Technische Universität Berlin

Faculty of Electrical Engineering and Computer Science
Dept. of Computer Engineering and Microelectronics
Remote Sensing Image Analysis Group

Learning-Based Hyperspectral Image Compression Using A Spatio-Spectral Approach

---

Master of Science in Computer Science
August, 2023

Sprengel, Niklas

Matriculation Number: 380009

Supervisor: Prof. Dr. Begüm Demir

Advisor: Martin Fuchs

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt habe. Sämtliche benutzten Informationsquellen sowie das Gedankengut Dritter wurden im Text als solche kenntlich gemacht und im Literaturverzeichnis angeführt. Die Arbeit wurde bisher nicht veröffentlicht und keiner Prüfungsbehörde vorgelegt.

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, Date

. . . . . .

Sprengel Niklas

Abstract

This template is intended to give an introduction of how to write diploma and master thesis.

Contents

Acronyms

1D One-dimensional

2D Two-dimensional

3D Three-dimensional

ANN Artifical neural network

ASCII American Standard Code for Information Interchange

ASI Italian Space Agency

bpppc Bits per pixel per channel

BTC Block Truncation Coding

CABAC Context-adaptive binary arithmetic coding

CAF Cross-layer adaptive fusion

CCSDS Consultative Committee for Space Data Systems

CNN Convolutional neural network

DCT Discrete cosine transform

DFT Discrete Fourier transform

DualMSE Dual Mean Squared Error

DWT Discrete wavelet transform

EnMAP Environmental Mapping and Analysis Program

ESA European Space Agency

GAN Generative adversial network

GDN Generalized divisive normalization

GMM Gaussian mixture model

GPU Graphical processor unit

GSD Ground sampling distance

H.264/AVC Advanced Video Coding

HEIC High Efficiency File Format

HEVC High Efficiency Video Coding

IGDN Inverse generalized divisive normalization

JPEG Joint Photographic Experts Group

KL Kullback-Leibler

KLT Karhunen-Loeve transform

LeakyReLU Leaky rectified linear unit

MLP Multi-layer perceptron

MSE Mean Squared Error

NLP Natural Language Processing

PCA Principal component analysis

PNG Portable network graphics

PReLU Parameterised Rectified Linear Unit

PRISMA PRecursore IperSpettrale della Missione Applicativa

PSNR Peak signal-to-noise ratio

RD Rate Distortion

RGB Red, green and blue

S2 Sentinel-2

SAM Spectral Angle Mapper

SVM Support vector machine

SWIR Shortwave infrared

TD Tucker decomposition

VAE Variational autoencoder

VNIR Visible and near infrared

List of Figures

List of Tables

1 Introduction

Hyperspectral imaging is a quickly growing field. It is the technique of capturing images with a specialized camera in order to obtain a spectrum of many wavelengths of light for each pixel of a taken image. There are three categories of hyperspectral cameras that are used to capture such images. The first, push broom scanners, use a linear arrangement of spectroscopic sensors, that are sensors able to capture a spectrum of many wavelengths of light at once. The sensor arrangement is then moved over the subject of the image, for example by being attached to a satellite orbiting earth. The second category, whisk broom scanners, functions similarly to push broom scanners with the difference being that the stationary sensor arrangement is replaced by a moving mirror reflecting light into a single detector that collects the spectral information in a step-by-step manner. Lastly, snapshot hyperspectral imaging works by using a sensor array to capture a complete hyperspectral image with a single activation of the sensors.

Regardless of capture method, there are many applications for the use of hyperspectral images since much information can be extracted from the combination of spatial and spectral data in them. One such application is pixel-wise classification of materials, that being the process of determining the material of specific pixels in a hyperspectral image. This is possible because of the high information present per pixel resulting from the large amount of spectral information. An instance of this approach can be seen in Zea et al. [1]. They use classification of hyperspectral images in order to detect the presence of the toxic metal cadmium in soil, lessening the need for chemical methods that require the plant to be harvested to test for cadmium stress. Using hyperspectral imaging the detection of cadmium can be performed on living plants. Another example of pixel-wise classification materials is in Henriksen et.al. [2], where it is used to separate plastic waste by twelve kinds of plastics better than previously used methods such as near-infrared technology.

Another field where the use of hyperspectral imaging is rising in importance is in geology and environmental sciences. An example of this is the Environmental Mapping and Analysis Program (EnMAP) mission which launched a satellite into earths orbit in 2022 that takes images of the surface of the earth with a comparatively high ground sampling distance (GSD) of 30 square meters per pixel, allowing for regional geographic anal-

ysis [3]. In the spectral domain, the satellite yields high resolution. By combining a sensor in the visible and near infrared (VNIR) spectrum and a sensor in the shortwave infrared (SWIR) spectrum it can capture light with wavelengths between 420 and 2450 nm. Wavelengths between 420 and 1000 nm are captured by the VNIR sensor with a resolution of 6.5 nm per spectral band, while wavelengths between 900 and 2450 nm are captured by the SWIR sensor with a resolution of 10 nm per band. The EnMAP mission has already lead to many interesting studies including of glacier ice surface properties of the Ice Sheet in South-West Greenland, moisture content of soil below grassland and identification of specific crop traits such as the chlorophyll content or the leaf water content [4–6]. A dataset created from the images produced by the EnMAP mission is also the main dataset studied in this thesis.

Another ongoing hyperspectral imaging mission is the PRISMA (PRecursore IperSpettrale della Missione Applicativa) mission led by the Italian Space Agency (ASI) that was launched in 2019 [7]. Similar to the EnMAP mission, the PRISMA mission also consists of a satellite that observes earth using both a sensor for the VNIR spectrum and for the SWIR spectrum. It also has a GSD of 30 square meters per pixel and a slightly lower spectral resolution given as less than 12 nm per band [8, 9]. In contrast to the EnMAP mission the satellite also carries a panchromatic camera, meaning that the camera only has a single spectral band. This camera however has a higher GSD of 5 square meters per pixel. The data from the PRISMA mission has been used to increase spatial resolution in hyperspectral images. This was done by combining the original hyperspectral images with multispectral images with a higher spatial resolution in a process called hyperspectral-multispectral fusion [10]. The term 'multispectral image' refers to images with more spectral bands than traditional red, green and blue (RGB) images, but less bands than hyperspectral images. Here the multispectral images contain 10 bands and are obtained from the Sentinel-2 (S2) project, a mission by the European Space Agency (ESA) that continually performs multispectral imaging with fine spatial resolution [11]. Another use of the PRISMA data has been in the disaster monitoring where it has been used to segment wildfires into areas of active fire, smoke, burned ground, bare soil and remaining vegetation [12]. Another related system developed using PRISMA data performs wildfire detection in real-time from satellites [13].

The increasing usage of hyperspectral imaging makes it essential to address the disadvantages of the technology. One major disadvantage of this type of imaging is the required amount of disk space for the resulting images. Because there are many more spectral bands compared to the three bands of RGB in traditional photography the resulting file size rises accordingly. Furthermore, hyperspectral images often use high precision for the specific brightness values captured for each point in the image. The EnMAP satellite images are composed of 32 bit values, while in RGB imaging 8 bits

per pixel per band are most common [3]. A hyperspectral image might for example have 300 bands. In combination with 32 bit values per pixel per band the resulting file size would be 400 times larger than an RGB image with the same pixel density. In addition to this it is important to notice that even RGB images need to be compressed for many use cases in order to be efficiently stored and transmitted, which is why there are many compression standards for RGB images in wide usage. Furthermore, compression of hyperspectral images is a complex problem. This arises from the combination of spatial image compression complexities and the added challenge of encoding information in the spectral dimension which RGB compression algorithms do not consider. The latter is one of the reasons why many RGB compression algorithms do not perform well on hyperspectral data as will be shown in this thesis. In addition to this there are more technical reasons for the difficulty of adapting algorithms for RGB compression that will also be explored in this thesis. For the above mentioned reasons it is clear that it is important to research compression algorithms for hyperspectral images.

Algorithms for image compression can be categorized into multiple broad groups. They can firstly be grouped by whether they compress the image losslessly or with loss. Lossless algorithms restore the compressed image exactly whereas lossy reconstructions cause distortion. The disadvantage of lossless compression is however that there are mathematical limits to the compression rate given by the entropy in the data that is to be compressed. This limit is given by Shannon's source coding theorem [14], which has been stated as follows [15]:

> N independent identically distributed random variables each with entropy $H(X)$ can be compressed into more than $NH(X)$ bits with negligible risk of information loss, as $N \to \infty$; but conversely, if they are compressed into fewer than $NH(X)$ bits it is virtually certain that information will be lost.

This means that the average bitrate, the compression rate given by the quotient of input bits and output bits of the compression algorithm, that is possible to achieve using lossless compression algorithms is given by the entropy of the data that is to be compressed. In contrast, lossy compression methods can achieve much higher compression rates by allowing the introduction of distortion. Furthermore, they can also adapt their rate of compression based on either the desired amount of distortion or the target compression rate. They can also be combined with a lossless compression method to further optimise their compression rate. Another important factor that determines the trade-off between lossless and lossy compression methods is that many applications do not require the perfect reconstruction given by lossless methods. For example, García-Vílchez et al. [16] showed that in hyperspectral image classification lossy compression of the image does not always reduce the performance of the classifier. For some lossy compression

algorithms the classifier even produced better results on the compressed data than the uncompressed data. This is explained by an introduction of smoothing from the compression which is advantageous for the support vector machine (SVM) classifier used in the experiment. For these reasons this thesis focuses on lossy compression methods.

The category of lossy compression algorithms can be further divided into traditional compression methods and learning-based compression methods. Traditional compression methods employ algorithms such as the discrete cosine transform which is used in the Joint Photographic Experts Group (JPEG) image compression standard or the wavelet transform, often in combination with a lossless entropy coding method. Examples of traditional methods other than JPEG are Block Truncation Coding (BTC) which splits images into blocks and then rounds pixel values inside the blocks efficiently and the High Efficiency File Format (HEIC) standard, which uses wavelet transforms similar to JPEG [17, 18]. Learning-based approaches use the powerful capability of artificial neural networks to universally approximate arbitrary functions using gradient descent [19]. These networks are then used to build models that learn to compress and decompress images. Lately these networks have been shown to outperform traditional compression methods for many applications, especially for RGB images using for example the hyperprior model given by Ballé et al. which will be discussed in detail in Section 2.1 [20–22]. Compression for hyperspectral images is in the early stages of research, however even in this domain there are promising results showing the capabilities of learning-based autoencoders for this task [23–26].

1.1 Objective

This thesis addresses the problem of hyperspectral image compression using machine learning models consisting of two parts, an encoder and a decoder. The encoder learns to map the original image to a low-dimensional latent space, thereby performing compression. Analogously, the decoder is trained to reconstruct the input by mapping elements from this latent dimension to full-size images that are as close as possible to the original image.

Contributions to the research in on learning-based hyperspectral image compression are made in this thesis by introducing a new architecture that enables the use of spatial compression algorithms such as models that might be used for RGB image compression by combining them with a model that performs compression in the spectral domain while keeping spatial relationships intact. In this way, it is possible to compress the spatial information in hyperspectral images using models that are not ordinarily applicable to

these images.

Using this architecture, multiple model combinations are designed and compared with each other as well as with the base versions of these models. These models include models based on convolutional neural networks (CNNs) as well as transformer-based architectures. A model using a hyperprior architecture is also used for the spatial compression. This architecture yields much higher compression ratios than other models by using an arithmetic encoder in the bottleneck. With this model compression rates much higher than the current state of the art for hyperspectral image compression are achieved while distortion is only reduced by a comparatively small amount.

Additionally, the latent spaces of both the encoder in the spectral domain as well as the latent space of the spatial encoder will be analysed.

1.2 Outline

This section gives a brief introduction into the chapters in the thesis, which is split into 7 chapters.

Chapter 2 presents the related work. This relates to the research of learning-based hyperspectral image compression by itself. However, as learning-based compression for hyperspectral images, in contrast to traditional hyperspectral compression methods, is a recent and relatively unexplored field of study, the adjacent research topic of learning-based RGB image compression is also explored. There is also some exploration of the general studies done on convolutional neural networks as well as transformers.

Chapter 3 gives an overview of the theoretical ideas used in the proposed methods, such as CNNs, transformers, arithmetic coding as well as the hyperprior architecture for compression.

Chapter 4 details the individual submodels making up the models that are proposed to address the hyperspectral image compression problem as well as the models in their totality. It also gives an overview over the loss functions used for training the models, one of which is a loss function specifically designed for the models proposed in this thesis.

Chapter 5 explains the design of the experiments done, the dataset that is used for these experiments as well as the results of these experiments. It also details some of the

challenges encountered during the experimentation phase.

Chapter 6 gives a summary of the results from the thesis as well as possible improvements that could be made as well as ideas that could be explored in future research.

2 Related Work

Learning-based hyperspectral image compression is a developing field of study. While there are papers published on this topic, there are some problems making it difficult to assess and compare the results of these studies, as will be illuminated further in this chapter. An overview of hyperspectral image compression algorithms by Dua et.al. [27] shows the discrepancy between traditional transform-based and prediction-based compression techniques and techniques based on machine learning. The review contains 21 transform-based and 19 prediction based techniques but only five learning-based models. While the overview over the models based on machine learning is no longer complete since the review was done in 2020, it still shows that learning-based approaches for hyperspectral image compression have been a less researched field until recently. Lately however, research in this field has been increasing quickly SOURCE. Since learning-based red, green and blue (RGB) image compression is a closely related field and much more widely researched, the studies done regarding this topic are also analysed.

2.1 Hyperspectral Image Compression

Most learning-based hyperspectral image compression papers use a convolutional neural network (CNN)-based model architecture to reduce the dimensions of the input image [23–25]. Another model proposed by Guo et.al. [26] uses a hyperprior architecture, originally developed by Ballé et.al. [20], which also uses convolutional layers in an artifical neural network (ANN) but combines them with an arithmetic coder to improve compression rate. Some other models that will be explored later are a model based on a support vector machine (SVM) by Aidini et.al. [28], a generative adversial network (GAN) model by Deng et.al. [29] and a model using a simple multi-layer perceptron (MLP) by Kumar et.al. [30]. Before detailing these methods, an overview of some traditional hyperspectral image compression architectures is given.

2.1.1 Traditional Architectures

Dua et. al. [27] describes eight categories of hyperspectral image compression techniques, one of which are learning-based approaches. Of the traditional methods, that is methods not based on machine learning, the most researched categories are transform-based techniques and prediction-based techniques. Transform-based techniques work by transforming the spatial dimension, the spectral dimension or both into the frequency domain by using a transformation function that is applied to the input image [27]. Possible transformation functions are the Karhunen-Loeve transform (KLT), the discrete cosine transform (DCT), the discrete Fourier transform (DFT) or a discrete wavelet transform (DWT). The transformation can then be used to decorrelate the image in the spatial, spectral or both domains depending on the specific technique used. More specifically, the KLT directly decorrelates the data, while for the other transformation functions additional algorithms are commonly used to decorrelate the coefficients resulting from the transformation [31, 32]. After decorrelation, the coefficients are quantized by removing coefficients that are close to zero [27]. The quantized coefficients are then compressed further using an entropy coding algorithm in the final step of compression. Decompression is possible because the used transformation functions are invertible, although loss is introduced during the quantization step. Some of these general steps may be modified for specific transform-based architectures.

Looking at a specific example of a transform algorithm, Karami et.al. [32] uses a two-dimensional DWT in combination with the Tucker decomposition (TD) algorithm, an extension of principal component analysis (PCA), in order to reduce correlation. They apply the 2DWT to each spectral band of the input image and obtain four tensors containing the coefficients for each band. The TD algorithm is then used to decorrelate the coefficients both in the spatial and spectral domain. The least significant coefficients are then removed using an iterative process before the remaining coefficients are compressed using the entropy coding technique of arithmetic coding.

The other common category of traditional hyperspectral image compression techniques are prediction-based approaches. These approaches compress images by predicting pixel values from the other values of the other pixels using some algorithm and then only storing the prediction residuals [27, 33]. The current compression standard proposed by the Consultative Committee for Space Data Systems (CCSDS), CCSDS 123.0-B2, is a prediction-based approach [34]. The predictor for this standard predicts a pixel value by computing the sum of its neighbours [33]. The difference between these sums and the original pixel values are then stored in a local difference vector which can then be compressed using an entropy coder.

2.1.2 Learning-based Architectures

In contrast to the traditional methods, learning-based architectures use ANNs trained using gradient descent to learn a mapping from the space of input images to a latent space [19]. In image compression, this latent space is lower-dimensional than the input space, thereby performing compression. The most common approach for learning-based hyperspectral compression are CNN-based architectures. These architectures can be split into two categories, the first being the models using two-dimensional (2D) convolutional layers to learn the spatial dependencies of the hyperspectral images [25]. The second category are models using one-dimensional (1D) convolutional layers to learn the spectral dependencies of the input data [23, 24]. Prior to the release of this thesis there are no purely CNN-based papers using both the spatial and the spectral dependencies of hyperspectral images for compression. The model proposed by Guo et.al. [26] does use both spatial and spectral dependencies, it does however use a hyperprior architecture and not a purely CNN-based model. This model as well as the hyperprior architecture will be detailed later in this chapter. As mentioned above, comparing the results from these papers directly is difficult. The reason for this is that the models use different datasets since there is currently no accepted standard dataset for hyperspectral image compression. Furthermore, the models are designed for different compression rates. Since a higher compression rate also leads to a higher distortion for the same model in most cases as described by rate-distortion theory [35], this makes it hard to directly compare the results from papers that use both a different dataset and different compression rates, therefore requiring a reimplementation of the models in order to be able to fairly compare them.

A model exploiting only the spectral dependencies in the input images is provided by Kuester et.al. [23, 24]. They use a model consisting of a 1D convolutional layer to learn spectral dependencies followed by a max pooling layer applied to the spectral dimension to reduce the dimensionality of the image. This block of 1D convolutional layer and a max pooling layer is repeated, using leaky rectified linear unit (LeakyReLU) as the activation function. Following that, two more 1D convolutional layers are added, now without max pooling layers inbetween. Each convolutional layer uses less filters than the layer before it. Finally, the last layer uses only one filter, thereby providing the bottleneck of the compression algorithm with a fixed compression rate of 4. Decompression is performed using a reversed model, symmetrical to the encoder. The decoder only differs by replacing the blocks consisting of a convolutional and a max pooling layer by transposed 1D convolutional layers with a stride of two to increase the spectral dimensionality instead of reducing it [24]. In an earlier variation of the model, upsampling layers are used instead for this purpose [23]. In contrast to most other models, this model is trained and applied not to a whole image at once, but rather each pixel individ-

ually. This has the advantage of reducing the complexity of the task the model has to perform. Instead of compressing a complete hyperspectral image, the model only learns to compress the spectral signatures of arbitrary pixels from these images. The number of training samples also increases dramatically, as each hyperspectral image consists of many pixels, although each training sample contains much less information than it does for a model that trains on complete images at once. A disadvantage of the approach is that modelling spatial dependencies is not possible using a per-pixel training approach. Furthermore, training is much slower when compared to other models. The reasons for this and more detailed comparisons with other model architectures are explored in Chapter 5.

In contrast to the aforementioned model, La Grassa et.al. [25] propose a model that focuses on the spatial dependencies in the image data. This model is trained and applied on complete hyperspectral images in one step. They use a combination of 2D convolutional layers and max pooling layers applied to the spatial dimension of a input image to reduce the spatial dimensionality while simultaneously increasing the number of filters in the convolutional layers to keep the amount of stored information stable. After multiple of these blocks, the output of the last block is flattened into a single dimension, followed by a linear layer that maps that output into a one-dimensional latent space of the chosen size. The decoder is again a symmetrical, reversed version of the encoder, replacing only the blocks of 2D convolutional and max pooling layers by transposed convolutional layers with a stride of 2. This model has the advantaqe of having the ability to set a precise bit rate by changing the output dimensions of the final linear layer. However, applying the model to hyperspectral images with many bands is difficult. This is because in the proposed model, the first convolutional layer uses 64 filters. Since this layer is applied to the complete input image, an input image with dimensions (C,H,W) where C is the number of spectral bands and H and W are the height and width of the image respectively is transformed into the dimensions (64,H,W). If C is significantly higher than 64, this leads to large information loss in the first layer. This is common in hyperspectral datasets. This can be seen in the HySpecNet-11k dataset [36], the main dataset used in this thesis, which uses 202 channels. Possible solutions to this problem are explored in Chapter 4.

There are also some approaches that do not use a CNN-based architecture. While some of these architectures use models that contain a CNN in addition to other parts, some architectures use different model types completely. Among these are both different types of neural networks as well as learning-based algorithms not using neural networks. One such architecture is proposed by Aidini et.al. [28]. They use quantization to compress the original image, meaning that the precision of the spatial and spectral values of the image are decreased. Then an algorithm tries to recover the original tensor values by try-

ing to reconstruct low-rank tensors as a constrained optimization problem. Afterwards a spatial super-resolution algorithm using trained dictionary learning is used to increase the resolution of this image, after which a classifier is trained on these super-resolved images. While this architecture is interesting, it is not directly used in this thesis as its methodology is very different to the neural network based methodologies used in the thesis.

Another category of models use ANNs to determine parameters for lossless compression algorithms. Shen et.al. [37] use a deep belief network to determine the optimal parameters for golomb-rice coding, a lossless coding algorithm that normally assumes a geometric underlying distribution. Using a neural network to determine the parameter removes that necessity. This core strategy is also used by Guo et.al. [26]. They use a hyperprior architecture to compress hyperspectral images. Hyperprior models use an ANN-based model to transform the image data into a latent space that is commonly lower-dimensional than the input image. Then a second ANN is trained on the latent space to determine parameters for an arithmetic coder, a lossless coding algorithm. In both Guo et.al. and the original paper introducing the hyperprior architecture for RGB images, Ballé et.al. [20], both ANNs are CNNs and the latent space is indeed a lower-dimensional space.

Guo et.al. innovates on the original approach in the fact that they assume a student's T distribution instead of a gaussian distribution for the arithmetic coder and in the design of the first CNN. Their version includes both a spatial and a spectral part in the main CNN, making it the only model to learn both spatial and spectral information for hyperspectral image compression. They achieve this by using 2-D convolutional layers for the spatial domain and 3-D convolutional layers with a kernel size of one to include the spectral dependencies. However, a disadvantage of their model is that it is developed only for datasets with a low amount of channels compared to other hyperspectral datasets with the highest having 102 spectral channels. The approach is also not easily adaptable to datasets with a much higher number of channels. This is because the first convolutional layer of the model uses 192 filters and a stride of two, therefore transforming an input image with the dimensions (C, H, W), where C is the number of spectral channels and H and W are the height and width respectively, into the dimensions (192, H/2, W/2). For one of the two datasets studied in Guo et.al., the KAIST dataset, which contains 31 spectral channels, this is a large increase in information. For the other dataset, ROSIS-Pavia, with 102 to 103 spectal channels depending on the scene, this is already a loss of information, requiring the first layer to already encode some of the spatial information of the image appropriately in order to not introduce large amounts of distortion. For datasets with more channels this problem increases. Furthermore, because of graphical processor unit (GPU) memory limitations, increasing the amount

of filters appropriately is not always possible.

Another model using neural networks is proposed by Kumar et.al. [30]. Instead of CNNs they use a simple multi-layer perceptron as the decoder for the reason that they use this model for real-time onboard image compression which requires a much more simple model for faster execution speed. Another uncommon trait of their architecture is that they do not use a symmetric model, meaning that the encoder and decoder are mirrors of one another. Instead, they only use an ANN for the decoder and use a low-complexity encoder based on matrix multiplication. Hong et.al. [38] propose an novel architecture as well. They use a transformer that works on a spectral embedding by linearly projecting groups of neighbouring spectral bands to an embedding vector. This improves the capabilities of the network since neighbouring bands in hyperspectral images capture detailed changes in the absorption of the underlying material and therefore contain important information, especially for classification tasks. In addition to this they implement cross-layer adaptive fusion (CAF) to improve exchange of information in the transformer section of the model. This means that they use multiple transformer layers and, in addition to the direct connection between adjacent transformer layers, add connections that skip one layer and connect with the layer after using a special CAF module. The transformers can be applied either per-pixel or for small patches. However, their work is not used in the context of image compression but rather image classification and therefore only contains an encoder combined with an MLP head to classify hyperspectral image pixels or patches based on categories such as ”Corn”, ”Grass Pasture” or ”Wheat”. This means that an application of this architecture to the task of image compression would require substantial additions to the model.

2.2 RGB Image Compression

While the study of hyperspectral image compression strongly increased in recent years, RGB image compression has been a widely researched field for many years. There are many traditional compression algorithms, some of which are installed on every modern operating system and browser. Examples of these are portable network graphics (PNG), a lossless image compression format as well as Joint Photographic Experts Group (JPEG), a collection of compression algorithms, the most common of which performs lossy compression of images. An improved version of JPEG, JPEG 2000, also exists, offering increased compression rate for similar distortion values [39]. However, while these algorithms are very popular, learning-based image compression has outperformed methods such as JPEG in both compression ratio and distortion [20, 21]. Furthermore, many of the ideas in learning-based hyperspectral image compression

originate from RGB image compression studies and many of the ideas in RGB image compression have not yet been adapted to the hyperspectral realm. For these reasons RGB image compression papers are directly relevant even for a thesis that only concerns itself with the hyperspectral domain.

2.2.1 The Hyperprior Architecture

One of the most important recent works in RGB image compression was released by Ballé et.al. [21]. The model proposed in this builds on a previous work [20] using a CNN to reduce the dimensionality of the input image, followed by a quantization of the resulting latent and the usage of an arithmetic autoencoder to losslessly compress the quantized latent. The output of the arithmetic autoencoder is then decoded by a CNN that is symmetrical to the encoder CNN, similar to the models for hyperspectral image compression that were already discussed. This model already outperformed JPEG and the improved JPEG 2000 on the tested data.

The performance of the arithmetic autoencoder depends on the accuracy of the estimated probability distribution that is used to compress the given data. This area is where improvements were found. Ballé et.al. [21] used a smaller, separate CNN that learns to extract parameters for a good probability distribution estimate from the latent resulting from the main CNN. This network also uses an autoencoder structure, meaning that the estimate can also be transmitted in compressed form as side channel information using only a small amount of space. Training a network using gradient descent for this purpose would not ordinarily be possible since the quantized latents have discrete values resulting in zero gradients everywhere. For this reason the quantization is substituted during training by addition of a small amount of uniform noise to dediscretize the latents.
This hyperprior architecture is part of a large portion of modern learning-based RGB image compression models and also the hyperspectral image compression model by Guo et.al. [26] that was discussed in Chapter 2.1.

One such example is a paper by Hu et.al. [40] where the model is generalised to include not only two but an adaptable number of CNNs, where each CNN learns the probability distribution of the CNN before it. This leads to slight improvements in the bitrate of the model while not changing the distortion. The distortion remains unchanged since the only loss occurs within the first CNN which compresses the input image . The other CNNs are only used to improve the performance of the lossless arithmetic coders.

The architecture was also further improved in two ways by Minnen et.al. [22]. The first

improvement is a generalisation in the structure of the probalitiy distribution from the original scale mixture of gaussians [41] to a gaussian mixture model. This means that the second CNN generating the probability distribution parameters has to predict both a scale and a mean instead of only a scale as before. This allows for a better modeling of the true underlying distribution and the paper shows that the increase in necessary side channel information is lesser than the improvement created by the improvement in probability distribution prediction. The second new idea is the addition of an autoregressive model over the latents of the first, main CNN. This also improves compression performance as the more structure from the latents can be exploited, it does however also increase the computational costs of the network as autoregressive models cannot be trained in parallel.

Another application of the hyperprior architecture is found in Cheng et.al. [42]. They improve on the hyperprior model including a joint autoregressive model given by Minnen et.al. [22] by introducing self-attention modules into the encoder and the decoder CNN. Self-attention modules are an idea taken from the field of Natural Language Processing (NLP). There it is used as an integral part of the novel transformer architecture, which yields state of the art results in machine translation and other NLP fields [43]. In computer vision, it has been used in order to generate attention masks that enable the model to use more bits for the more important parts of an image such as foreground or high-contrast elements and less bits to the less important parts of an image [44–46]. This approach of using self-attention modules is also the one used by Cheng et al. They model the architecture of their self-attention blocks by using a simplified version of the architecture proposed in Liu et al. [44]. The simplification is achieved by removing the non-local blocks, which reduces the training time by four times in their testing while achieving a similar loss [42].

3 Theoretical Foundations

3.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of artifical neural network (ANN), distinguished from other ANNs by the fact that they use convolutional layers as part of their architecture [47]. They were originally designed for the purpose of image classification, which remains as one of the main applications for CNNs [48]. However, CNNs are also used in other fields of machine learning such as Natural Language Processing (NLP). The most common type of CNN in the field of computer vision uses two-dimensional (2D) convolutional layers in combination with subsampling or pooling layers and activation functions in order to learn features of an input image.

A 2D convolutional layer takes a 2D image-like input and produces a given number of features each represented as a 2D matrix [48]. These features are computed for a given pixel by convolving the neighbourhood of this pixel with a kernel, which can be seen in Figure 3.1. These kernels are comprised of the weights of the convolutional layer and are therefore learned using gradient descent. A pooling layer is then used to reduce the spatial dimensionality of the input. This set of a convolutional and a pooling layer can be repeated multiple times, since the output of both the convolutional layer and most pooling layers are again image-like and can be used as input for additional such layers. Between sets of convolutional and pooling layers nonlinear activation functions are applied in order for the network to be able to learn nonlinear features. The combination of multiple sets of convolutional and pooling layers is central to learning-based image compression and is used in many of the proposed models [20, 21, 23, 24, 26].

Compared to other neural network architectures such as multi-layer perceptrons (MLPs), CNNs have low complexity because of the small number of connections between neurons, as each output neuron of a convolutional layer is only connected to a small number of neurons from the input of the convolutional layer, defined by the kernel size [48]. The low complexity results in CNNs being resilient to overfitting [47, 48]. However, regardless of the low complexity, CNNs are still able to learn high-level features of the input

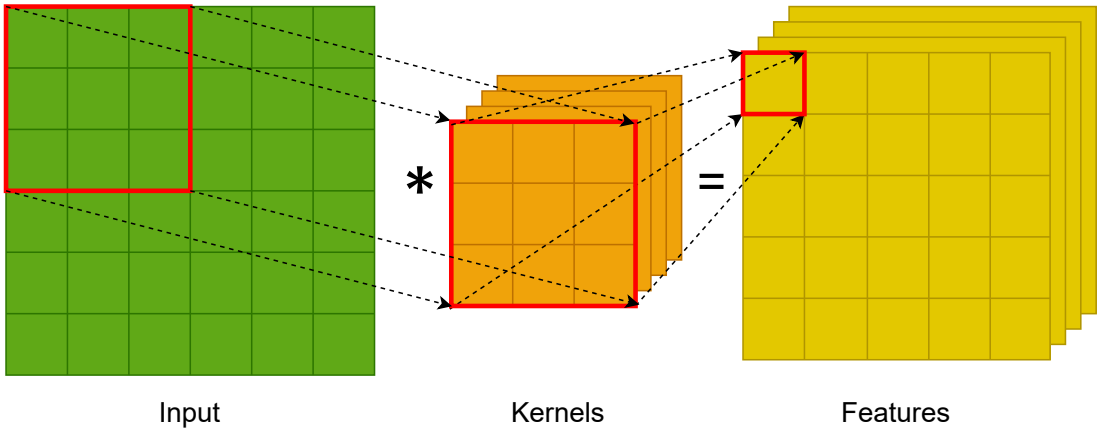

Figure 3.1: Example of a two-dimensional (2D) convolutional layer computing a feature pixel.

because the use of multiple convolutional layers allows for a hierarchical learning of features, allowing the early convolutional layers to learn to detect low-level features such as corners or edges while the later convolutional layers learn higher-level features [48]

In addition to the 2D convolutional layer, there are other variations of convolutional layers that are useful, in particular for analysing hyperspectral image data. Firstly, one can define a one-dimensional (1D) convolutional layer, which differs from the 2D layer in that the kernel moves only in a single dimension. In hyperspectral image analysis this can be used to learn spectral information, since this information is 1D [23, 24]. Similarly, three-dimensional (3D) convolutional layers can be defined by using a 3D cube-shaped kernel. In hyperspectral image processing, this type of layer can be used to learn features of the spatial and spectral domain within a single layer [26].

3.2 Self-Attention modules

The self-attention mechanism is a technique that is widely used in the field of NLP. For example, Vaswani et.al. [43] used it as the main part of their proposed transformer architecture which outperformed other machine-learning approaches in various NLP tasks at the time, such as language translation. The mechanism aims to compute a latent representation of an input sequence by learning to relate the elements of the input sequence to each other. While a CNN can also achieve this, in a CNN only elements close to

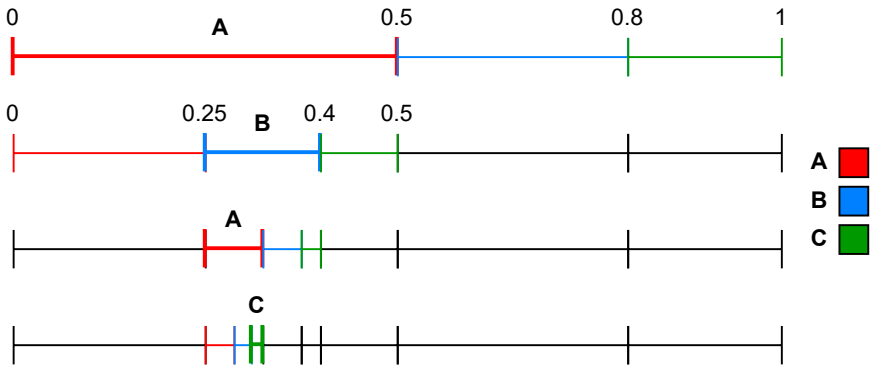

Figure 3.2: Arithmetic coding process of ”ABAC”.

one another are used to compute the latent representation. Using self-attention, the relation of all elements with all other elements can be incorporated, enabling learning of long-distance dependencies [43].

In recent years, self-attention modules have also been used in computer vision. In image compression for example, self-attention modules can be used in order to help the neural network focus on the more complex parts of the input image and subsequently use less bits for the compression of the less complex parts of the input, improving the compression overall [42, 44].

3.3 Arithmetic Coding

Arithmetic coding is an algorithm used for entropy coding, that being a technique for encoding and decoding a sequence of symbols in an efficient way [49]. As an example, a string of characters can be considered. A standard non-optimized way of storing a string is by using the American Standard Code for Information Interchange (ASCII) encoding which used a fixed seven bytes per character of the string. Entropy coding algorithms improve on this by allocating fewer bits to more common characters and therefore more bits to less common characters [49]. In total, this leads to a lower number of required bits for the total message, given an accurate probability model for the frequencies of the characters. In fact, if the provided probability model is perfectly accurate, arithmetic coding yields an encoding close to an optimal encoding, that being an encoding that uses a number of bits equal to the entropy of the encoded input [49]. This is the fewest number of bits possible for a lossless encoding of an arbitrary message, which was proven with Shannon’s source coding theorem [14, 15].

Arithmetic coding, using the theoretical version of the algorithm, works by encoding the entire input message into one arbitrary-precision rational number q with $0 \leq q < 1$ [50]. In practice the algorithm is slightly modified to account for the lack of infinite precision in computers, as will be detailed later in this section. The encoding is performed using an iterative process. In order to understand this process, consider for an example an alphabet consisting of three symbols, 'A', 'B' and 'C' with 'A' having the probability 0.5, 'B' having the probability 0.3 and therefore 'C' having the probability 0.2. Encoding an message is performed symbol by symbol. To encode the first symbol, the interval $[0,1)$ (meaning numbers a with $0 \leq a < 1$) is split into three parts according to these probabilities. If the first symbol is 'A', the final encoding number q will be in the interval $[0,0.5)$. Likewise, if the first symbol is 'B' or 'C', q will be in the intervals $[0.5,0.8)$ or $[0.8,1)$ respectively. The second symbol is then encoded by splitting the interval determined from the first symbol again, using the same probabilities. A sequence of two 'A' symbols would therefore result in q being in the interval $[0,0.25)$. This procedure is iterated until each symbol of the input message is encoded, resulting in a single number q representing the entire input. This process can be seen in Figure 3.2 using the aforementioned probabilities for the word "ABAC". This number encodes the original message nearly optimally given a perfect probability distribution, with a difference of less than a bit to the entropy of the message. The reason for this discrepance is that q is always represented using an integer amount of bits. If the input image has a non-integer entropy, the required amount of bits to store q is the entropy rounded up to the nearest integer. However, the complete algorithm of arithmetic encoding requires an additional source of information that needs to be transmitted to the decoder. In order to decode the message, the decoder needs to know the probability model as well as a way to distinguish when the message ends. The first is constant with respect to message length. The second can be transferred using only logarithmic overhead with respect to message length. This is often done by adding an additional token that signifies the end of the message [50].

As mentioned before, the algorithm is adapted slightly in practice in order to use fixed precision numbers instead of arbitrary precision numbers. Rounding operations as a result of using fixed point numbers are the same for encoder and decoder and therefore do not inhibit the algorithm in principle. However, as a result of rounding, intervals can become small enough that they are not representable with fixed point numbers of the used precision. In this case, the intervals are changed in a determinable way so that they are again representable. This process is called renormalization and multiple algorithms to perform this procedure exist. Arithmetic coding also allows for the possibility of changing the probability distribution during the encoding process based on previously encoded symbols. This is possible because the decoder decodes the symbols in the same order as the encoder encodes them and can therefore perform the same change to the

probability distribution as the encoder, as long as the method for changing the probability model was decided beforehand [50]. An example of such an algorithm is context-adaptive binary arithmetic coding (CABAC), which is an entropy coding method used in the video encoding standards Advanced Video Coding (H.264/AVC) and High Efficiency Video Coding (HEVC) as well as multiple learning-based red, green and blue (RGB) compression models [20–22]. CABAC is a binary encoding method, therefore requiring the data that is to be encoded to be transformed beforehand. It works by using multiple probability models and, for each element that is to be encoded, selecting one of these models based on previously encoded elements [51]. Since the choice of probability model for each element is only based on elements that were encoded before it, decoding is still possible by performing the same probability model choices as the encoder. Further more, the probability models are updated after each encoding step to adapt them to the data.

In learning-based lossy image compression, arithmetic coding is used as part of the Hyperprior architecture order to encode the latent representation of an input image [20–22]. The latent representation of the input image and the probability distribution used for the arithmetic coder can be learned using ANNs.

3.4 Hyperprior Architecture

The hyperprior architecture was first proposed in Ballé et.al. [21] to improve on an earlier work by Ballé et.al. [20], both in the field of grayscale and RGB image compression. In the earlier work, a CNN using 2D convolutional layers, downsampling layers and generalized divisive normalization (GDN) as the activation function was used to transform an input image into a latent representation. This network is called the analysis transform. The latent representation was then compressed using an arithmetic coder. Afterwards, another CNN is used to transform the decoded output from the arithmetic coder. This CNN, called the synthesis transform, is symmetric to the analysis transform in that it contains the same convolutional layers in inverse order. Additionally, instead of downsampling layers and GDN upsampling layers and the activation function inverse generalized divisive normalization (IGDN) is used. The whole model is then trained using a loss function called rate-distortion loss that optimizes for compression rate and reconstruction distortion simultaneously. Since the arithmetic coder performs lossless compression and the compression rate of the CNN is fixed, optimizing the compression rate corresponds to optimizing the probability distribution estimate of the arithmetic coder and optimizing the distortion corresponds to optimizing the quality of the reconstruction produced by the CNNs. A disadvantage of this approach is that the probability

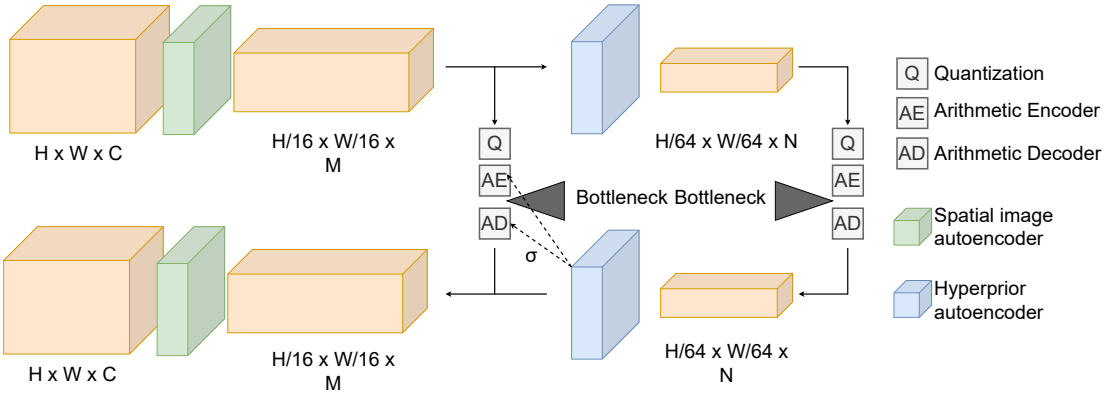

Figure 3.3: Architecture of the variational scale hyperprior model proposed in [21].

distribution used for the arithmetic coder is a fully factorized distribution, which is not optimal if the latent representation computed by the analysis transform contains statistical dependencies [21].

The hyperprior architecture attempts to solve this problem by transmitting information about the probability distribution of the image that is being compressed as side channel information [21]. This is in contrast to the former model, in which only the probability distribution of the training dataset as a whole is known to both encoder and decoder. They noticed that when modeling the probability distribution using gaussians, the standard deviations of latent pixels near each other in specific images tend to vary from the overall probability distribution similarly. Note that while the latent vectors are not images, they can be interpreted as images by treating the filters as different channels of an image with multiple channels. In this way it is also possible to refer to pixels of latents, referring then to the values of all filters for a specific point in the spatial dimensions. To address the spatial dependencies of the standard deviations of the latents, an additional CNN was added to exploit the spatial dependencies in the latents [21]. This network is called ”hyperprior”. The hyperprior network is also split into an analysis and a synthesis transform and is therefore similar to a smaller version of the original CNN. A visualisation of this architecture is shown in Figure 3.3. A difference between the networks is that the hyperprior network uses leaky rectified linear unit (LeakyReLU) as the activation function instead of GDN. This makes sense because GDN is shown to be especially useful for improving visual similarity in an image compression network. This is not useful for the hyperprior network since it does not attempt to learn image reconstruction. Instead, the hyperprior network is used to learn the probability distribution of the latents of the main CNN. More specifically, the hyperprior analysis transform produces a bottleneck tensor, which can be compressed with an arithmetic coder using a fully factorized distribution similar to the earlier work by Ballé et.al. [20]. The hyperprior

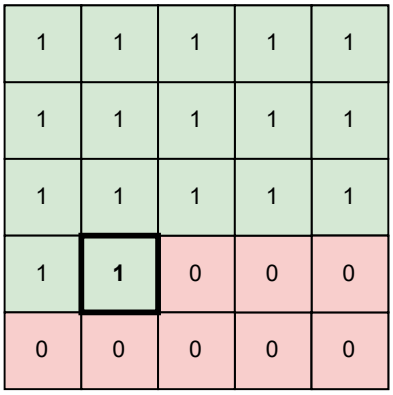

Figure 3.4: Example of a kernel for a 2D masked convolutional layer. The pixels to the bottom and right of the current pixel are not used. The current pixel is marked bold.

synthesis transform then predicts the standard deviations for each pixel in the latent of the hyperprior from that bottleneck. This improves the performance of the arithmetic coder significantly, because the probability distribution predicted by the hyperprior network is closer to the true distribution than the fully factorised distribution that was used in the model without a hyperprior network. This is especially true if the analysed dataset is heterogeneous, since the fully factorised distribution by definition is the same for each compressed image, while the inclusion of a hyperprior enables the arithmetic coder to use different standard deviations for the distributions of each image that depend on that specific image [21]. A tradeoff exists in that in contrast to the non-hyperprior approach, the bottleneck of the hyperprior CNN has to be transmitted alongside the image data. However, Ballé et.al. [21] showed that the advantage of increasing the performance of the main arithmetic coder outweighs the additional bits of information that have to be transmitted.

A further improvement to the hyperprior model was proposed in Minnen et.al. [22], seen in Figure 3.5. They firstly added a context model to the main arithmetic encoder. This context model adjusts the probability distributions for a specific image during the decoding process based on the already decoded elements. This is a common approach in arithmetic coding [50]. See also Section 3.3 for more elaboration on this idea. While the arithmetic coder in the previous models was based on the CABAC algorithm and therefore also context-sensitive, the context model did not improve the model significantly because images that were to be compressed were fed into the autoencoder sequentially pixel by pixel, which is not optimal [20]. Minnen et.al. improves the context model by introducing a masked 2D convolutional layer applied to to the quantized latents of the main CNN. A masked convolutional layer in this case refers to a convoltional layer

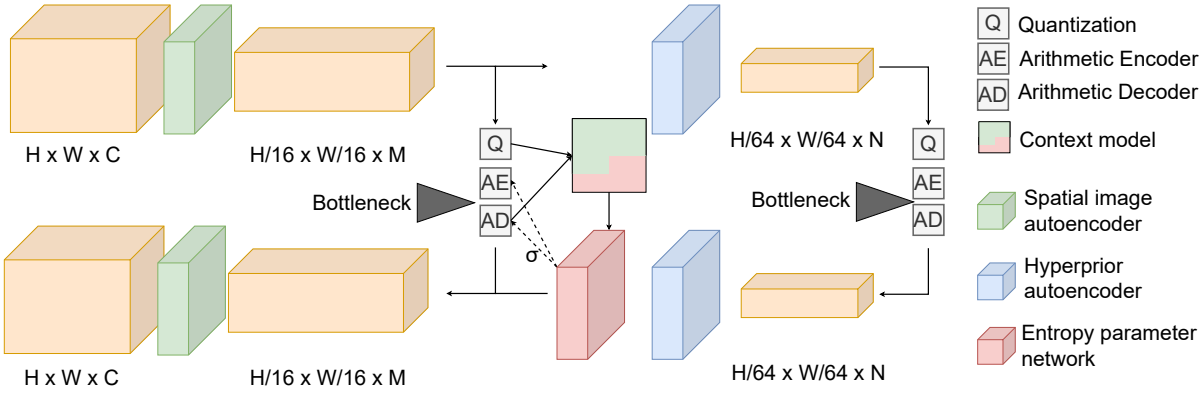

Figure 3.5: Hyperprior model using an autoregressive context model proposed in [22]

in which the kernel only contains pixels up and to the left of the currently transformed pixel. This idea, first proposed in Oord et.al. [52], ensures the layer only uses the parts of the input that are at this point known to both the arithmetic encoder and decoder. If a standard convolutional layer would be used, the output of the arithmetic coder would not be decodeable, since it would be partly based on pixels that are not yet decoded. The output of this layer is then combined with the output of the hyperprior synthesis transform using a three-layer CNN with 1x1 kernels to produce the parameters used for the arithmetic coder. Since the inputs to the context model can also be seen as coming from the output of the arithmetic decoder because of the losslessness of the arithmetic coder, the context model can be seen as an autoregressive component improving the probability distribution estimates of the arithmetic coder [22]. The other improvement proposed in Minnen et.al. [22] is a change from a gaussian scale hyperprior to a gaussian mixture model (GMM). In contrast to the hyperprior in Ballé et.al. [21] that predicted only the standard deviations of the gaussians, the proposed model also predicts the means of the gaussians. This is achieved in the model architecture by making the number of filters in the last layer of the hyperprior synthesis transform and the number of filters in the masked convolutional layer of the context model double the number of filters in the output layer of the main decoder. In this way, for each element in the latent two values in the probability distribution are predicted, the mean and the standard deviation [22].

4 Methodology

As discussed in Chapter 2, learning-based compression of red, green and blue (RGB) imagery is a hot topic. Therefore, being able to use and further research methods used in that field for hyperspectral image compression would be useful. However, direct application of most learning-based RGB image compression methods on hyperspectral data is not possible. This is mainly for two reasons. Firstly, learning-based RGB image compression methods ignore dependencies between the three channels in the RGB images and focus solely on spatial dependencies. This is useful for RGB images because the spectral dependencies within the three channels are low, since the corresponding wavelengths are far apart. Additionally, having only three channels means that there is not much spectral information per pixel to be compressed, meaning that even if spectral dependencies existed, the compression gains that could be extracted would be small. This is different in hyperspectral image compression where there are large amounts of spectral dependencies and the potential compression gains are much larger because of the high number of spectral channels. It is therefore necessary to modify RGB models to incorporate spectral dependencies in order to adapt them to the task of hyperspectral image compression.

The second reason for the difficulty in applying RGB image compression methods to hyperspectral image is the strong increase in spectral dimension. Compared to an RGB image, a hyperspectral image of the same size with 200 channels has $200/3 \approx 67$ times more dimensions. For this reason RGB models are often too small to learn the compression of high-dimensional hyperspectral image data. For example, many convolutional neural network (CNN) models use an input layer transforming the input from the dimensions $H \times W \times C$, where H and W are the height and width and C is the number of channels to $H' \times W' \times N$, where H' and W' are often either the original height and width or half the original height and width and N is the number of filters used in the input layer. If H' and W' are equal to half the original height and width, this is achieved either using a pooling layer or a convolutional layer with a stride of two. In RGB models, 128 and 192 are common values for N. For RGB images, this means that the input layer increases the dimensionality drastically in order to learn multiple features of the image. However, in hyperspectral images, the same layer would result in a decrease of dimensionality leading to loss of information. A simple idea in order to address this

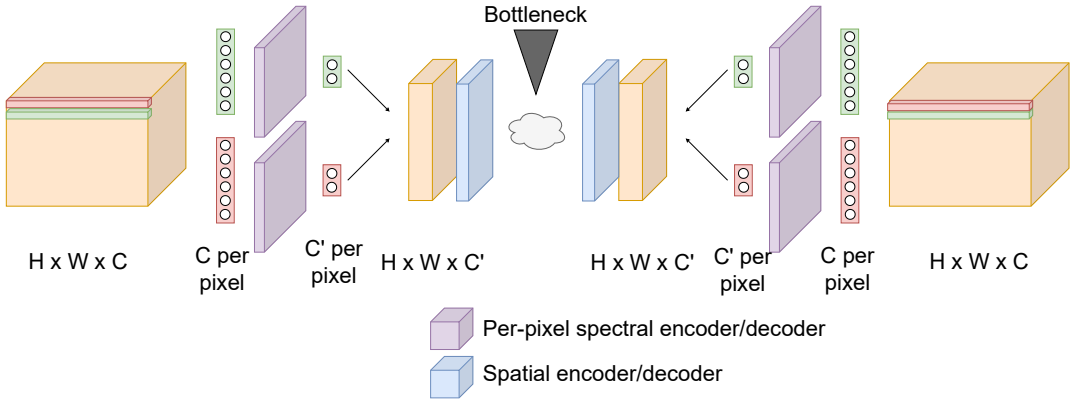

Figure 4.1: Architecture of the Combined Model using the per-pixel spectral CNN

issue is to increase the number of filters in each layer according to the increase in dimensionality between RGB and hyperspectral images. The input layer would then for example have $192 * 67 = 12864$ filters. However, this idea would result in a much larger model than the corresponding RGB compression model and therefore require a higher amount of computational resources for training, potentially making the model impossible to train on the available hardware. Additionally, even if the models were trainable, it is unclear if the convolutional layer would be able to learn 12864 distinct features for hyperspectral images.

Therefore, a different model architecture is needed. This thesis presents a novel architecture that allows the use of spatial compression methods designed for an almost arbitrary number of channels in hyperspectral image compression while also incorporating spectral dependencies. This not only allows the use of RGB image compression methods but also multispectral compression methods. This architecture is called ”Combined Model” and is shown in Figure 4.1.

4.1 The Combined Model

The main idea of the combined model is to use a spectral encoder network that preserves spatial dependencies in order to reduce the number of spectral channels in the input. Then, a spatial compression method is applied on the output of that spectral encoder in order to exploit the spatial dependencies. The resulting bottleneck is then decoded using the corresponding spatial decoder. Finally, a spectral decoder network symmetrical to the spectral encoder restores the original image. During training, the spectral decoder

learns to adapt to the distortion introduced by the spatial autoencoder. If the spatial autoencoder is learning-based, the spatial autoencoder similarly learns to produce output in such a way that the spectral autoencoder can better decode it. This mechanism is studied in detail in Chapter 5. Adaptation in either the spatial autoencoder or the spectral autoencoder are necessary, especially if the latent space of the spectral autoencoder is not continuous. In that case, a small change introduced by the spatial autoencoder could result in a large change in the output of the spectral decoder. In practice the adaptation process is successful in avoiding chaotic outputs of the spectral decoder, which is also shown in Chapter 5.

The aforementioned approach has multiple advantages. It is highly flexible with regard to the spatial autoencoder. Many different learning-based approaches can be used and the number of input channels to the spatial autoencoder can be modified so it best fits the spatial autoencoder. This can be achieved by modifying the compression ratio of the spectral encoder/decoder pair, which is possible for all spectral autoencoder models studied in this thesis. Another advantage is that the distinct separation between the spatial and the spectral encoding allows for informative studies to be done both regarding explainability and regarding the amount of spatial and spectral dependencies in the data. The model also allows for different spectral encoder and decoder architectures. There are two necessary conditions required for a spectral encoder/decoder pair to be possible for use in the combined model. Firstly, the output of the encoder must be an image-like shape in order for spatial image compression models to be used for incorporating the spatial dependencies. Secondly, the spectral encoder needs to preserve spatial dependencies, ideally resulting in an output with similar spatial dependencies to the actual input image. Because the spatial autoencoder models were designed for image compression, having spatial dependencies close to the dependencies found in images means that the task the spatial autoencoder is used for is close to the one it was designed for.

However, a difference between the task of image compression and the task the spatial autoencoder performs, that being compression of the latent of the spectral autoencoder, is still present. This is one of the disadvantages of the combined model approach and therefore requires the spatial autoencoder to be resilient to this change of task. However, for the spectral autoencoder models studied in this thesis, the spatial dependencies of the latents very closely resemble the spatial dependencies of the original input, mitigating this disadvantage. For further detail on this see Chapter 5.

For training the combined model, an additional idea is useful. If the model is trained in a standard way by endtraining both the spatial autoencoder and the spectral encoder/decoder pair end-to-end from a newly initialized state at the same time, the results can be unstable. Depending on the hyperparameters, it may not be possible to train the com-

bined model in this way. This follows from the fact that inputs to the spatial autoencoder depend on the outputs of the spectral encoder. During the start of training, the spectral encoder often outputs similar or equal values for all inputs. The spatial autoencoder may then specifically learn to compress latents with these specific values, which is a task very different from compressing spatial dependencies. In this case, the dependencies between the model parts can inhibit training and lead to the model being stuck in the local minimum of outputting the same output for any input. We solve this problem by pretraining the spectral encoder/decoder pair as an autoencoder without any spatial component. Then the combined model is trained using the pretrained spectral encoder and decoder. In this way, the spatial autoencoder receives useful input data from the beginning of training.

An interesting modification option for the model follows from pretraining the spectral encoder/decoder pair. After pretraining the spectral component, the combined model can be trained while freezing the spectral component, training only the spatial autoencoder. If trained in this way, the combined model can be seen as a spatial image compression model using an involved preprocessing and postprocessing step. This option is analysed and compared with the normal method of training the combined model in Chapter 5.

4.2 Spectral Autoencoder Methods

Three different architectures were studied for the spectral encoder and decoder models. Firstly, we analysed a per-pixel CNN using one-dimensional convolutional layers on the spectral domain of the hyperspectral images. We then developed another CNN-based model using two-dimensional convolutional layers to address some of the disadvantages of the one-dimensional CNN approach. Finally, we studied the possibility of directly using a variational autoencoder (VAE) for compression of the spectral information in hyperspectral images.

4.2.1 Per-Pixel Convolutional Neural Network

The first of the spectral autoencoder methods is based on the model proposed in Kuester et.al. [23, 24]. In their proposed method a hyperspectral input is split up into its pixels. The spectrum for each pixel then forms one datapoint on which the model is trained. The encoder of the model being trained is a one-dimensional (1D) CNN consisting of

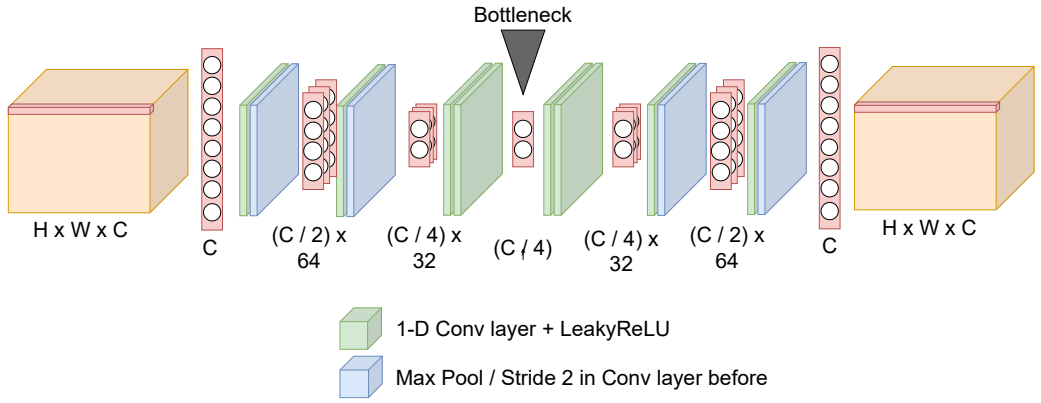

Figure 4.2: Architecture of the per-pixel spectral CNN using two downsampling layers resulting in a compression factor of four

two sets of a 1D convolutional layer with a large kernel size of 11 and a max pooling layer followed by two additional 1D convolutional layers with a kernel size of 9 and 7. The number of filters is also reduced by a factor of two in each convolutional layer and set to one in the last layer, resulting in a bottleneck with one fourth of the original spectral channels. The decoder is built analogously, only replacing the max pooling layers by upsampling layers. After each convolutional layer a leaky rectified linear unit (LeakyReLU) is used as the activation function. Additionally, the number of channels are padded with channels containing zero values in order for them to be divisible by the downsampling factor of 2. Since the number of channels is reduced by a factor of twice the downsampling factor during encoding, this can result in the bottleneck not having exactly one fourth of the channels of the input, if the number of channels in the padded input is not divisible by four. The decoder then results in one channel more than the original input. In this case, the last channel of the output is removed. A whole image can be compressed by encoding and decoding each pixel of the image separately, then reassembling the output image from the decoded pixels. The architecture of this model is shown in Figure 4.2.

This model is highly compatible with the combined model approach. Because the encoding of the spectral channels is performed per pixel, spatial dependencies are guaranteed to remain during encoding. The latent image is produced similarly to the output of the decoder by reassembling the latents for each pixel into a latent image, which can then be further encoded using a spatial autoencoder. In order to optimize the model for use in the combined model architecture and for the used dataset HySpecNet-11k [36], we adapted the model in various ways.

In the model as proposed by Kuester et.al. [23, 24], the number of layers and therefore

the compression ratio is fixed to four. However, different datasets can have different amounts of spatial and spectral dependencies. In the context of the combined model, this can result in the optimal model requiring a different spectral compression factor than 4. To allow for different compression ratio, the number of sets of a convolutional and a max pooling layer was made variable. In addition to this, the number of filters in each convolutional layer was adapted to the number of channels in the data. In the original paper, the first layer has a number of filters equal to half the number of padded channels of the data. We adapted the number of filters to the higher number of channels in the HySpecNet-11k dataset by keeping this ratio equal. This has the advantage that the number of filters is always appropriately divisible regardless of the number of layers ensured by the padding on the spectral channels of the input. We also changed the padding strategy. Instead of padding to a number of spectral channels divisible by half the compression ratio and then conditionally removing some number of channels at the output, we directly padded to a number of spectral channels divisible by the compression ratio. This removes the need for removing channels at the output, simplifying the model without loss of performance.

This spectral compression model is not only interesting for its applicability to the combined model. It is also the hyperspectral compression model with the highest reconstruction accuracy for multiple bitrates among many compared models. For more on this see Chapter 5. However, there are also some disadvantages of the model architecture. The reduction of spectral channels using max pooling layers results in divisions by two in the number of channels. This means that the compression ratio can only be a power of two. Additionally, the higher the chosen compression ratio is, the higher the number of needed padding channels is. As an example, for the HySpecNet-11k dataset with 202 channels two padding channels are needed for the compression ratio 4, for a compression ratio of 32 14 padding channels would be needed. Finally, the training process of this model is much slower than for which take complete images as input. Since each pixel is compressed separately, the number of optimization steps per epoch is much higher. This is also further discussed in Chapter 5.

4.2.2 Two-dimensional Spectral CNN

To address the issues of the 1D spectral CNN, we developed another model. This model uses two-dimensional (2D) convolutional layers with a kernel size of one to compress a whole image at once. For convolutional layers with a kernel size of one the convolution operation collapses down to a scalar multiplication. Additionally, the output has the same height and width as the input, as each pixel in the output is calculated only from the channels of a corresponding input pixel. The output for a specific pixel and filter

is therefore equivalent to a weighted sum over all channels in the input for this pixel. Importantly however, the weight used for a specific input channel is the same for each pixel in the image. A two-dimensional convolutional layer with a kernel size of one can therefore only encode spectral information when applied to a hyperspectral image, making it usable as part of the combined model. Such layers are commonly used for decreasing the number of filters between other convolutional layers TODO REF. Because of this, they are also referred to as channel or feature map pooling layers. They are then able to learn non-trivial summarizations of the preceding layer because of the nonlinearity introduced by the used activation function, in our case LeakyReLU. However, they can also instead be used to the opposite effect of increasing the number of filters while retaining the semantic information within these filters. An example of both of these uses is present in the "Inception" architecture proposed in Szegedy et.al. [53].

We show that it is also possible to use such layers to learn spectral dependencies in hyperspectral image data. The model is comprised of two-dimensional convolutional layers starting with a number of filters slighly higher than the number of channels in the hyperspectral dataset the model is trained on, 256. In each layer, the number of filters is doubled in order to increase the amount of information available. Finally, a last convolutional layer is used to reduce the number of channels to result in the desired bottleneck size. All convolutional layers use a kernel size of one and the LeakyReLU activation function.

This architecture has multiple advantages compared with the one-dimensional spectral CNN. Instead of being applied per pixel, it can be applied to the entire image at the same time. It also requires less parameters, as each layer has a number of parameters equal to $N_p = C * F$, where C is the number of channels of the input to the layer and F is the number of filters in the layer. Both of these factors result in a much shorter training time as seen in Chapter 5. The model is also more flexible with regard to the compression ratio. In contrast to the one-dimensional spectral encoder, where the compression ratios are necessarily given by $R \approx 2^N$, where N is the number of pooling layers, the two-dimensional spectral encoder allows setting of the number of output channels to an arbitrary number. This also removes the need for padding, which is an advantage especially for high compression ratios which would otherwise require much padding. Furthermore, the one-dimensional spectral CNN gets slower for higher compression ratios as it requires more layers while the number of layers and therefore the speed of the two-dimensional spectral CNN is constant with respect to the compression ratio.

Two modifications were also tested for this model. First, a Layer Normalization was performed after each convolutional layer in order to improve generalization performance and further reduce training time. We also modified the model by changing the kernel

sizes of the convolutional layers. With higher kernel sizes, the model becomes able to incorporate some amount of local spatial dependencies in addition to the spectral dependencies. We used this to study how sensitive the combined model is to spatial dependencies being partially encoded in the spectral encoder/decoder.

4.2.3 Variational autoencoder

As mentioned in Section 4.1, a requirement for spectral encoder / decoder pair is that the decoder is able to decode the latent of the encoder after some distortion to it has been applied by the spatial autoencoder. Therefore, using a VAE for the spectral compression is interesting, because VAEs produce a continuous latent space as shown by TODO REF. A VAE learns a probability distribution for a latent space and produces reconstructions by generating a samples from this distribution. We designed a one-dimensional VAE with the purpose of learning to autoencode spectral dependencies for single pixels, similar to the per-pixel CNN. This per-pixel VAE in fact uses the per-pixel CNN to produce a latent for a given input pixel. Two independent linear layers are then applied to this latent to produce a mean and a variance tensor respectively. The dimensionality of these vectors is variable and therefore a hyperparameter of the model. The linear layer producing the mean uses a tanh activation function to constrain the means between -1 and 1. The linear layer producing the variance also uses tanh but follows by an application of the softplus function to produce a positive variance. A new latent is then sampled from this distribution to produce the input for the decoder which is a symmetric inverted version of the encoder with only one linear layer instead of two, since only the new latent needs to be transformed to the latent space of the per-pixel CNN.

A basic VAE has not yet been used for hyperspectral image compression. Especially compression of spectral signatures using a one-dimensional VAE has not been studied. The only VAEs architecture commonly used for image compression is the hyperprior architecture, which is described in Chapter 4. However, these models are not applicable to be a spectral autoencoder for the combined model for two reasons. First, hyperprior models so far have been used for spatial dependencies in images, being applied to whole images at once and not one-dimensionally to spectral signatures. Secondly, since hyperprior models use an arithmetic coder in the bottleneck, they can only be used as the last stage of a compression model since the output of the arithmetic coder cannot be further compressed using another model. This model attempts to overcome these limitations to produce a continuous latent space that can be further compressed using a spatial autoencoder method while being permissible with regard to distortions being introduced during this process.

4.3 Spatial Autoencoder Methods

The second part of the combined model is the spatial autoencoder that further compresses the bottlneck of the spectral encoder. While there are some preconditions required for the spectral encoder/decoder pair, the choice of the spatial autoencoder is highly flexible. Since the latents of the spectral encoder have similar spatial dependencies to actual images, any autoencoder that is able to compress images can be used for this part of the model. The models analysed in this thesis are a CNN-based autoencoder, a hyperprior model and an optimized hyperprior model using self-attention modules and a context model. We also analysed the use of Joint Photographic Experts Group (JPEG) 2000 as the spatial autoencoder to study the viability of using a spatial autoencoder as part of the combined model that is not learning-based.

4.3.1 CNN-based Spatial Autoencoder

The CNN-based spatial autoencoder model uses a low-complexity approach using only two-dimensional convolutional layers. A similar architecture was used for hyperspectral image compression by La Grassa et.al. [25]. However, there are multiple differences between the model. La Grassa et.al. use a linear layer as the final layer of their network. This was not done for the CNN-based spatial autoencoder as it would increase the number of parameters of the model and therefore increase both training time and the required computational resources. The model proposed in this thesis also adds regularization into the model using layer normalization [54]. The number of filters in each layer is also different. Further, La Grassa et.al. uses max pooling layers to reduce the spatial dimensionality of the latent while we use convolutional layers with a stride. Additionally, we experiment with different kernel sizes to vary the amount of spatial dependencies introduced to the autoencoder.

The CNN-based spatial autoencoder model first uses two-dimensional convolutional layers to increase the dimensionality of an input image. This is done by using number of filters much higher than the number of channels in the input. This input layer is followed by another convolutional layer with the same number of filters to increase the depth of the model. Following that is a convolutional layer with a stride of two to reduce the spatial dimensions. This layer also increases the number of filters in order to reduce the information loss while downsampling. Finally, two further convolutional layers reduce the filter dimension to be identical to the number of channels in the input. In this way the dimensionality of the input is reduced by a factor of two in each spatial dimension, resulting in an overall compression factor of four. Additional layers with a stride of

two can be added to increase the compression factor. The decoder is symmetrical to the encoder but replaces convolutional layers with transposed convolutional layers. The model uses the Parameterised Rectified Linear Unit (PReLU) activation function after each convolutional layer and the sigmoid activation function after the final layer of the encoder and the decoder to keep outputs in the interval $[0, 1]$. Layer normalization [54] is also performed after each convolutional layer to increase robustness and rate of convergence during training. Layer normalization was used instead of the more common batch normalization because it is able to perform well even with a low batch size and the combined model with the per-pixel CNN was trained using a batch size of one.

This model was deliberately designed to be low-complexity to reduce the confounding factors for studying properties of the combined model. As there is no prior research for combining models in the way it was performed in this thesis for hyperspectral or RGB image compression, the properties of the combined architecture as a whole was a focus of the experiments. A low-complexity spatial model is therefore a useful baseline that allows for better attribution of specific experimental results to the combined architecture.

4.3.2 Hyperprior-based Spatial Autoencoder

To also include a more complex model with high relevancy in RGB compression research, we implemented the scale hyperprior model proposed by Ballé et.al. [20]. Since this model was explored in detail in Chapter 4, we will not go into detail here. The only necessary adaptation to the model in order to use it as part of the combined model was a change of the input and output layer to accomodate the chosen number of channels in the latent of the spectral encoder. This model was not studied intensively but rather used as a baseline. The main hyperprior-based architecture used was the following model using self-attention modules and a context model.

4.3.3 Attention-based Model Using Hyperprior Architecture

The main hyperprior-based model that was studied is based on work by Cheng et.al. [42]. They further improve the joint autoregressive and hierarchical hyperprior model by Minnen et.al. [22] discussed in detail in Chapter 4 by introducing self-attention modules. The attention modules are based on work by Liu et.al. [44]. However, the non-local blocks have been removed in order to simplify the attention modules. This allowed them to reduce the complexity of the model while still being able to adequately

capture long-range dependencies in the data. These self-attention modules were inserted into the main encoder and decoder in two places as shown in Fig. TODO.

We adapted this model in multiple ways for use in the combined model. First, the model was adapted to be able to receive different numbers of input channels by changing the input and output layers. The second modification regards the strength of the spatial downsampling in the convolutional parts of the hyperprior model. In the original model the main encoder reduces the spatial dimensionality by a factor of 8 in each direction, 64 in total, to produce the latent that is input to both the arithmetic coder and the hyperprior encoder. The hyperprior encoder reduces the spatial dimensions by a factor of 4 in each direction, 16 total, again. Since they used a dataset with 256x256 pixels, this results in a bottleneck of size 32x32 after the main encoder and 8x8 after the hyperprior encoder. However, for hyperspectral data in general and the HySpecNet-11k dataset especially a reduction in spatial dimension with a factor of 64 can result in large distortion, since there are less spatial dependencies than in typical RGB datasets. We therefore modified both the numbers of convolutional layers and the strides in the main encoder/decoder pair and the hyperprior encoder/decoder pair in order to allow for different options for the spatial dimensionality reduction factors. For this, we group models by the spatial reduction factors in the main encoder/decoder pair and the bottleneck size of the hyperprior encoder. The bottleneck size of the hyperprior being larger results in more information remaining present in the hyperprior which increases performance. However, more side information needs to be transmitted, which can decrease performance. This is because the loss used for training these models, Rate Distortion (RD) loss, tries to achieve a balance between bitrate and distortion. Since a larger amount of transmitted side information raises bitrate, RD loss with the same parameters will also result in higher distortion. Therefore, an optimum needs to be found depending on the amount of spatial dependencies present in the dataset.

5 Experiments

5.1 Description of the Data Set

The dataset used in this thesis is the HySpecNet-11k dataset developed by Fuchs et.al. [36]. HySpecNet-11k is a hyperspectral benchmark dataset that consists of 11,483 image patches containing large-scale views of the surface of the earth. The image patches do not overlap which is advantageous for use in learning-based approaches as there are no duplicated image parts in the training set. The source data for the dataset uses 224 spectral bands and 128×128 pixels spatially with a ground sampling distance (GSD) of 30 meters. However, we use the preprocessed version of the dataset which removes 20 spectral bands due to high water vapor absorption in those bands. We therefore only use 202 spectral bands. The image patches are of high quality because the tiles they were obtained from contain less than 10% snow and cloud cover and were radiometrically, geometrically and atmospherically corrected [36]. Furthermore, cropped patches at the borders of tiles were discarded, so each image patch contains information in all 128×128 pixels. For splitting the dataset into training, validation and test set we used the provided patch-wise splitting set. This means that patches from the same tile can be present in different sets, unlike in the tile-wise splitting set. We did not perform experiments with the tile-wise splitting set as it is outside of the scope of this thesis. It is however an interesting research direction for the future.

5.2 Loss Functions and Metrics

5.2.1 MSE Loss

The Mean Squared Error (MSE) function is a common loss function used for training artifical neural networks (ANNs). For the task of compressing a hyperspectral image it

can be defined as follows:

$$\text{MSE}(X,\hat{X}) = \frac{1}{B*C*H*W}\sum_{b=1}^{B}\sum_{c=1}^{C}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{bchw}-\hat{x}_{bchw})^2$$

Here X is defined as a batch of hyperspectral input images, $\hat{X}$ the reconstruction of the images in X obtained from the compression model, B the number of images per batch and C,H and W the number of spectral channels, height and width of the hyperspectral images respectively. We use MSE loss for pre-training the spectral autoencoder models as well as for training the convolutional neural network (CNN)-based spatial autoencoder outside of the combined model.

5.2.2 Rate Distortion Loss

In contrast to the purely CNN-based models with a fixed compression rate for a given set of hyperparameters, the hyperprior-based models have a variable compression rate that changes during the training process. This comes from the fact that the bitrate of the arithmetic coder depends on the performance of the hyperprior CNN. Because of this MSE loss is not optimal for these models. While it is possible to train hyperprior-based models using MSE loss because the bitrate dependency is contained in the lossless arithmetic coder, this leads to very high bitrates. However, using MSE loss for a hyperprior model can still be useful to obtain an upper bound on the reconstruction accuracy for a given set of hyperparameters. For more elaboration on this see TODO.

To train a model using an arithmetic coder in the bottleneck with both a low distortion and a low bitrate, a different loss function is needed. The type of problem that arises is a rate-distortion optimization problem [21]. This can problem can parametrized in the following way [22]:

$$R+\lambda\cdot D = \underbrace{\mathbf{E}_{x\sim p_x}[-\log_2 p_{\hat{y}}(\lfloor f(x)\rceil)]}_{\text{rate}} + \underbrace{\lambda\cdot\mathbf{E}_{x\sim p_x}[d(x,g(\lfloor f(x)\rceil))]}_{\text{distortion}}.$$

We parametrize the rate-distortion trade-off with the Lagrange multiplier λ. p_x is the unknown distribution of input images, $\lfloor\cdot\rceil$ refers to quantization and f is the main encoder. Accordingly, $y = f(x)$ is the output of the main encoder applied to the input x, $\hat{y} = \lfloor y\rceil$ are the quantized latents of the main encoder, $p_{\hat{y}}$ is a discrete entropy model and $\hat{x} = g(\hat{y})$ is the reconstruction of x obtained from applying the main decoder g to the quantized latents $\hat{y}$. As the metric for computing the distortion we use MSE, because Ballé et.al. showed that the rate-distortion problem is equivalent to minimizing

the Kullback-Leibler (KL) divergence over p_x [21]. Using this equivalence they showed that using MSE as distortion metric is equivalent to assuming a Gaussian distribution for $p_{x|\hat{y}}(x|\hat{y})$. While there are approaches assuming different probability distributions like the Student's T distribution, this was not a focus of this thesis and the Gaussian often provides a good estimate for unknown probability distributions. For this reason we also assumed a Gaussian distribution for the probability distribution.

However, this formulation does not include the hyperprior network that learns and transmits side information about the entropy model $p_{\hat{y}}$. In order to incorporate the hyperprior into the Rate Distortion (RD) loss function, we need to add an additional term to capture the bitrate for the transmitted side information. The final loss function is then defined as follows:

$$\begin{aligned} R + \lambda \cdot D &= \underbrace{\mathbf{E}_{x\sim p_x}[-\log_2 p_{\hat{y}}(\lfloor f(x)\rceil)]}_{\text{rate (main latents)}} + \underbrace{\mathbf{E}_{x\sim p_x}[-\log_2 p_{\hat{z}}(\lfloor f_h(f(x))\rceil)]}_{\text{rate (hyper-latents)}} \\ &\quad + \underbrace{\lambda \cdot \mathbf{E}_{x\sim p_x}[\|x - g(\lfloor f(x)\rceil))\|_2^2]}_{\text{distortion}}. \\ &= \underbrace{\mathbf{E}_{x\sim p_x}[-\log_2 p_{\hat{y}}(\hat{y})]}_{\text{rate (main latents)}} + \underbrace{\mathbf{E}_{x\sim p_x}[-\log_2 p_{\hat{z}}(\hat{z})]}_{\text{rate (hyper-latents)}} + \underbrace{\lambda \cdot \mathbf{E}_{x\sim p_x}[\|x - \hat{x}\|_2^2]}_{\text{distortion}} \end{aligned}$$

Here z and $\hat{z}$ refer to the latent of the hyperprior and the quantized latent of the hyperprior respectively, that being the result of applying both the main encoder f and the hyperprior encoder f_h to the input x.

With regard to the quantization operation, care has to be taken. When executing the trained hyperprior model, quantization is defined as rounding to the nearest integer. However, during training this would result in zero gradients everywhere. To avoid this, while training we instead apply a uniform noise to the input in place of rounding. From this follows that the arithmetic coder is only executed during testing, not during training. In training the achieved bitrate is estimated using a heuristic. However, we verified that the bitrate estimates are close to the real achieved bitrate in Section 5.7.

5.2.3 Dual MSE Loss

The combined model consists of an outer autoencoder model and an inner autoencoder model. However, applying a loss function such as MSE loss in the standard way results in the inner autoencoder being trained through a layer of distortion at both the input and the output, as the input to the loss function are the input to outer encoder x and the

output of the outer decoder $\hat{x}$. The input to the inner encoder, $y = f_o(x)$, where f_o is the outer encoder and the output of the inner decoder, $\hat{y} = g_i(f_i(y))$, where f_i and g_i are the inner encoder and decoder respectively, are not direct inputs to the loss functions. To alleviate that, we designed an experimental loss function called Dual Mean Squared Error (DualMSE) loss. It is defined as follows:

$$\text{DualMSE}(x,\hat{x},y,\hat{y}) = \lambda * \text{MSE}(x,\hat{x}) + (1-\lambda) * \text{MSE}(y,\hat{y}).$$

The DualMSE loss is therefore a linear interpolation between the MSE loss of the complete model and the MSE loss of the inner model. This allows for fine-grained control over the weight of the loss of the inner model in the overall loss. Of note are also the two edge cases of $\lambda = 1$, where the DualMSE loss function becomes equivalent to MSE loss and $\lambda = 0$, where only the loss of the inner model is used. The DualMSE loss was used for the combined models using the CNN-based spatial autoencoder. Results regarding the use of DualMSE loss for these models can be seen in Section 5.4. While the use of a Dual Rate Distortion loss would be possible for the combined models using a hyperprior-based spatial autoencoder, this was not done because the results in Section 5.4 did not show a clear and significant improvement when using a dual loss function.

5.2.4 Metrics

We use two different metrics to evaluate our experiments. The first metric is Peak signal-to-noise ratio (PSNR), which is a common metric to quantify reconstruction accuracy for lossy learning-based image compression [20–25]. PSNR is defined in general as follows:

$$\text{PSNR}(x,\hat{x}) = 10 * \log_{10} \frac{\text{MAX}}{\text{MSE}(x,\hat{x})} \quad = 20 * \log_{10}(\text{MAX}) - 10 * \log_{10}(\text{MSE}(x,\hat{x})).$$

Here MAX is the maximum possible pixel value in the used dataset and $\text{MSE}(x,\hat{x})$ refers to the Mean Squared Error between x and $\hat{x}$. However, since the data range for the HySpecNet-11k dataset used in this thesis is the intervall $[0,1]$, the equation simplifies to:

$$\text{PSNR}(x,\hat{x}) = -10 * \log_{10}(\text{MSE}(x,\hat{x})).$$

It is therefore directly correlated with the log of the MSE. While PSNR is a commonly used metric in lossy learning-based hyperspectral compression, it is computed as an average over all channels. We therefore want to employ another metric to more accurately evaluate the reconstruction of the spectral signatures which are important for lossy reconstructions of hyperspectral images.

For this we use the spectral angle. It is computed by calculating the angle between the spectra of the input image and the reconstruction for each pixel using the Spectral Angle Mapper (SAM) algorithm [55]. The computed angles are then averaged to compute the metric for a given input image and reconstruction. Using the combination of PSNR and spectral angle we can compare different models with regard to spatial and spectral reconstruction accuracy.

5.3 Impact of Hughes Phenomenon

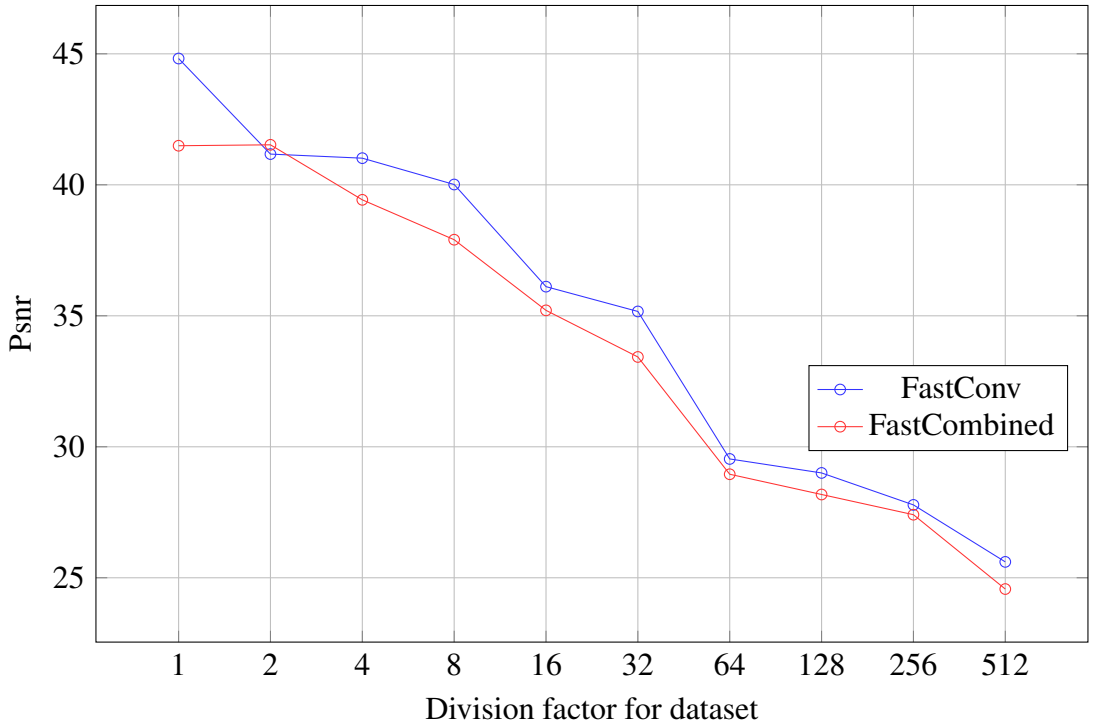

Figure 5.1: Hughes Phenomenon combined vs fast combined

5.4 Dual MSE Loss Results And Latent Space Analysis

For this experiment shown in Figure 5.3 we tested multiple different values for λ in the DualMSE loss. The spectral angle values for this experiment are shown in Figure 1

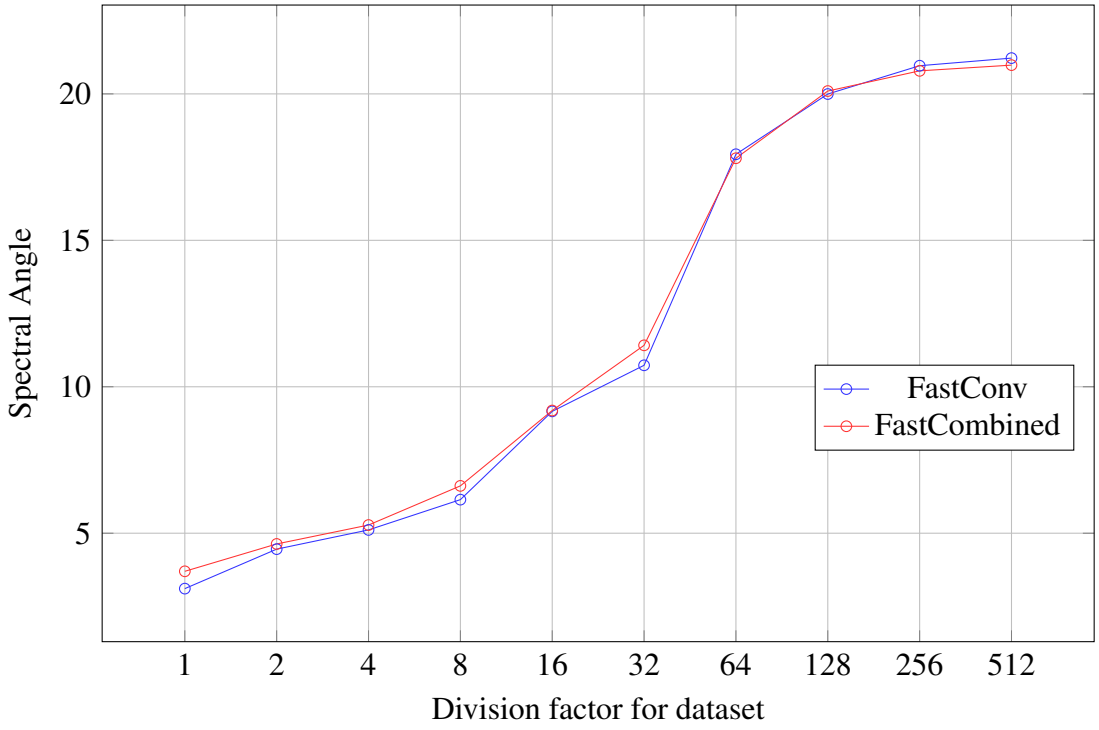

Figure 5.2: Hughes Phenomen combined vs fast combined

in the appendix, as it shows similar results to Figure 5.3. We did this for multiple variations of the Combined Model. All models in this experiments were trained using the Adam optimizer with the learning rate set to $0.5 * 10^{-4}$. All models were trained by first pretraining the spectral autoencoder for 70 epochs and then training the complete Combined Model for 50 epochs. Furthermore all models had a spectral downsampling factor of 16 and a spatial downsampling factor of 4, resulting in a total compression ratio of 64 or a bitrate of 0.5 bits per pixel per channel (bpppc).

The tested models differ in two ways. Firstly, some models use the per-pixel spectral autoencoder for the spectral part of the Combined Model while some use the two-dimensional (2D) CNN-based spectral autoencoder. We refer to the first kind of model simply as ”Combined” and to the second model as ”FastCombined” stemming from the fact that the models using this spectral encoder have a higher throughput as explained in Subsection 4.2.2. The other variable in which the models differ is whether the weights of the spectral autoencoder are frozen after pretraining. We tested freezing all weights after pretraining, making the complete model equivalent to training the spatial autoencoder while using the pretrained spectral encoder/decoder pair for pre- and postprocessing respectively. We also tested only freezing the weights of the spectral encoder after pretraining. In this way, the latent space produced by the spectral encoder does not change

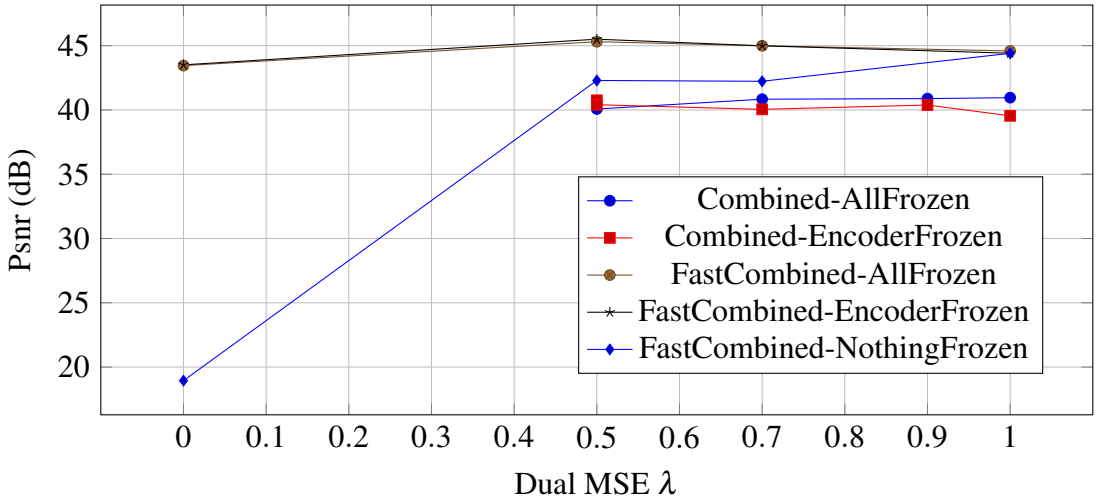

Figure 5.3: Dual MSE lambda plot

during training which we hoped could improve the training of the spatial autoencoder. Additionally, in contrast to the models where all weights of the spectral autoencoder are frozen after pretraining, the spectral decoder can still learn to adapt to the distortion introduced by the spatial autoencoder.

However, with the exception of $\lambda = 0$ for the model where only the encoder is frozen after pretraining, no clear trends in the PSNR or spectral angle could be observed regarding the value of λ. However, regarding the decision on whether to freeze parts of the spectral autoencoder after pretraining it is valuable to freeze either the spectral encoder or the complete spectral autoencoder, since the PSNR of the model FastCombined-NothingFrozen is $1.5 - 2$ dB below the other FastCombined models for $\lambda = 0.5$ and $\lambda = 0.7$. For $\lambda = 0$ FastCombined-NothingFrozen is much worse with a difference of over 20 dB. To analyse why the model FastCombined-NothingFrozen with $\lambda = 0$ performed much worse than all other models, we computed the minimum and maximum values for the latent vectors y of the spectral encoder and for the reconstructed latents $\hat{y}$ produced by the spatial autoencoder. For the model FastCombined-EncoderFrozen and FastCombined-AllFrozen these values behaved as expected with the maxima close to 1 and the minima close to 0. This is expected because the last layer of the spectral encoder is a sigmoid layer which constrains the latents to be in the interval $[0, 1]$. However, for the model FastCombined-NothingFrozen both the minimum and the maximum approached 0.5. On the test set the minimum latent value was 0.4875 and the maximum 0.5118. The spatial autoencoder then reconstructed a constant output close to 0.5 for each channel. Because for $\lambda = 0$ only the MSE between y and the $\hat{y}$ is relevant, the model managed to minimize this loss by learning to reduce the distance of the inputs

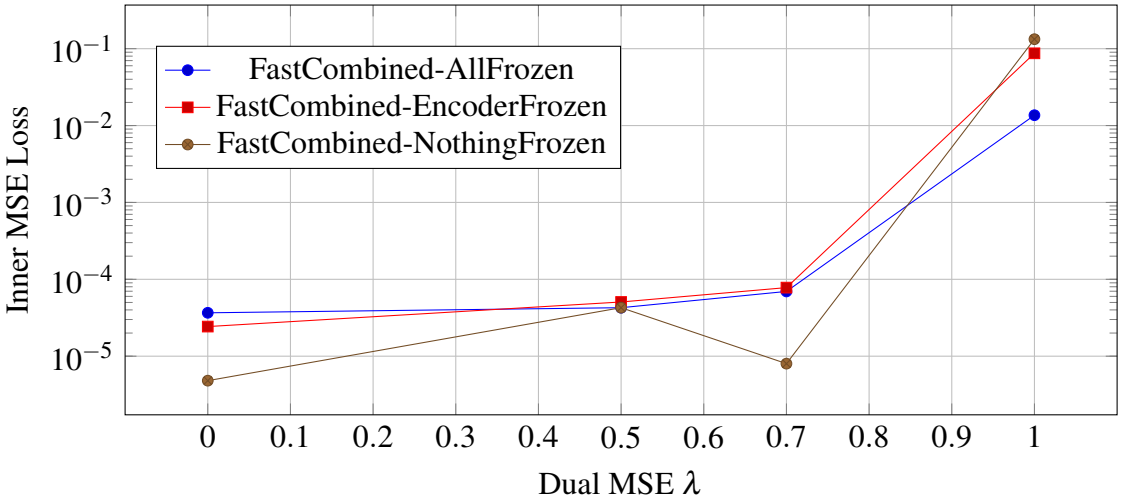

Figure 5.4: Inner mse plot

between the latents.

However, while the usage of the DualMSE loss does not have a strong effect on PSNR or spectral angle with our models on the HySpecNet-11k dataset, it does succeed in strongly reducing the differences between the latent produced by the spectral encoder and its reconstruction. As shown in Figure 5.4, using DualMSE loss with $\lambda < 1$ drastically reduces the inner MSE, which is the MSE between y and $\hat{y}$. For FastCombined-NothingFrozen the inner MSE reduces from 0.133 to $8.00 * 10^{-6}$, which is a factor of $\approx$ 16606. For FastCombined-EncoderFrozen this factor is $\approx$ 1116 and for FastCombined-AllFrozen the reduction factor is $\approx$ 196, which are still large decreases in loss. This can also be seen visually when comparing the reconstructed latents as we have done in Figure 5.5. We show the effect for FastCombined-AllFrozen since the frozen encoder gives the advantage of producing the same latent representations for all compared models, as the weights of the encoder are constant during training. It can be seen that with $\lambda = 1$ the latents have a visible grid pattern that is not present when using $\lambda < 1$. A similar pattern can also be seen during the first epoch of training for models with $\lambda < 1$ and is therefore likely a result of the convolutional layers used in the 2D CNN-based spatial autoencoder. During training these patterns disappear when the MSE between y and $\hat{y}$ is part of the loss function, which is not the case for $\lambda = 1$. However, the models with $\lambda = 1$ produce similar overall reconstructions $\hat{x}$ as shown by the similar PSNR and spectral angle. The spectral decoder therefore is able to ignore these grid patterns even without additional training, since even the models FastCombined-AllFrozen and Combined-AllFrozen with $\lambda = 1$ have a high PSNR and a low spectral angle. This shows a high robustness of the spectral autoencoders with regard to distortions intro-

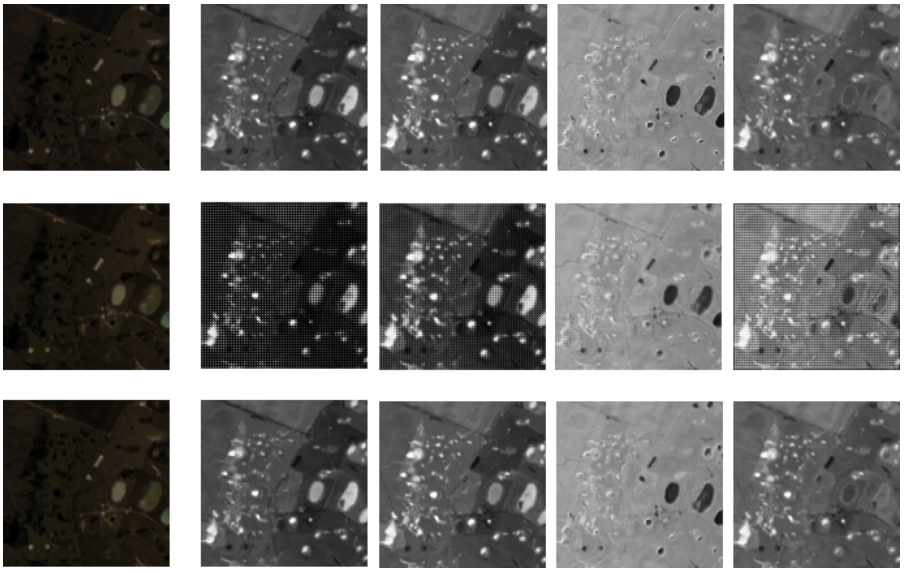

Figure 5.5: Latent image comparison

duced to the bottleneck. It also demonstrates the high degree of explainability provided by the Combined Model architecture. The latent representations seen in Figure 5.5 also confirm the thesis that the 2D spectral encoder does preserve spatial dependencies. It can also be seen by the inner latents z in Figure 5.6 that the 2D CNN-based spatial autoencoder partially exploits the spatial dependencies in the latents as the inner latents have less spatial structure than the outer latents y. Especially for the second channel shown the much of the spatial structure seems to be exploited, while for the first channel the effect is not as strong. However, it can also be seen that some spatial dependencies still remain in z. For this reason an improved spatial autoencoder that is able to fully exploit the spatial dependencies in y could potentially improve the model.

The best performing model for this experiment was FastCombined-EncoderFrozen with $\lambda = 0.5$ with a PSNR of 45.50 dB and a spectral angle of 3.06°. The second best model was FastCombined-AllFrozen with $\lambda = 0.7$ with a PSNR of 45.31 and a spectral angle of 3.22°.

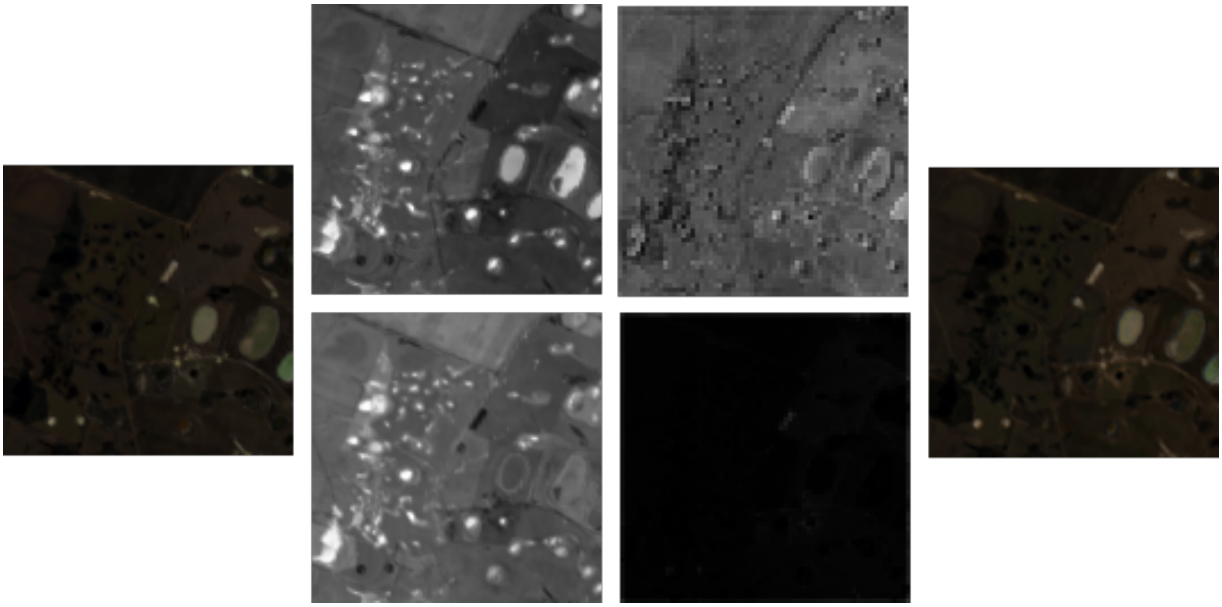

Figure 5.6: Inner latent comparison

Kernel Size	Per Channel?	PSNR (dB)	Spectral Angle (°)
3	No	**45.50**	**3.06**
5	No	43.72	3.30
7	No	42.59	3.58
3	Yes	42.33	3.30

Table 5.1: Comparison of different kernel sizes for the spatial autoencoder and per-channel spatial autoencoder with the baseline model. For all models the spectral encoder is frozen after pretraining and $\lambda = 0.5$

5.5 Kernel size change in 2D CNN-based spatial autoencoder

To improve the amount of spatial dependencies exploited by the 2D CNN-based spatial autoencoder as part of the Combined Model, we experimented with changing the kernel sizes of the convolutional layers of the spatial autoencoder from three to the higher values five and seven. These experiments were performed using the model FastCombined-EncoderFrozen with $\lambda = 0.5$. However, as seen in Table 5.1, this did not improve either PSNR or spectral angle. The highest metrics were achieved by the base model using a kernel size of three with 45.50 dB and a spectral angle of 3.06°. The model using a kernel size of five achieved a lower PSNR of 43.72 dB and a higher spectral angle of 3.30°, the model using kernel size of seven resulted in the lowest acpsnr of 42.59 dB and the lowest spectral angle with 3.58°. We therefore conclude that a low kernel size of three is optimal for the CNN-based spatial autoencoder.

5.6 Per-Channel 2D CNN-based spatial autoencoder

On the HySpecNet-11k dataset, the Joint Photographic Experts Group (JPEG) 2000 baseline performed surprisingly well. JPEG 2000 is executed per channel, therefore performing only spatial compression on 202 single-channel images. However, the 2D CNN-based spatial autoencoder uses the same filters to compress all channels in the bottleneck of the spectral autoencoder. We theorized that the differences between the latent channels might be too large to be appropriately captured using the same filters.

To test this thesis, we modified the 2D CNN-based spatial autoencoder to use a separate set of filters for each channel in the latent. Since this multiplies the number of parameters by the number of latent channels, we reduced the number of filters per channel. For the layers which originally had 256 filters, the modified model only uses 32 filters. For the layers that originally had 512 filters, the modified model uses 64 filters. The reduction in the number of filters is not only appropriate to reduce the number of parameters for the model, it follows also from the fact that less parameters are required for a convolutional layer that captures dependencies for a single channel than a convolutional layer that captures dependencies for multiple channel simultaneously. In total, the number of parameters in the spatial encoder after the reduction in the number of filters rises by 49.3 % from 50,849,818 to 75,928,346 parameters. We tested the modified model using the model FastCombined-EncoderFrozen with $\lambda = 0.5$. However, the modification did not

significantly change the performance of the Combined Model with regard to the metrics PSNR or the spectral angle as seen in Table 5.1.

5.7 Results of the bitrate heuristic for the hyperprior model

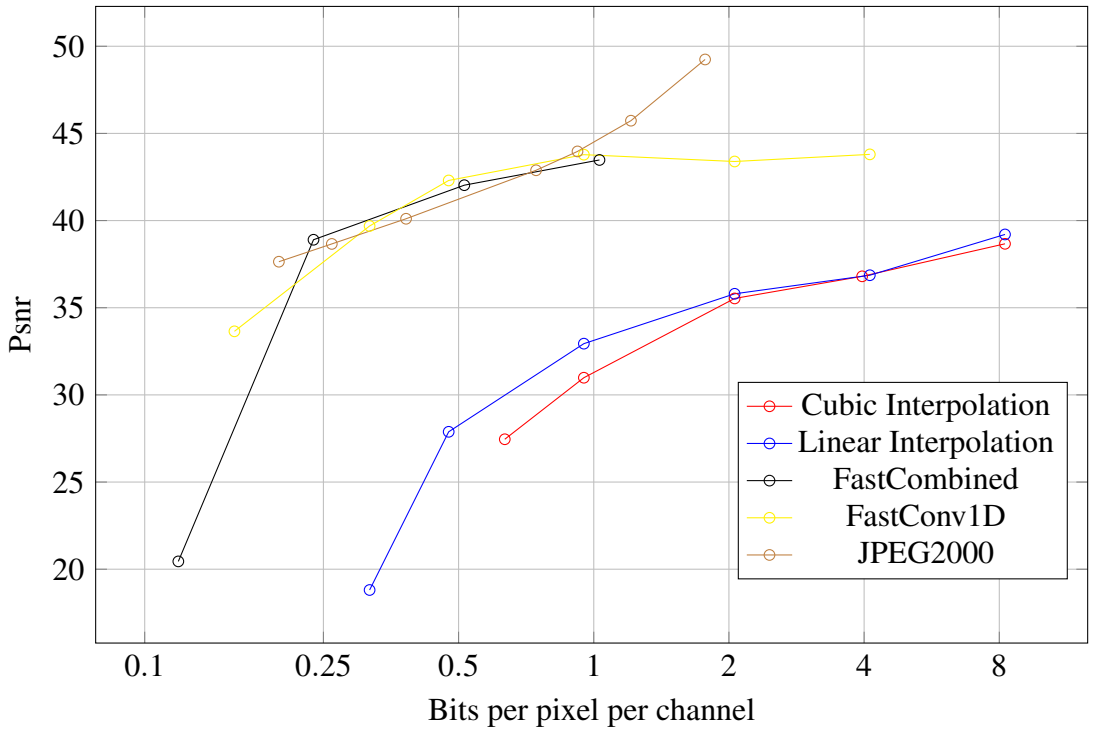

Figure 5.7: Rate distortion plot

6 Discussion and Conclusion

Bibliography

[1] M. Zea, A. Souza, Y. Yang, L. Lee, K. Nemali, and L. Hoagland, "Leveraging high-throughput hyperspectral imaging technology to detect cadmium stress in two leafy green crops and accelerate soil remediation efforts", *Environmental Pollution*, vol. 292, p. 118 405, Jan. 1, 2022, ISSN: 0269-7491. DOI: 10.1016/j.envpol.2021.118405.

[2] M. L. Henriksen, C. B. Karlsen, P. Klarskov, and M. Hinge, "Plastic classification via in-line hyperspectral camera analysis and unsupervised machine learning", *Vibrational Spectroscopy*, vol. 118, p. 103 329, Jan. 1, 2022, ISSN: 0924-2031. DOI: 10.1016/j.vibspec.2021.103329.

[3] L. Guanter, H. Kaufmann, K. Segl, S. Foerster, C. Rogass, S. Chabrillat, T. Kuester, A. Hollstein, G. Rossner, C. Chlebek, C. Straif, S. Fischer, S. Schrader, T. Storch, U. Heiden, A. Mueller, M. Bachmann, H. Mühle, R. Müller, M. Habermeyer, A. Ohndorf, J. Hill, H. Buddenbaum, P. Hostert, S. Van der Linden, P. J. Leitão, A. Rabe, R. Doerffer, H. Krasemann, H. Xi, W. Mauser, T. Hank, M. Locherer, M. Rast, K. Staenz, and B. Sang, "The EnMAP spaceborne imaging spectroscopy mission for earth observation", *Remote Sensing*, vol. 7, no. 7, pp. 8830–8857, Jul. 2015, Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2072-4292. DOI: 10.3390/rs70708830.

[4] N. Bohn, B. Di Mauro, R. Colombo, D. R. Thompson, J. Susiluoto, N. Carmon, M. J. Turmon, and L. Guanter, "Glacier ice surface properties in south-west greenland ice sheet: First estimates from PRISMA imaging spectroscopy data", *Journal of Geophysical Research: Biogeosciences*, vol. 127, no. 3, e2021JG006718, 2022, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2021JG006718, ISSN: 2169-8961. DOI: 10.1029/2021JG006718.

[5] A. B. Pascual-Venteo, E. Portalés, K. Berger, G. Tagliabue, J. L. Garcia, A. Pérez-Suay, J. P. Rivera-Caicedo, and J. Verrelst, "Prototyping crop traits retrieval models for CHIME: Dimensionality reduction strategies applied to PRISMA data", *Remote Sensing*, vol. 14, no. 10, p. 2448, Jan. 2022, Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2072-4292. DOI: 10.3390/rs14102448.

[6] V. Döpper, A. D. Rocha, K. Berger, T. Gränzig, J. Verrelst, B. Kleinschmit, and M. Förster, "Estimating soil moisture content under grassland with hyperspectral data using radiative transfer modelling and machine learning", *International Journal of Applied Earth Observation and Geoinformation*, vol. 110, p. 102 817, Jun. 1, 2022, ISSN: 1569-8432. DOI: 10.1016/j.jag.2022.102817.

[7] R. Loizzo, M. Daraio, R. Guarini, F. Longo, R. Lorusso, L. Dini, and E. Lopinto, "Prisma Mission Status and Perspective", in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, ISSN: 2153-7003, Jul. 2019, pp. 4503–4506. DOI: 10.1109/IGARSS.2019.8899272.

[8] R. Guarini, R. Loizzo, F. Longo, S. Mari, T. Scopa, and G. Varacalli, "Overview of the prisma space and ground segment and its hyperspectral products", in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, ISSN: 2153-7003, Jul. 2017, pp. 431–434. DOI: 10.1109/IGARSS.2017.8126986.

[9] R. Guarini, R. Loizzo, C. Facchinetti, F. Longo, B. Ponticelli, M. Faraci, M. Dami, M. Cosi, L. Amoruso, V. De Pasquale, N. Taggio, F. Santoro, P. Colandrea, E. Miotti, and W. Di Nicolantonio, "Prisma Hyperspectral Mission Products", in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, ISSN: 2153-7003, Jul. 2018, pp. 179–182. DOI: 10.1109/IGARSS.2018.8517785.

[10] N. Acito, M. Diani, and G. Corsini, "PRISMA Spatial Resolution Enhancement by Fusion With Sentinel-2 Data", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 62–79, 2022, Conference Name: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, ISSN: 2151-1535. DOI: 10.1109/JSTARS.2021.3132135.

[11] M. Drusch, U. Del Bello, S. Carlier, O. Colin, V. Fernandez, F. Gascon, B. Hoersch, C. Isola, P. Laberinti, P. Martimort, A. Meygret, F. Spoto, O. Sy, F. Marchese, and P. Bargellini, "Sentinel-2: ESA's optical high-resolution mission for GMES operational services", *Remote Sensing of Environment*, The Sentinel Missions - New Opportunities for Science, vol. 120, pp. 25–36, May 15, 2012, ISSN: 0034-4257. DOI: 10.1016/j.rse.2011.11.026.

[12] D. Spiller, S. Amici, and L. Ansalone, "Transfer Learning Analysis For Wildfire Segmentation Using Prisma Hyperspectral Imagery And Convolutional Neural Networks", in *2022 12th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, ISSN: 2158-6276, Sep. 2022, pp. 1–5. DOI: 10.1109/WHISPERS56178.2022.9955054.

[13] D. Spiller, K. Thangavel, S. T. Sasidharan, S. Amici, L. Ansalone, and R. Sabatini, "Wildfire segmentation analysis from edge computing for on-board real-time alerts using hyperspectral imagery", in *2022 IEEE International Conference on*

Metrology for Extended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE), Oct. 2022, pp. 725–730. DOI: 10.1109/MetroXRAINE54828.2022.9967553.

[14] C. E. Shannon, "A mathematical theory of communication", *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948, ISSN: 1538-7305. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[15] D. J. C. MacKay, *Information theory, inference, and learning algorithms*. Cambridge, UK ; New York: Cambridge University Press, 2003, 628 pp., ISBN: 978-0-521-64298-9.

[16] F. Garcia-Vilchez, J. Munoz-Mari, M. Zortea, I. Blanes, V. Gonzalez-Ruiz, G. Camps-Valls, A. Plaza, and J. Serra-Sagrista, "On the impact of lossy compression on hyperspectral image classification and unmixing", *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 2, pp. 253–257, Mar. 2011, ISSN: 1545-598X, 1558-0571. DOI: 10.1109/LGRS.2010.2062484.

[17] E. Delp and O. Mitchell, "Image Compression Using Block Truncation Coding", *IEEE Transactions on Communications*, vol. 27, no. 9, pp. 1335–1342, Sep. 1979, Conference Name: IEEE Transactions on Communications, ISSN: 1558-0857. DOI: 10.1109/TCOM.1979.1094560.

[18] M. M. Hannuksela, J. Lainema, and V. K. Malamal Vadakital, "The High Efficiency Image File Format Standard [Standards in a Nutshell]", *IEEE Signal Processing Magazine*, vol. 32, no. 4, pp. 150–156, Jul. 2015, Conference Name: IEEE Signal Processing Magazine, ISSN: 1558-0792. DOI: 10.1109/MSP.2015.2419292.

[19] S. Ruder, *An overview of gradient descent optimization algorithms*, Jun. 15, 2017. DOI: 10.48550/arXiv.1609.04747. arXiv: 1609.04747[cs].

[20] J. Ballé, V. Laparra, and E. P. Simoncelli, *End-to-end Optimized Image Compression*, Number: arXiv:1611.01704, Mar. 3, 2017. DOI: 10.48550/arXiv.1611.01704. arXiv: 1611.01704[cs,math].

[21] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, *Variational image compression with a scale hyperprior*, Number: arXiv:1802.01436, May 1, 2018. DOI: 10.48550/arXiv.1802.01436. arXiv: 1802.01436[cs,eess,math].

[22] D. Minnen, J. Ballé, and G. D. Toderici, "Joint Autoregressive and Hierarchical Priors for Learned Image Compression", in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018.

[23] J. 1. Kuester, W. 1. Gross, W. 1. 1. F. I. Middelmann, G. Image Exploitation, E. Fraunhofer IOSB, and G. Image Exploitation, "1d-Convolutional Autoencoder Based Hyperspectral Data Compression", ISSN: 16821750 Num Pages: 15-21, Copernicus GmbH, 2021, pp. 15–21. DOI: 10.5194/isprs-archives-XLIII-B1-2021-15-2021.

[24] J. Kuester, W. Gross, S. Schreiner, M. Heizmann, and W. Middelmann, "Transferability of Convolutional Autoencoder Model For Lossy Compression to Unknown Hyperspectral Prisma Data", in *2022 12th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, ISSN: 2158-6276, Sep. 2022, pp. 1–5. DOI: 10.1109/WHISPERS56178.2022.9955109.

[25] R. La Grassa, C. Re, G. Cremonese, and I. Gallo, "Hyperspectral data compression using fully convolutional autoencoder", *Remote Sensing*, vol. 14, no. 10, p. 2472, Jan. 2022, Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2072-4292. DOI: 10.3390/rs14102472.

[26] Y. Guo, Y. Chong, Y. Ding, S. Pan, and X. Gu, "Learned hyperspectral compression using a student's t hyperprior", *Remote Sensing*, vol. 13, no. 21, p. 4390, Jan. 2021, Number: 21 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2072-4292. DOI: 10.3390/rs13214390.

[27] Y. Dua, V. Kumar, and R. S. Singh, "Comprehensive review of hyperspectral image compression algorithms", *Optical Engineering*, vol. 59, no. 9, p. 090 902, Sep. 2020, Publisher: SPIE, ISSN: 0091-3286, 1560-2303. DOI: 10.1117/1.OE.59.9.090902.

[28] A. Aidini, M. Giannopoulos, A. Pentari, K. Fotiadou, and P. Tsakalides, "Hyperspectral image compression and super-resolution using tensor decomposition learning", in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA: IEEE, Nov. 2019, pp. 1369–1373, ISBN: 978-1-72814-300-2. DOI: 10.1109/IEEECONF44664.2019.9048735.

[29] C. Deng, Y. Cen, and L. Zhang, "Learning-based hyperspectral imagery compression through generative neural networks", *Remote Sensing*, vol. 12, no. 21, p. 3657, Jan. 2020, Number: 21 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2072-4292. DOI: 10.3390/rs12213657.

[30] S. Kumar, S. Chaudhuri, B. Banerjee, and F. Ali, "Onboard hyperspectral image compression using compressed sensing and deep learning", in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds., vol. 11130, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2019, pp. 30–42, ISBN: 978-3-030-11011-6 978-3-030-11012-3. DOI: 10.1007/978-3-030-11012-3_3.

[31] J. A. Saghri, "Adaptive two-stage karhunen-loeve-transform scheme for spectral decorrelation in hyperspectral bandwidth compression", *Optical Engineering*, vol. 49, no. 5, p. 057 001, May 1, 2010, ISSN: 0091-3286. DOI: 10.1117/1.3425656.

[32] A. Karami, M. Yazdi, and G. Mercier, "Compression of Hyperspectral Images Using Discerete Wavelet Transform and Tucker Decomposition", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 444–450, Apr. 2012, Conference Name: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, ISSN: 2151-1535. DOI: 10.1109/JSTARS.2012.2189200.

[33] M. Conoscenti, R. Coppola, and E. Magli, "Constant SNR, Rate Control, and Entropy Coding for Predictive Lossy Hyperspectral Image Compression", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, pp. 7431–7441, Dec. 2016, Conference Name: IEEE Transactions on Geoscience and Remote Sensing, ISSN: 1558-0644. DOI: 10.1109/TGRS.2016.2603998.

[34] M. Hernández-Cabronero, A. B. Kiely, M. Klimesh, I. Blanes, J. Ligo, E. Magli, and J. Serra-Sagristà, "The CCSDS 123.0-B-2 "Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression" Standard: A comprehensive review", *IEEE Geoscience and Remote Sensing Magazine*, vol. 9, no. 4, pp. 102–119, Dec. 2021, Conference Name: IEEE Geoscience and Remote Sensing Magazine, ISSN: 2168-6831. DOI: 10.1109/MGRS.2020.3048443.

[35] T. Berger, "Rate-distortion theory", in *Wiley Encyclopedia of Telecommunications*, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471219282.eot142, John Wiley & Sons, Ltd, 2003, ISBN: 978-0-471-21928-6. DOI: 10.1002/0471219282.eot142.

[36] M. H. P. Fuchs and B. Demir, *HySpecNet-11k: A Large-Scale Hyperspectral Dataset for Benchmarking Learning-Based Hyperspectral Image Compression Methods*, Jun. 2, 2023. DOI: 10.48550/arXiv.2306.00385. arXiv: 2306.00385[cs,eess].

[37] H. Shen, W. D. Pan, Y. Dong, and Z. Jiang, "Golomb-rice coding parameter learning using deep belief network for hyperspectral image compression", in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Fort Worth, TX: IEEE, Jul. 2017, pp. 2239–2242, ISBN: 978-1-5090-4951-6. DOI: 10.1109/IGARSS.2017.8127434.

[38] D. Hong, Z. Han, J. Yao, L. Gao, B. Zhang, A. Plaza, and J. Chanussot, "SpectralFormer: Rethinking Hyperspectral Image Classification with Transformers", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022, ISSN: 0196-2892, 1558-0644. DOI: 10.1109/TGRS.2021.3130716. arXiv: 2107.02988[cs].

[39] J. E. Fowler and J. T. Rucker, "Three-dimensional wavelet-based compression of hyperspectral imagery", in *Hyperspectral Data Exploitation*, Section: 14 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470124628.ch14, John Wiley & Sons, Ltd, 2007, pp. 379–407, ISBN: 978-0-470-12462-8. DOI: 10.1002/9780470124628.ch14.

[40] Y. Hu, W. Yang, and J. Liu, "Coarse-to-fine hyper-prior modeling for learned image compression", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, pp. 11 013–11 020, Apr. 3, 2020, Number: 07, ISSN: 2374-3468. DOI: 10.1609/aaai.v34i07.6736.

[41] M. J. Wainwright and E. P. Simoncelli, "Scale mixtures of gaussians and the statistics of natural images", 1999.

[42] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, *Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules*, Mar. 30, 2020. DOI: 10.48550/arXiv.2001.01568. arXiv: 2001.01568[eess].

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention Is All You Need*, Dec. 5, 2017. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762[cs].

[44] H. Liu, T. Chen, P. Guo, Q. Shen, X. Cao, Y. Wang, and Z. Ma, *Non-local Attention Optimized Deep Image Compression*, version: 1, Apr. 22, 2019. DOI: 10.48550/arXiv.1904.09757. arXiv: 1904.09757[cs,eess].

[45] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, *Learning Convolutional Networks for Content-weighted Image Compression*, Sep. 19, 2017. DOI: 10.48550/arXiv.1703.10553. arXiv: 1703.10553[cs].

[46] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, *Conditional Probability Models for Deep Image Compression*, Jun. 4, 2019. DOI: 10.48550/arXiv.1801.04260. arXiv: 1801.04260[cs].

[47] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks", *An Introduction to Convolutional Neural Networks*, Nov. 26, 2015, Publisher: arXiv.

[48] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks", *Pattern Recognition*, vol. 77, pp. 354–377, May 1, 2018, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2017.10.013.

[49] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression", *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, Jun. 1987, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/214762.214771.

[50] A. Said, *Introduction to Arithmetic Coding – Theory and Practice*, Feb. 1, 2023. DOI: 10.48550/arXiv.2302.00819. arXiv: 2302.00819[cs,math].

[51] V. Sze and M. Budagavi, "High Throughput CABAC Entropy Coding in HEVC", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1778–1791, Dec. 2012, Conference Name: IEEE Transactions on Circuits and Systems for Video Technology, ISSN: 1558-2205. DOI: 10.1109/TCSVT.2012.2221526.

[52] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, *Conditional Image Generation with PixelCNN Decoders*, Jun. 18, 2016. DOI: 10.48550/arXiv.1606.05328. arXiv: 1606.05328[cs].

[53] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going Deeper with Convolutions*, Sep. 16, 2014. DOI: 10.48550/arXiv.1409.4842. arXiv: 1409.4842[cs].

[54] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer Normalization*, Jul. 21, 2016. DOI: 10.48550/arXiv.1607.06450. arXiv: 1607.06450[cs,stat].

[55] F. A. Kruse, A. B. Lefkoff, J. W. Boardman, K. B. Heidebrecht, A. T. Shapiro, P. J. Barloon, and A. F. H. Goetz, "The spectral image processing system (SIPS)—interactive visualization and analysis of imaging spectrometer data", *Remote Sensing of Environment*, Airbone Imaging Spectrometry, vol. 44, no. 2, pp. 145–163, May 1, 1993, ISSN: 0034-4257. DOI: 10.1016/0034-4257(93)90013-N.

Appendix A

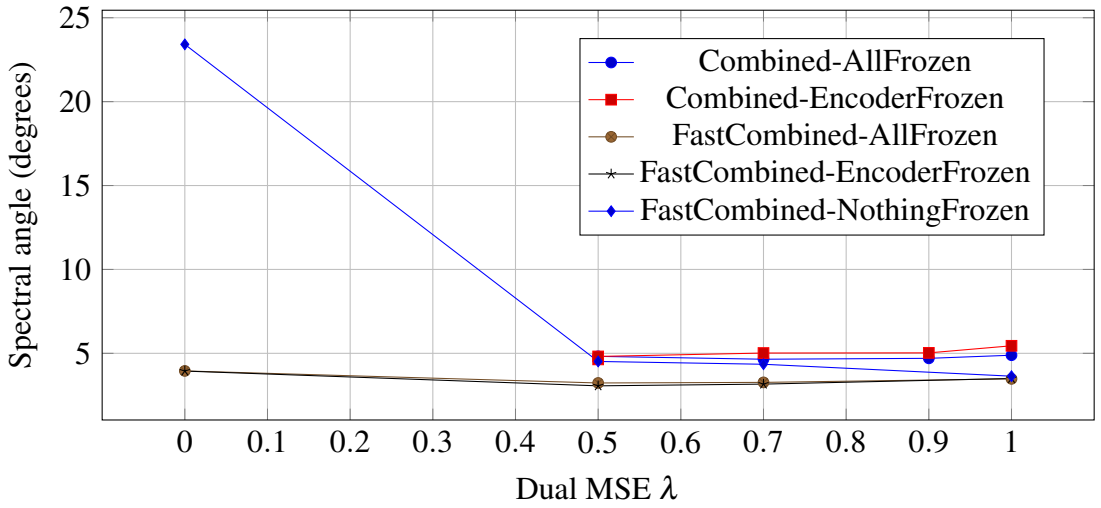

Figure 1: Dual MSE lambda plot

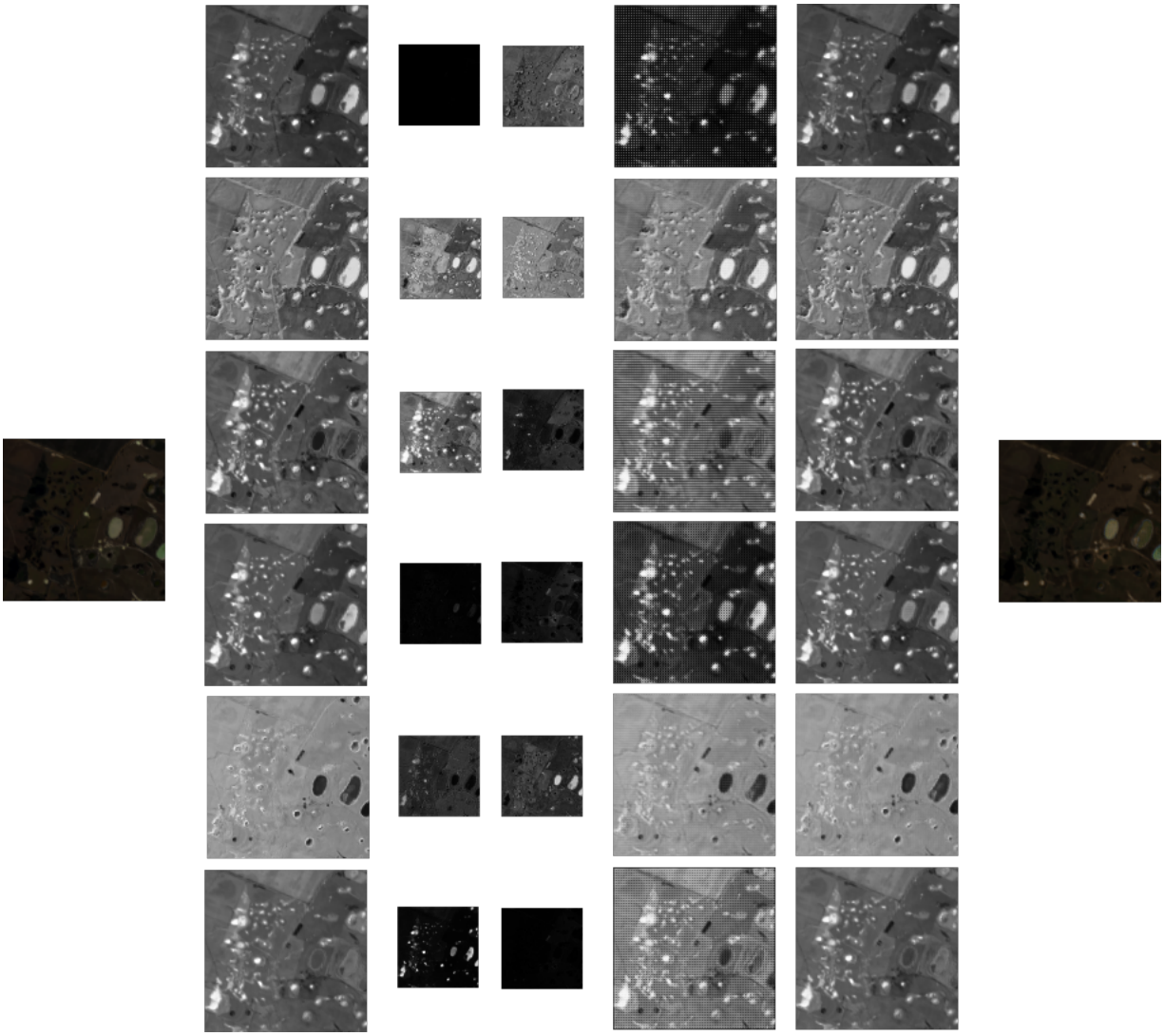

Figure 2: Latent image comparison