full Resolution Image Compression with Recurrent Neural Networks." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2017.577.

[117] Mentzer, Fabian, George Toderici, Michael Tschannen, and Eirikur Agustsson. 2020. “High-Fidelity Generative Image Compression.” *arXiv [eess.IV]*. arXiv. http://arxiv.org/abs/2006.09965.

[118] Franzen, Richard W. n.d. “True Color Kodak Images.” Accessed August 5, 2022. http://r0k.us/graphics/kodak/.

[119] Yao, Fupin. 2021. “Machine Learning with Limited Data.” *arXiv [cs.CV]*. arXiv. https://www.gov.uk/government/publications/machine-learning-with-limited-data.

[120] Krutz, David, Rupert Müller, Uwe Knodt, Burghardt Günther, Ingo Walter, Ilse Sebastian, Thomas Säuberlich, et al. 2019. “The Instrument Design of the DLR Earth Sensing Imaging Spectrometer (DESIS).” *Sensors* 19 (7). https://doi.org/10.3390/s19071622.

[121] Manthey, Kristian, David Krutz, and Ben Juurlink. n.d. “A New Real-Time System for Image Compression on-Board Satellites.” Accessed March 30, 2022. https://elib.dlr.de/95110/1/74093_manthey.pdf.

[122] Kaiser, Angela. n.d. “SCIAMACHY Data Processing.” Accessed August 7, 2022. https://www.dlr.de/eoc/en/desktopdefault.aspx/tabid-5447/9970_read-20673/.

[123] Ramdan, Ade, Ana Heryana, Andria Arisal, R. Budiarianto S. Kusumo, and Hilman F. Pardede. 2020. “Transfer Learning and Fine-Tuning for Deep Learning-Based Tea Diseases Detection on Small Datasets.” In *2020 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, 206–11.

[124] Thomas M Cover and Joy A Thomas. Elements of information theory. John Wiley & Sons, 2012

[125] Windrim, Lloyd, Rishi Ramakrishnan, Arman Melkumyan, Richard J. Murphy, and Anna Chlingaryan. 2019. “Unsupervised Feature-Learning for Hyperspectral Data with Autoencoders.” *Remote Sensing* 11 (7): 864.

[126] Gokar, Abhijeet. 2018. “Pavia University Hyperspectral Dataset.” https://www.kaggle.com/datasets/abhijeetgo/paviauniversity.

[127] Windrim, Lloyd, Rishi Ramakrishnan, Arman Melkumyan, and Richard Murphy. 2017. “Hyperspectral CNN Classification with Limited Training Samples.” In *Procedings of the British Machine Vision Conference 2017*. British Machine Vision Association. https://doi.org/10.5244/c.31.4.

[128] Windrim, Lloyd, Arman Melkumyan, Richard Murphy, Anna Chlingaryan, and Juan Nieto. 2016. "Unsupervised Feature Learning for Illumination Robustness." In 2016 IEEE International Conference on Image Processing (ICIP), 4453–57.

[129] "MSTAR Overview." n.d. Accessed August 8, 2022. https://www.sdms.afrl.af.mil/index.php?collection=mstar.

[130] "Intelligence Resource Program." Federation of American Scientists, Accessed August 8, 2022. https://irp.fas.org/index.html.

[131] SpaceNet on Amazon Web Services (AWS). "Datasets." The SpaceNet Catalog. Last modified October 1st, 2018. Accessed on August 9, 2022. https://spacenet.ai/datasets/.

[132] "Zynq DPU Product Guide (PG338)." Xilinx, Accessed August 8, 2022. https://docs.xilinx.com/r/3.2-English/pg338-dpu/Introduction?tocId=7o8Y673V3imhiImm8i69dg.

[133] P. Chatziantoniou, A. Tsigkanos, D. Theodoropoulos, N. Kranitis and A. Paschalis, "An Efficient Architecture and High-Throughput Implementation of CCSDS-123.0-B-2 Hybrid Entropy Coder Targeting Space-Grade SRAM FPGA Technology," in *IEEE Transactions on Aerospace and Electronic Systems*, doi: 10.1109/TAES.2022.3173583.

[134] V. Malviya, I. Mukherjee and S. Tallur, "Edge-Compatible Convolutional Autoencoder Implemented on FPGA for Anomaly Detection in Vibration Condition-Based Monitoring," in IEEE Sensors Letters, vol. 6, no. 4, pp. 1-4, April 2022, Art no. 7001104, doi: 10.1109/LSENS.2022.3159972.

[135] Mijares i Verdu, Sebastia, Johannes Balle, Valero Laparra, Joan Bartrina Rapesta, Miguel Hernandez-Cabronero, and Joan Serra-Sagrista. 2022. "Hyperspectral Remote Sensing Data Compression with Neural Networks." In *2022 Data Compression Conference (DCC)*. IEEE. https://doi.org/10.1109/dcc52660.2022.00087.

[136] J. Fjeldtvedt, M. Orlandić and T. A. Johansen, "An Efficient Real-Time FPGA Implementation of the CCSDS-123 Compression Standard for Hyperspectral Images," in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 10, pp. 3841-3852, Oct. 2018, doi: 10.1109/JSTARS.2018.2869697.

[137] Kamata, Hiromichi, Yusuke Mukuta, and Tatsuya Harada. 2021. "Fully Spiking Variational Autoencoder." *arXiv [cs.NE]*. arXiv. http://arxiv.org/abs/2110.00375.

[138] Bamberg, Lennart, Arash Pourtaherian, Luc Waeijen, Anupam Chahar, and Orlando Moreira. 2021. “Synapse Compression for Event-Based Convolutional-Neural-Network Accelerators.” *arXiv [cs.AR]*. arXiv. http://arxiv.org/abs/2112.07019.

[139] Ma, Hailan, Chang-Jiang Huang, Chunlin Chen, Daoyi Dong, Yuanlong Wang, Re-Bing Wu, and Guo-Yong Xiang. 2020. “On Compression Rate of Quantum Autoencoders: Control Design, Numerical and Experimental Realization.” *arXiv [quant-Ph]*. arXiv. http://arxiv.org/abs/2005.11149.

[140] Lüdtke, Daniel, Karsten Westerdorff, Kai Stohlmann, Anko Börner, Olaf Maibaum, Ting Peng, Benjamin Weps, Görschwin Fey, and Andreas Gerndt. 2014. “OBC-NG: Towards a Reconfigurable on-Board Computing Architecture for Spacecraft.” In *2014 IEEE Aerospace Conference*, 1–13.

[141] Lyons, R.E., Vanderkulk, W.: The use of triple-modular redundancy to improve computer reliability. IBM J. Res. Dev. 6(2), 200–209 (1962)

[142] Tambara, L.A., Akhmetov, A., Bobrovsky, D.V., Kastensmidt, F.L.: On the characterization of embedded memories of zynq7000 all programmable soc under single event upsets induced by heavy ions and protons. In: 2015 15th European Conference on Radiation and Its Efects on Components and Systems (RADECS), pp. 1–4. IEEE (2015)

[143] Brownlee, Jason. 2018. “A Gentle Introduction to LSTM Autoencoders.” Machine Learning Mastery. November 4, 2018. https://machinelearningmastery.com/lstm-autoencoders/.

[144] Wood, Thomas. 2020. “Sigmoid Function.” DeepAI. September 27, 2020. https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function.

[145] Fast Machine Learning Lab. n.d. “Welcome to hls4ml’s Documentation! — hls4ml 0.5.1 Documentation.” Accessed August 19, 2022. https://fastmachinelearning.org/hls4ml/.

[146] Mustafa, Aamir, Aliaksei Mikhailiuk, Dan Andrei Iliescu, Varun Babbar, and Rafal K. Mantiuk. 2021. “Training a Task-Specific Image Reconstruction Loss.” *arXiv [eess.IV]*. arXiv. http://arxiv.org/abs/2103.14616.

[147] F. Li, S. Krivenko and V. Lukin, "An Approach to Better Portable Graphics (BPG) Compression with Providing a Desired Quality," 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT), 2020, pp. 13-17, doi: 10.1109/ATIT50783.2020.9349289.

[148] W. Chen, Q. He, S. Li, B. Xiao, M. Chen and Z. Chai, "Parallel Implementation of H.265 Intra-Frame Coding Based on FPGA Heterogeneous Platform," 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International

Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2020, pp. 736-743, doi: 10.1109/HPCC-SmartCity-DSS50907.2020.00096.

[149] Song, Myungseo, Jinyoung Choi, and Bohyung Han. 2021. “Variable-Rate Deep Image Compression through Spatially-Adaptive Feature Transform.” *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/iccv48922.2021.00238.

[150] T. Jiang and J. Cheng, "Target Recognition Based on CNN with LeakyReLU and PReLU Activation Functions," 2019 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC), 2019, pp. 718-722, doi: 10.1109/SDPC.2019.00136.

[151] Z. Zhang, "Improved Adam Optimizer for Deep Neural Networks," 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), 2018, pp. 1-2, doi: 10.1109/IWQoS.2018.8624183.

[152] Cheng, Zhengxue, Heming Sun, Masaru Takeuchi, and Jiro Katto. 2018. “Performance Comparison of Convolutional AutoEncoders, Generative Adversarial Networks and Super-Resolution for Image Compression.” *arXiv [eess.IV]*. arXiv. http://arxiv.org/abs/1807.00270.

[153] Zuo, Chao, Jiaming Qian, Shijie Feng, Wei Yin, Yixuan Li, Pengfei Fan, Jing Han, Kemao Qian, and Qian Chen. 2022. “Deep Learning in Optical Metrology: A Review.” *Light, Science & Applications* 11 (1): 39.

[154] M. Ahmadi, S. Vakili, J. M. P. Langlois and W. Gross, "Power Reduction in CNN Pooling Layers with a Preliminary Partial Computation Strategy," 2018 16th IEEE International New Circuits and Systems Conference (NEWCAS), 2018, pp. 125-129, doi: 10.1109/NEWCAS.2018.8585433.

[155] D. -A. Nguyen, X. -T. Tran, K. N. Dang and F. Iacopi, "A lightweight Max-Pooling method and architecture for Deep Spiking Convolutional Neural Networks," 2020 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2020, pp. 209-212, doi: 10.1109/APCCAS50809.2020.9301703.

[156] M. Zhang and Q. Ling, "Bilateral Upsampling Network for Single Image Super-Resolution With Arbitrary Scaling Factors," in IEEE Transactions on Image Processing, vol. 30, pp. 4395-4408, 2021, doi: 10.1109/TIP.2021.3071708.

[157] M. Kolarik, R. Burget and K. Riha, "Upsampling Algorithms for Autoencoder Segmentation Neural Networks: A Comparison Study," 2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2019, pp. 1-5, doi: 10.1109/ICUMT48472.2019.8970918.

[158] Joseph Raj, Alex Noel, Lianhong Cai, Wei Li, Zhemin Zhuang, and Tardi Tjahjadi. 2022. "FPGA-Based Systolic Deconvolution Architecture for Upsampling." *PeerJ. Computer Science* 8 (May): e973.

[159] Wojna, Zbigniew, Vittorio Ferrari, Sergio Guadarrama, Nathan Silberman, Liang-Chieh Chen, Alireza Fathi, and Jasper Uijlings. 2019. "The Devil Is in the Decoder: Classification, Regression and GANs." *International Journal of Computer Vision* 127 (11-12): 1694–1706.

[160] "Vivado ML Overview." n.d. Xilinx. Accessed August 22, 2022. https://www.xilinx.com/products/design-tools/vivado.html.

[161] "Vitis AI." n.d. Xilinx. Accessed August 22, 2022. https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html.

[162] Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2016. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." *arXiv [cs.DC]*. arXiv. http://arxiv.org/abs/1603.04467.

List of tables

List of figures

Appendix

A Source code base

Please consider that some code modules may be listed publicly, and others may require access permission. For access permissions, contact office@mse.tu-berlin.de using the designator MSE-22-004. The source code base for this thesis is listed on the following links:

GitLab: https://git.tu-berlin.de/master-thesis/DLR-ScOSA-Autoencoder

Any open-source code which require citation when used among the natively developed code is referenced in the code repository.

B Is there a Devil in the autoencoder?

When it comes to generative neural networks like autoencoders, a human might think that there is a devil in the details of the autoencoder, especially when the results are too good when visualized by eyes. Thus, engineers justified the performance of the autoencoder with symbolic mathematical representations, such as probabilistic formulas and difference images.

In 2019 Google engineers and researchers from University College London published a paper titled “The Devil is in the Decoder” to discuss the autoencoder functionality in-depth, focusing on the decoder to solve the scientific community doubts [159]. The following figures elaborate on the up-sampling process in the convolutional autoencoders.

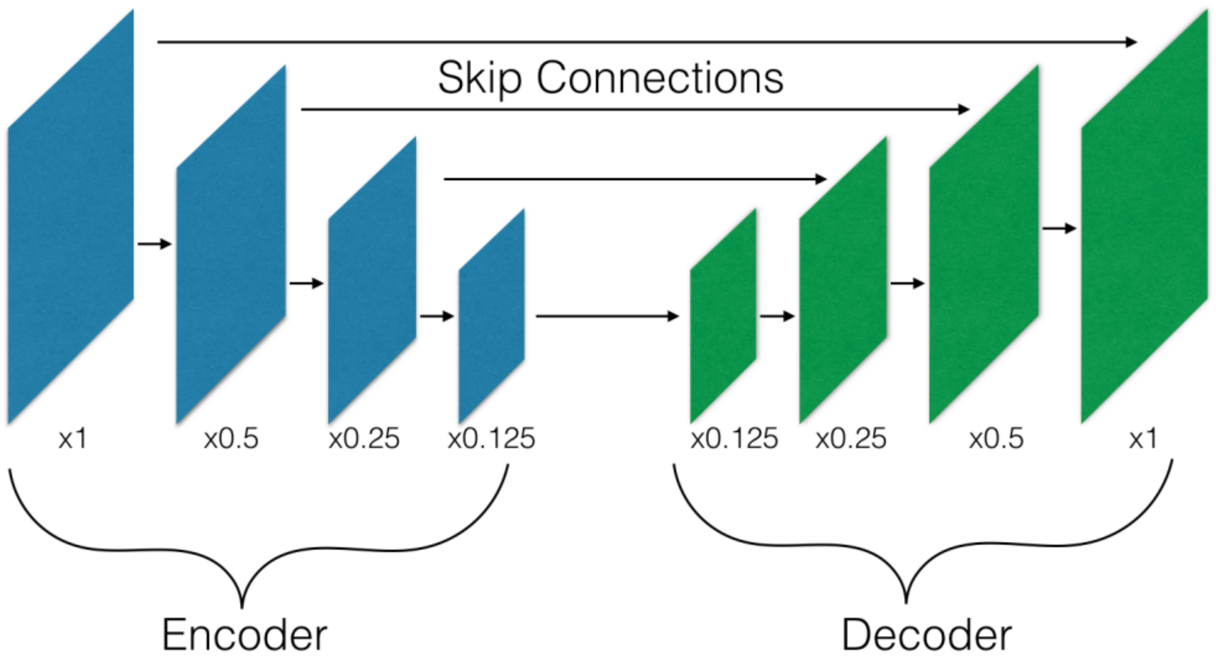
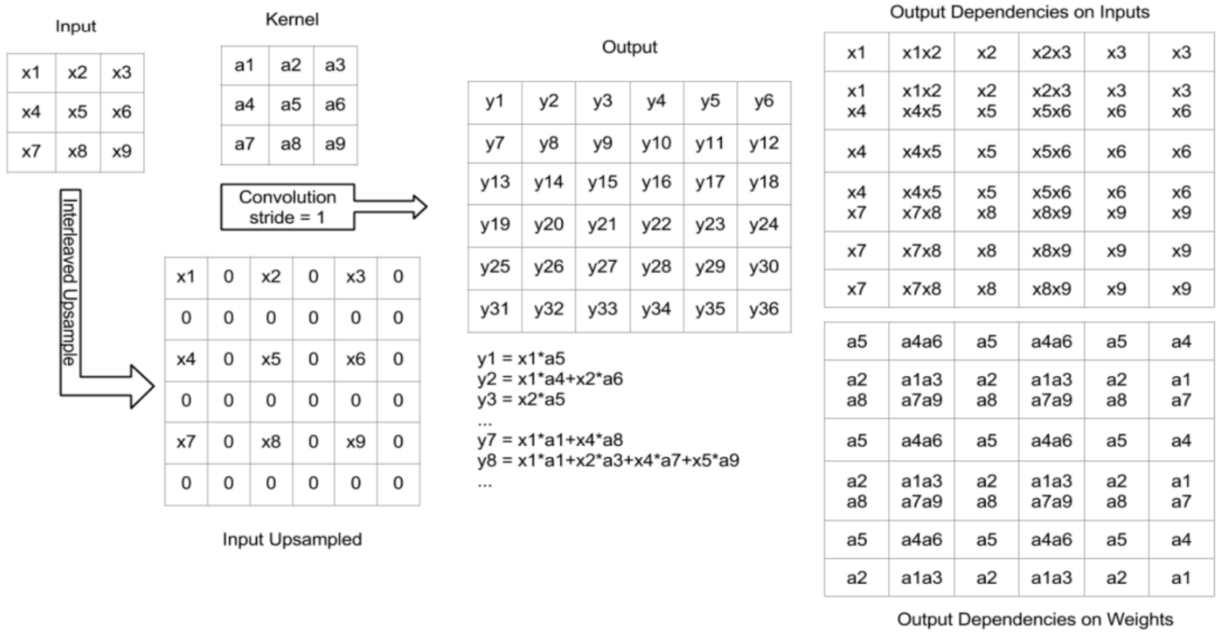


Figure 59. Compression evolution through the autoencoder. [Source: Google, UCL [159]]



Input

x1	x2	x3
x4	x5	x6
x7	x8	x9

Kernel

a1	a2	a3
a4	a5	a6
a7	a8	a9

Input Upsampled

x1	0	x2	0	x3	0
0	0	0	0	0	0
x4	0	x5	0	x6	0
0	0	0	0	0	0
x7	0	x8	0	x9	0
0	0	0	0	0	0

Output

y1	y2	y3	y4	y5	y6
y7	y8	y9	y10	y11	y12
y13	y14	y15	y16	y17	y18
y19	y20	y21	y22	y23	y24
y25	y26	y27	y28	y29	y30
y31	y32	y33	y34	y35	y36

y1 = x1*a5
y2 = x1*a4+x2*a6
y3 = x2*a5
...
y7 = x1*a1+x4*a8
y8 = x1*a1+x2*a3+x4*a7+x5*a9
...

Output Dependencies on Inputs

x1	x1x2	x2	x2x3	x3	x3
x1 x4	x1x2 x4x5	x2 x5	x2x3 x5x6	x3 x6	x3 x6
x4	x4x5	x5	x5x6	x6	x6
x4 x7	x4x5 x7x8	x5 x8	x5x6 x8x9	x6 x9	x6 x9
x7	x7x8	x8	x8x9	x9	x9
x7	x7x8	x8	x8x9	x9	x9

a5	a4a6	a5	a4a6	a5	a4
a2 a8	a1a3 a7a9	a2 a8	a1a3 a7a9	a2 a8	a1 a7
a5	a4a6	a5	a4a6	a5	a4
a2 a8	a1a3 a7a9	a2 a8	a1a3 a7a9	a2 a8	a1 a7
a5	a4a6	a5	a4a6	a5	a4
a2	a1a3	a2	a1a3	a2	a1

Output Dependencies on Weights

Figure 60. Illustration of the Transposed convolution with kernel size 3 and stride 2. [Source: Google, UCL [159]]

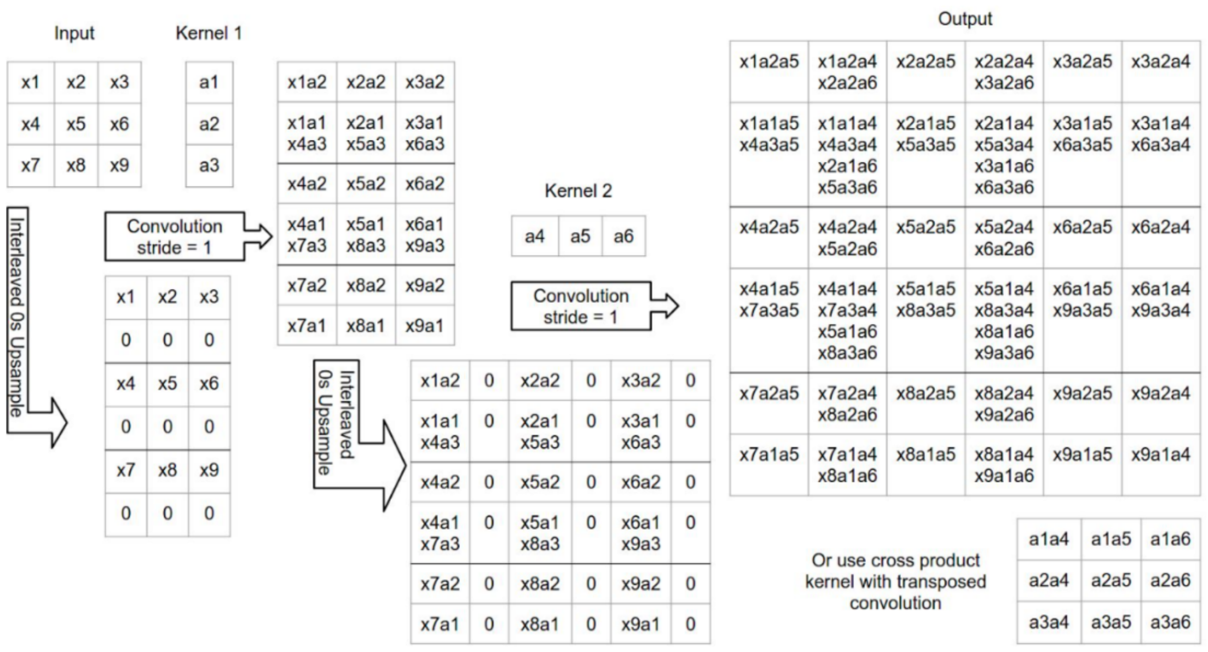
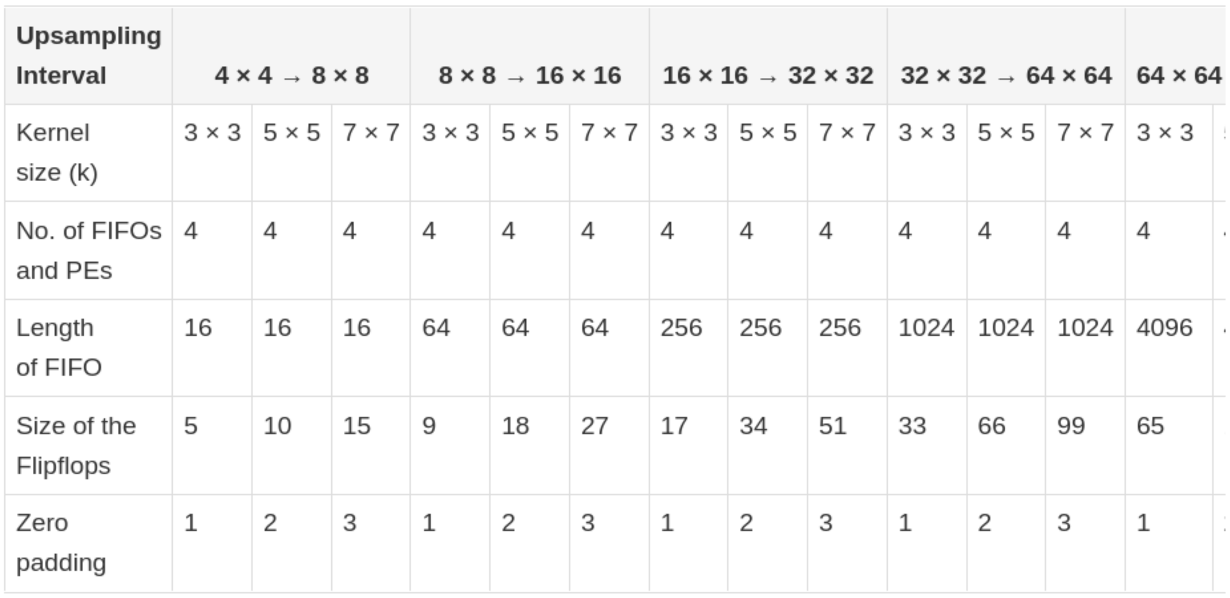

Input
x1 x2 x3
x4 x5 x6
x7 x8 x9
Kernel 1
a1
a2
a3
Interleaved 0s Upsample
Convolution stride = 1
x1 x2 x3
0 0 0
x4 x5 x6
0 0 0
x7 x8 x9
0 0 0
x1a2 x2a2 x3a2
x1a1 x4a3 x2a1 x5a3 x3a1 x6a3
x4a2 x5a2 x6a2
x4a1 x7a3 x5a1 x8a3 x6a1 x9a3
x7a2 x8a2 x9a2
x7a1 x8a1 x9a1
Interleaved 0s Upsample
Kernel 2
a4 a5 a6
Convolution stride = 1
x1a2 0 x2a2 0 x3a2 0
x1a1 x4a3 0 x2a1 x5a3 0 x3a1 x6a3 0
x4a2 0 x5a2 0 x6a2 0
x4a1 x7a3 0 x5a1 x8a3 0 x6a1 x9a3 0
x7a2 0 x8a2 0 x9a2 0
x7a1 0 x8a1 0 x9a1 0
Output
x1a2a5 x1a2a4 x2a2a6 x2a2a5 x2a2a4 x3a2a6 x3a2a5 x3a2a4
x1a1a5 x4a3a5 x1a1a4 x4a3a4 x2a1a6 x5a3a6 x2a1a5 x5a3a5 x2a1a4 x5a3a4 x3a1a6 x6a3a6 x3a1a5 x6a3a5 x3a1a4 x6a3a4
x4a2a5 x4a2a4 x5a2a6 x5a2a5 x5a2a4 x6a2a6 x6a2a5 x6a2a4
x4a1a5 x7a3a5 x4a1a4 x7a3a4 x5a1a6 x8a3a6 x5a1a5 x8a3a5 x5a1a4 x8a3a4 x8a1a6 x9a3a6 x6a1a5 x9a3a5 x6a1a4 x9a3a4
x7a2a5 x7a2a4 x8a2a6 x8a2a5 x8a2a4 x9a2a6 x9a2a5 x9a2a4
x7a1a5 x7a1a4 x8a1a6 x8a1a5 x8a1a4 x9a1a6 x9a1a5 x9a1a4
Or use cross product kernel with transposed convolution
a1a4 a1a5 a1a6
a2a4 a2a5 a2a6
a3a4 a3a5 a3a6

Figure 61. Decomposed illustration of the transposed convolution [Source: Google, UCL [159]]



Upsampling Interval	4 × 4 → 8 × 8			8 × 8 → 16 × 16			16 × 16 → 32 × 32			32 × 32 → 64 × 64			64 × 64
Kernel size (k)	3 × 3	5 × 5	7 × 7	3 × 3	5 × 5	7 × 7	3 × 3	5 × 5	7 × 7	3 × 3	5 × 5	7 × 7	3 × 3
No. of FIFOs and PEs	4	4	4	4	4	4	4	4	4	4	4	4	4
Length of FIFO	16	16	16	64	64	64	256	256	256	1024	1024	1024	4096
Size of the Flipflops	5	10	15	9	18	27	17	34	51	33	66	99	65
Zero padding	1	2	3	1	2	3	1	2	3	1	2	3	1

Figure 62. The requirements of kernel size for up-sampling on FPGA-Based systolic architecture. [Source: PeerJ [158]]

Source code from the author Joseph Raj, Alex Noel (2022), for Vivado implementation of the up-sampling layers on FPGA-BAsed Systolic architecture is listed on the following link:

https://doi.org/10.6084/m9.figshare.13668644.v2

Source code from the author Joseph Raj, Alex Noel (2022), for MATLAB code demonstrations is listed on the following link:

https://doi.org/10.6084/m9.figshare.19387118.v2

C Embedded deployment

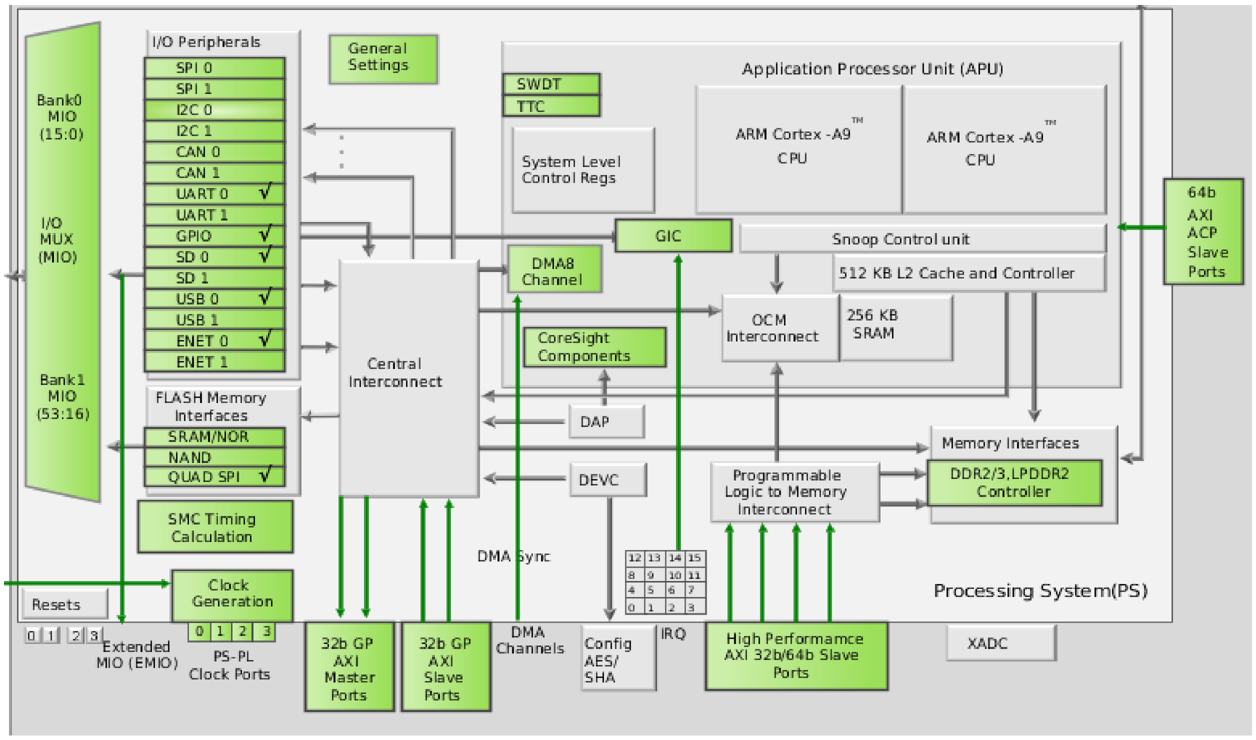


Figure 63. Typical Ultrascale Zynq-7020 SoC [Source: Vivado, Xilinx [160]]

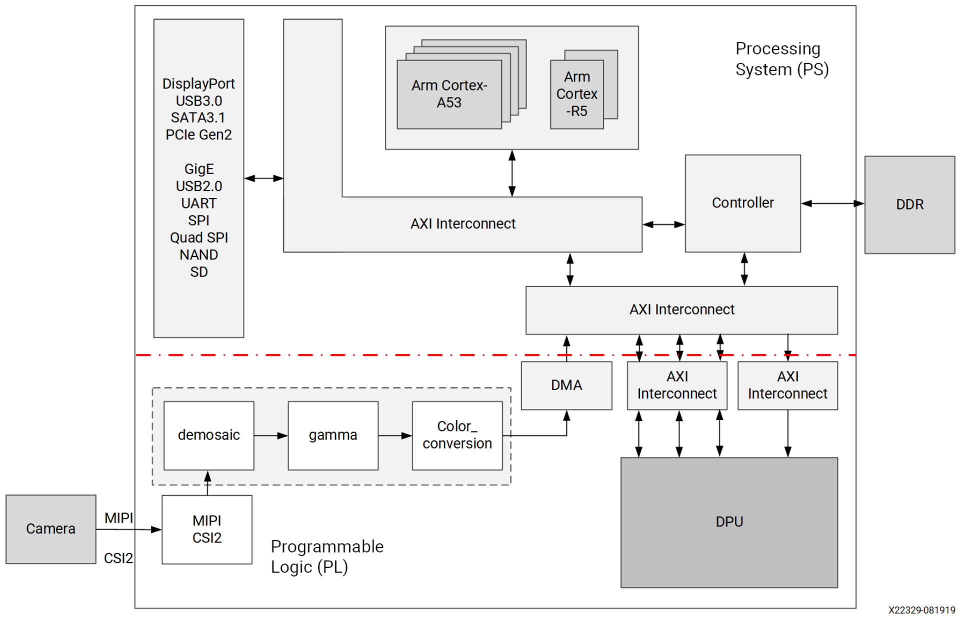


Figure 64. System with Integrated DPU [Source: Zynq DPU Product Guide [132]]

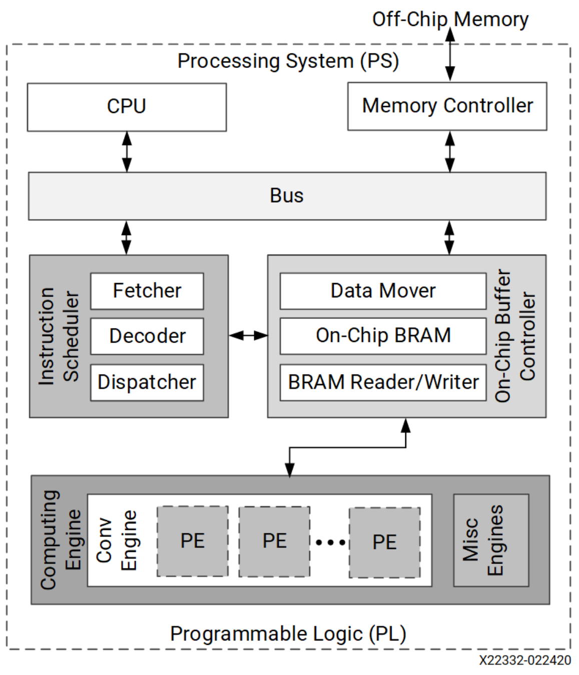


Figure 65. DPU Hardware Architecture [Source: Zynq DPU Product Guide [132]]

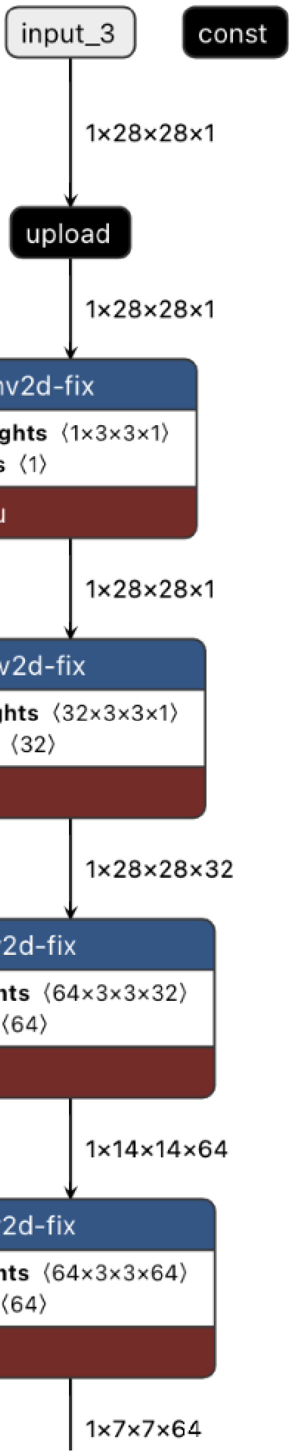


Figure 66. Quantized VAE Autoencoder architecture, part 1. [Source: Author]

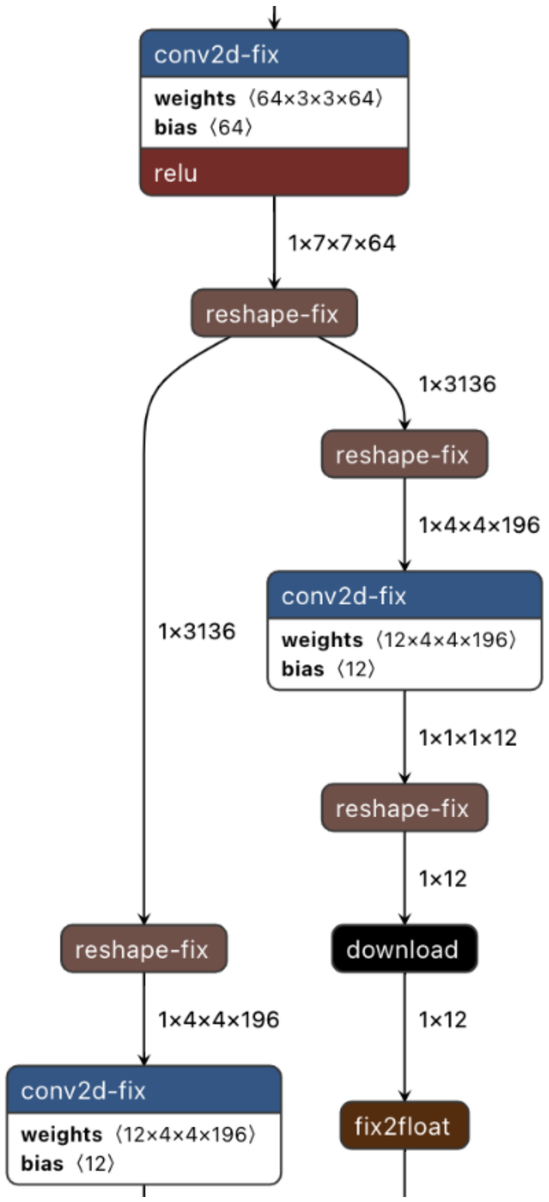


Figure 67. Quantized VAE Autoencoder architecture, part 2. [Source: Author]

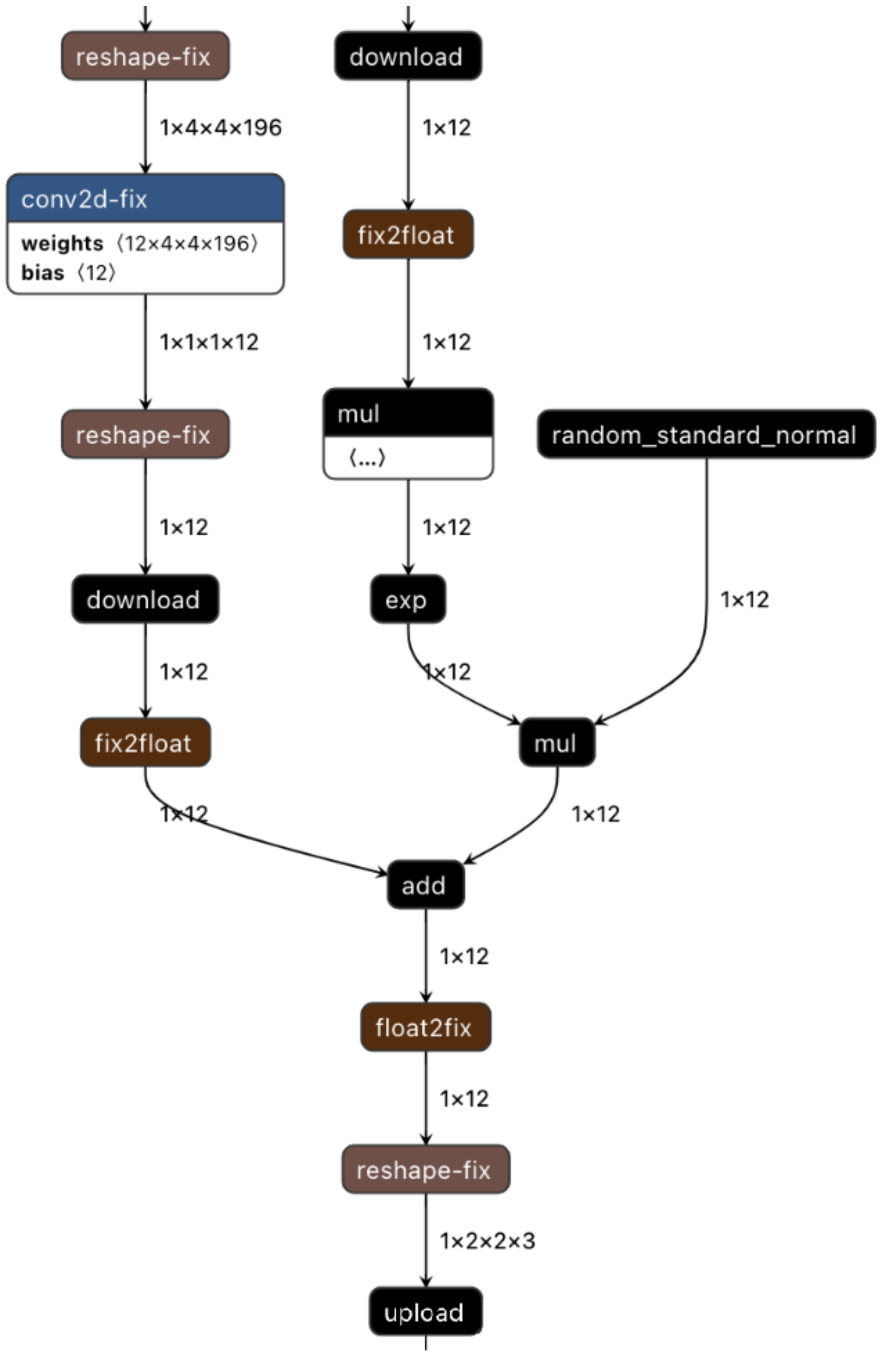


Figure 68. Quantized VAE Autoencoder architecture, part 3. [Source: Author]

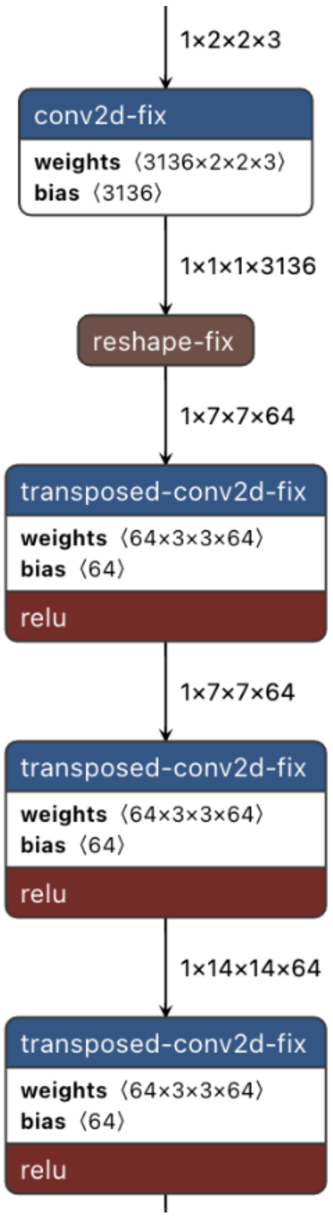


Figure 69. Quantized VAE Autoencoder architecture, part 4. [Source: Author]

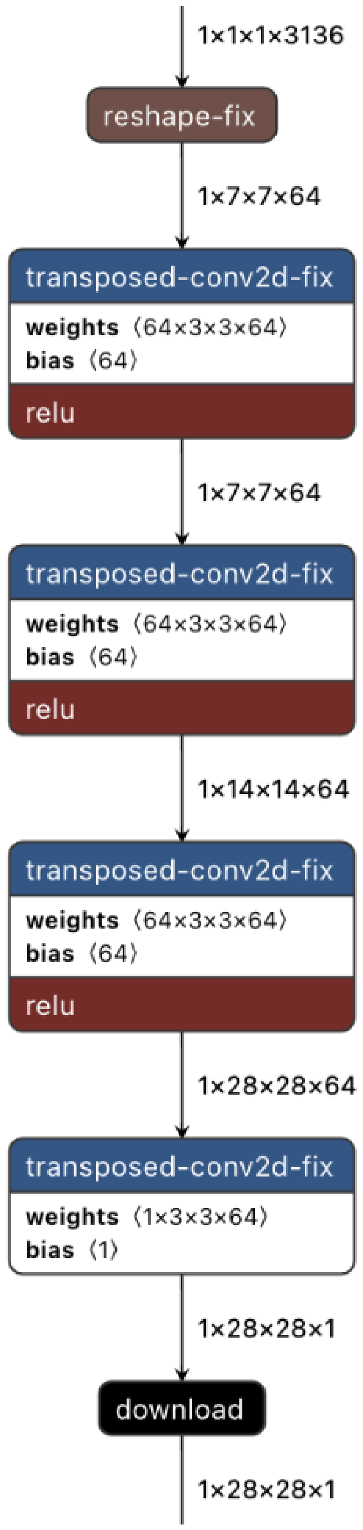


Figure 70. Quantized VAE Autoencoder architecture, part 5. [Source: Author]

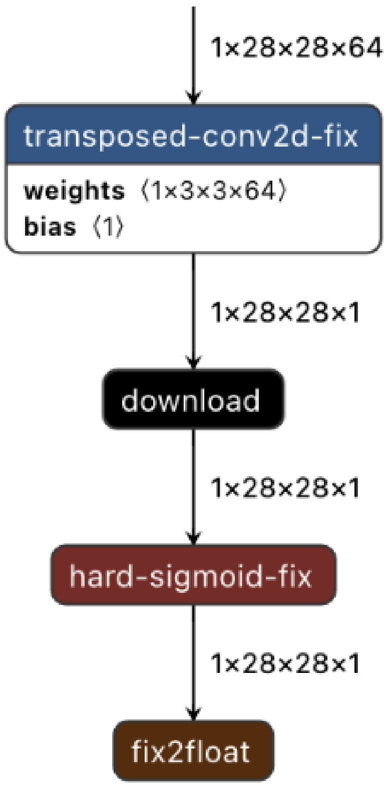


Figure 71. Quantized VAE Autoencoder architecture, part 6. [Source: Author]