Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2021

Reconstruction of Hyperspectral Images using Generative Adversarial Networks

Jacob Eek

LINKÖPING UNIVERSITY

Master of Science Thesis in Electrical Engineering

Reconstruction of Hyperspectral Images using Generative Adversarial Networks

Jacob Eek

LiTH-ISY-EX--21/5447--SE

Supervisors: **Magnus Malmström**
ISY, Linköping University
David Gustafsson
FOI, Swedish Defense Research Agency
Andreas Brorsson
FOI, Swedish Defense Research Agency
Ludwig Hollmann
FOI, Swedish Defense Research Agency

Examiner: **Isaac Skog**
ISY, Linköping University

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

Fast detection and identification of unknown substances is an area of interest for many parties. Raman spectroscopy is a laser-based method allowing for long range no contact investigation of substances. A Coded Aperture Snapshot Spectral Imaging (CASSI) system allows for fast and efficient measurements of hyperspectral images of a scene, containing a mixture of the spatial and spectral data. To analyze the scene and the unknown substances within it, it is required that the spectra in each spatial position are known. Utilizing the theory of compressed sensing allows for reconstruction of hyperspectral images of a scene given their CASSI measurements by assuming a sparsity prior. These reconstructions can then be utilized by a human operator to deduce and classify the unknown substances and their spatial locations in the scene. Such classifications are then applicable as decision support in various areas, for example in the judicial system.

Reconstruction of hyperspectral images given CASSI-measurements is an ill-posed inverse problem typically solved by utilizing regularization techniques such as Total Variation (TV). These TV-based reconstruction methods are time consuming relative to the time needed to acquire the CASSI measurements, which is in the order of seconds. This leads to a reduced number of areas where the technology is applicable.

In this thesis, a Generative Adversarial Network (GAN) based reconstruction method is proposed. A GAN is trained using simulated training data consisting of hyperspectral images and their respective CASSI measurements. The GAN provides a learned prior, and is used in an iterative optimization algorithm seeking to find an optimal set of latent variables such that the reconstruction error is minimized. The results of the developed GAN based reconstruction method are compared with a traditional TV method and a different machine learning based reconstruction method. The results show that the reconstruction method developed in this thesis performs better than the compared methods in terms of reconstruction quality in short time spans.

Sammanfattning

Snabb detektion och identifiering av okända substanser är av intresse för många olika parter. Ramanspektroskopi är en etablerad laserbaserad metod som tillåter kontaktfri analys av substanser från långa avstånd. Ett Coded Aperture Snapshot Spectral Imaging (CASSI) system tillåter snabba och effektiva mätningar av hyperspektrala bilder i en scen, som innehåller en blandning av spatiell och spektral information. Genom att utnyttja teorin om compressed sensing är det möjligt att rekonstruera hyperspektrala bilder i en scen givet dess CASSI-mätningar genom att anta att bilden är gles. Dessa rekonstruerade hyperspektrala bilder kan sedan användas av mänskliga operatörer för att klassificiera okända substanser och deras spatiella positioner i scenen. Sådana klassificieringar är sedan applicerbara som t.ex. bevisföring i rättsliga mål.

Rekonstruktionsproblemet är ett inverst problem som typiskt löses genom att använda sig av regulariseringstekniker såsom Total Variation (TV). Dessa TV-baserade rekonstruktionsmetoder är tidsineffektiva relativt till hur lång tid som krävs för att samla in CASSI-mätningar, vilka kan insamlas inom loppet av sekunder. Detta leder till ett reducerat antal områden för vilka tekniken är applicerbar.

I denna uppsats föreslås en rekonstrutionsmetod baserad på ett s.k. Generative Adversarial Network (GAN). Ett GAN tränas genom att använda träningsdata bestående av hyperspektrala bilder och deras respektive CASSI-mätningar. Det tränade GAN:et används sedan i en iterativ optimeringsalgoritm vars syfte är att finna de optimala latenta variablerna sådana att ett minimalt rekonstruktionsfel uppnås. Resultaten för den utvecklade GAN-baserade rekonstruktionsmetoden jämförs med en traditionell TV-metod, samt en annorlunda typ av maskininlärningsbaserad rekonstruktionsmetod. Resultaten visar att rekonstruktionsmetoden utvecklad i denna uppsats presterar bättre än de jämförda metoderna sett till rekonstruktionskvalitet i korta tidsspann.

Acknowledgments

First and foremost, I would like to thank the Swedish Defence Research Agency, FOI, for giving me this opportunity, and for introducing me to a very interesting area of technology previously unknown to me. Thank you to my supervisors at FOI, David Gustafsson, Andreas Brorsson, and Ludwig Hollman, for providing genuine interest and continuous support throughout the thesis.

I would also like to thank my examiner Isaac Skog, and my supervisor Magnus Malmström, at Linköping University for providing guidance, new ideas, and constructive feedback.

This thesis marks the end of an era. After five years of study I am now leaving university for new adventures. I would like to thank all people I have met and befriended through the studies we have endured, and via all of the student activities I have had the privilege to take part in. All of you have played a part into shaping my experience as a student into some of the most fun and memorable years of my life.

Lastly, I would like to thank my family for always believing in me, and for supporting me through thick and thin. Without you all of this would not have been possible. To my late grandmother, you were always my biggest supporter, in whatever endeavours I took on. I know that you are looking down on me proudly, and I miss you dearly.

Linköping, December 2021
Jacob Eek

Contents

Notation

Mathematical Notation

Notation	Meaning
D_ϕ	Discriminator network containing parameters ϕ
G_θ	Generator network containing parameters θ
$\mathcal{Z}$	Latent space
z	Latent variables
$\mathcal{X}$	True data distribution
x	Hyperspectral image
$\hat{x}$	Reconstructed hyperspectral image
F	Measurement matrix
y	Measurement of a hyperspectral image
$\hat{y}$	Reconstructed measurement of a hyperspectral image
$\|\cdot\|_p$	l_p-norm
$\mathbb{E}[\cdot]$	Expected value
k	BEGAN objective proportionality parameter
γ	BEGAN diversity constant

Abbreviations

Abbreviation	Meaning
BEGAN	Boundary Equilibrium Generative Adversarial Network
CASSI	Coded Aperture Snapshot Spectral Imager
CBEGAN	Conditional BEGAN
CGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
CS	Compressed Sensing
DCGAN	Deep Convolutional Generative Adversial Network
FOI	Swedish Defense Research Agency
GAN	Generative Adversarial Network
HSI	Hyperspectral Image
MAE	Mean Absolute Error
ML	Machine Learning
TV	Total Variation
TVAL3	Total Variation minimization by Augmented Lagrangian and ALternating ALgorithm

1

Introduction

1.1 Background

The ability to identify unknown substances such as explosives or drugs in a scene can be of significant interest for many parties. For example in the law enforcement conducting detective work at a crime scene, or for investigating a potential explosive device. In the latter application, time is usually of the essence. With the risk of an (potential) explosive device, it is critical that identification of the substances in the explosive device is quick, such that bomb disposal may commence as soon as possible. Additionally, it is desirable that the identification may take place from a safe distance from the potential explosive, to minimize the risk of injuries and/or fatalities.

At the Swedish Defence Research Agency (FOI), research has been conducted on the ability of identifying unknown substances in a scene using Raman spectroscopy, a laser-based technique allowing for long range no-contact investigation of unknown substances. A Coded Aperture Snapshot Spectral Imaging system (CASSI) is used to measure hyperspectral images (HSI) of a scene to be investigated. Through the theory of compressed sensing (CS) it is possible to reconstruct HSI from only a few 2-dimensional measurements containing mixed spatial and spectral data using reconstructions algorithms such as the Total Variance minimization by Augmented Lagrangian and ALternating ALgorithm (TVAL3) [1].

The reconstructed HSI allows for human operators to analyze and classify unknown substances at different spatial locations in a scene. The reconstructed HSI can therefore be seen as decision support, and be provided as parts of argumentation and evidence in, for example, the judicial system. Direct classifications from a black-box model such as a neural network are not analyzable in the same sense,

since such a system only produces a prediction with a certain probability score. Furthermore, the decision making of the neural network in classifying the substances can not be known to the users, making it not suitable as decision support. Therefore, direct classification is not preferable in the context of substance detection and identification as compared to providing reconstructed HSI to a human operator.

Although there exists great potential in using Raman spectroscopy based methods for safely identifying unknown substances, there are drawbacks as it currently stands. According to [2], a surface of size 3x3 cm^2 may be measured from a distance of 10 m in a matter of seconds using CASSI. However the reconstruction of the HSI from the measurements using TVAL3 may take up to several minutes. This inhibits close to real-time identification from being realizable. Reconstruction times in parity with the measurement time is desirable.

1.2 Motivation

Some initial research into utilizing machine learning (ML) in HSI reconstruction have been conducted at FOI, with the hope of ML providing much faster reconstruction without a significant loss in reconstruction quality [3]. The initial research has shown promising results, and a further investigation into the viability of ML based methods is therefore of interest.

In [3], the training of a neural network for HSI reconstruction was conducted using simulated HSI training data generated via a model of the physical CASSI-system at FOI. The HSI training data consisted of HSI containing a number of randomly placed rectangles and ellipses of varying size depicting the substances. These geometric shapes were then assigned a Raman spectra each, furthermore white background noise was added to the HSI.

In real-world situations, the substances can typically be mixed, such that one spatial pixel contains a mixture of several substances, and thus several Raman spectra. Therefore it is of interest to investigate whether an ML-based method is able to reconstruct the HSI, and thus the Raman spectra, when the data is more complex containing a mixture of substances.

Investigating whether a different network architecture could improve the quality of the reconstructions is also interesting. In the last years, research have been conducted exploring whether generative models, such as a Generative Adversarial Network (GAN) could be useful in CS, with promising results for simpler datasets [4, 5]. Exploration of the usage of generative models for CS in the context of HSI reconstruction and Raman spectra is of great interest. This by comparing reconstruction time and quality of generative model based reconstruction against reconstruction with TVAL3 and reconstruction with the architecture and objective used in [3]. Examples of reconstruction quality metrics include the mean

absolute error (MAE), and cosine similarity [6].

The iterative TVAL3 algorithm is developed such that an initial guess of a solution is required. Typically a zero matrix is provided when no alternative exists, but a good initial guess of what the solution looks like can possibly reduce the convergence time drastically. Therefore, it could be interesting to provide the output of the network, i.e., synthetic HSI, to TVAL3 as an initial guess, to see whether the number of iterations until convergence and therefore the computation time may reduce. A great reduction in the number of iterations would also indicate that the synthetic HSI from the generative model was of good quality.

In conclusion, for this thesis, the research questions aimed to be answered are:

1. Is a GAN-based reconstruction method viable for reconstruction of hyperspectral images?

2. How do reconstructions achieved by a GAN-based reconstruction method compare to the reconstructions using the TVAL3-algorithm and the encode-decode architecture in [3] in terms of computational time and precision?

3. Is reconstruction possible when the hyperspectral image contains two or more mixed substances?

4. Are synthetic hyperspectral images from the GAN useful and applicable as an initial guess to the TVAL3-algorithm?

2

Theory and Previous Works

This chapter introduces theory relevant for the work done during this thesis. Previous works which provides building blocks, and have inspired, the developed and investigated reconstruction method are also presented.

2.1 Raman Spectroscopy

Raman spectroscopy is a spectroscopy method which allows for identification of unknown substances. By illuminating an unknown substance with monochromatic light, the incident photons will hit the molecules and become scattered. There are two different types of scattering, depending on if the energy of the incoming photon E_{laser} has the same energy as the outgoing photon E_{out} after its collision with the molecule or not. The energy of a photon is given by

$$E = \frac{hc}{\lambda}, \tag{2.1}$$

where h is the Planck constant, and c is the speed of light in vacuum. Hence, the energy is inversely proportional to the wavelength of the photon. If the scattering causes the outgoing photon to have the same wavelength as the original photon, the scattering effect is called Rayleigh scattering. Should the outgoing photon have a different wavelength than the original photon, the scattering is called Raman scattering, see Figure 2.1. This shift in wavelength that occurs depends on the molecular structure of the unknown substance, and thus *is unique* to a specific substance. Hence, by measuring the shift in wavelength, one can identify the unknown substance [7].

According to [8], a general Raman spectrum r for some substance can be mod-

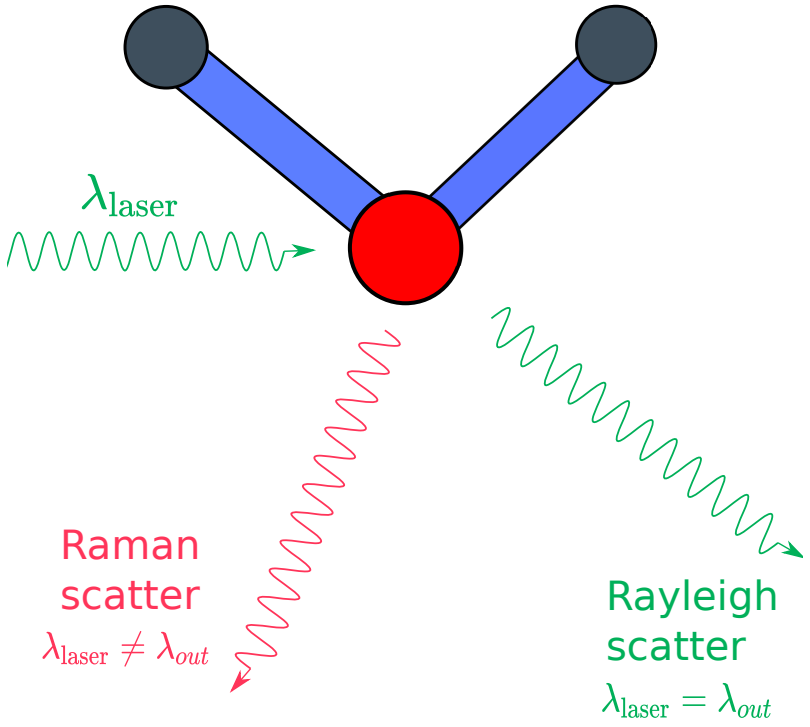


Figure 2.1: *The two different types of scattering, Rayleigh scattering and Raman scattering.*

eled as a sum of wavelength peaks and some underlying noise

$$r = \sum_{i=1}^{n} p_i + \epsilon, \tag{2.2}$$

where p_i denotes the peaks, and ϵ is the noise. One common model of the peaks p_i is the Lorentzian lineshape

$$p_i \approx L_i(\lambda) = \frac{a_i}{1 + \left(\frac{2(\lambda - \lambda_i)}{b_i}\right)^2}, \tag{2.3}$$

where a_i is the amplitude, b_i is the full width at half maximum and λ_i denotes the wavelength of the peak. Figure 2.2 contains an example of a Lorentzian lineshape. Figure 2.3 contains a toy model of a Raman spectrum generated using (2.2) with no added noise.

It is noticeable that typical Raman spectra are sparse in nature. The Raman spectra contain a few peaks with higher intensity, but are approximately zero intensity for all other wavelengths.

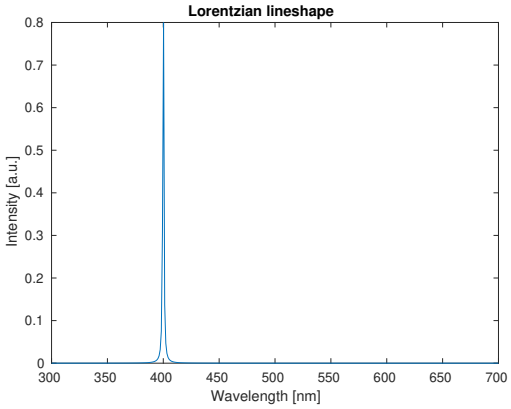


Figure 2.2: *Lorentzian lineshape $L_i(\lambda)$ with parameter values: $a_i = 1,\ b_i = 1,\ \lambda_i = 400$.*

2.2 Compressed Sensing

Compressed sensing is a signal processing technique allowing accurate signal reconstruction, with significantly less measurements than required by the Shannon-Nyquist theorem. This is possible by utilizing that the majority of natural images can be well approximated by sparse vectors in some appropriate basis, i.e., a sparsity prior is assumed on the images, where Fourier and wavelet bases are commonly used in image processing problems [9]. Let x denote a not fully observable signal that are to be reconstructed and

$$y = Hx + \epsilon, \tag{2.4}$$

are the available linear measurements of x, where ϵ denotes noise. If x is sparsely representable in some basis Ψ

$$x = \Psi\theta, \tag{2.5}$$

then (2.4) can be rewritten as

$$y = H\Psi\theta + \epsilon = A\theta + \epsilon, \tag{2.6}$$

where Ψ denotes the basis matrix for which x is transformed into a k-sparse vector θ, i.e., $\|\theta\|_0 = k$. By representing the original signal as a sparse vector, it is possible to reconstruct the signal accurately and computationally effectively [9].

The objective function of the signal recovery problem is to minimize the cardinality of θ, whilst fulfilling that the noiseless measurement $A\theta$ is ϵ-close to the actual measurement y. Thus, the optimization problem to be solved can be stated as

$$\min_{\theta} \quad \|\theta\|_0 \tag{2.7a}$$

$$\text{s.t.} \quad \|y - A\theta\|_2 \leq \epsilon. \tag{2.7b}$$

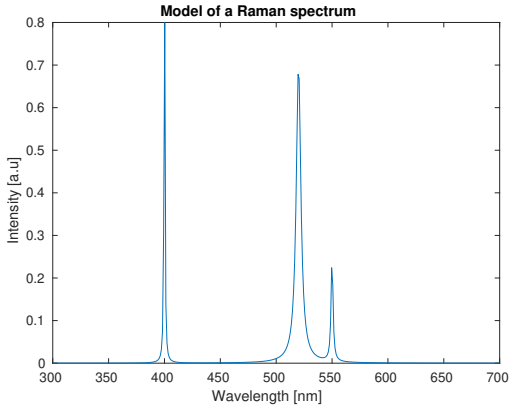


Figure 2.3: *Model of a Raman spectrum with three peaks.*

Unfortunately, is has been shown that finding an optimal solution to this optimization problem is an NP-hard combinatorial problem, thus rendering the problem not feasable for application in systems where reconstruction time is critical [9].

However, it has also been shown that a relaxation of the optimization problem can yield an exact solution given that some conditions are fulfilled. Given a random design matrix A, e.g., a binary matrix with i.i.d. elements, exact reconstruction is possible for a k-sparse n-dimensional signal given at least $\mathcal{O}(k \log n/k)$ measurements y. With these conditions fulfilled, the optimization problem in (2.7) may be relaxed

$$\min_{\theta} \quad \|\theta\|_1 \tag{2.8a}$$

$$\text{s.t.} \quad \|y - A\theta\|_2 \leq \epsilon. \tag{2.8b}$$

The optimization problem in (2.8) is convex, and can be formulated as a linear program. Hence, it is easily solvable by standard optimization techniques, as compared to the optimization problem in (2.7) [9].

2.3 CASSI

Coded aperture snapshot spectral imaging is a method for measuring 3-dimensional HSI (i, j, λ) based on the theory of CS. Compared with conventional sampling methods of HSI, where the measurements are typically acquired by temporal scanning in the spatial or spectral domain, the measurements acquired from CASSI consists of a mixture of spatial and spectral data encoded onto a 2-dimensional charged couple device (CCD) detector. The encoding of the data happens simultaneously with the usage of coded apertures and a dispersive element such as a

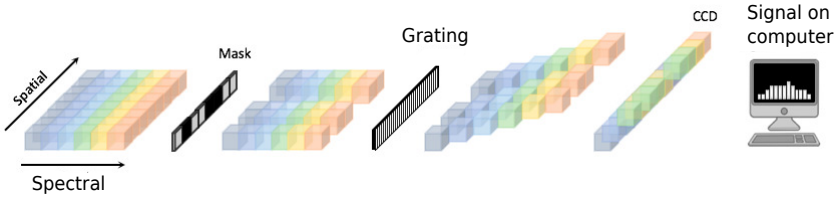


Figure 2.4: *The simplified CASSI model. Adapted from [3].*

prism or grating. Given a HSI x, the measurements y acquired from CASSI can be written as

$$y = Hx, \tag{2.9}$$

where the matrix operator H represents the light propagation through the system [1].

In previous work by Brorsson *et al.* [3], a simplified linear model F of the CASSI-system was used for producing measurements y from the HSI x. The simplifications made in the transfer function were:

- One spatial row at the time is measured.
- The dispersion was constant, i.e., the wavelength dependence of the grating step was ignored.
- No optical abberation.
- The spatial resolution was set to 64×64, and the spectral resolution was kept to 512 bins.
- Four different coded apertures was used per HSI x.

The CASSI model for measuring one row is illustrated in Figure 2.4. A spatial row of the HSI hits a coded aperture, resulting in some parts of the light being blocked. The remaining light is then dispersed through a grating. This dispersion is simplified in the model by shifting. The light is shifted one pixel spatially for each wavelength, resulting in e.g. the first wavelength being shifted by one pixel, and the p:th wavelength being shifted p pixels. The results after the dispersion is then encoded onto the CCD detector and comprises the measurement y.

2.4 TVAL3

The TVAL3-algorithm is a Total Variation (TV) regularization algorithm applicable for the HSI reconstruction [10]. The main difference between TVAL3 and other reconstruction algorithms is that TVAL3 assumes a sparsity prior on the gradients of the signal, instead of a sparsity prior on the signal itself. Since natural images often tend to contain sharp edges and piecewise smooth areas, the gradients of

these images should be mostly sparse, hence the TVAL3 performs fairly well in image reconstruction [11]. The optimization problem in TVAL3 is defined as

$$\min_x \quad \sum_i \|D_i x\|_1 \tag{2.10a}$$

$$\text{s.t.} \quad y = Fx, \tag{2.10b}$$

where D_i denotes the discrete gradient of x at position i. Thus, in TVAL3, the l_1−minimization problem in (2.8) is solved by letting the basis Ψ in (2.5) be chosen as the gradient basis of x.

An alternative to the vanilla TVAL3 algorithm is the TVAL3+ algorithm, which differs from the former simply by constraining all values in the image to be reconstructed to be non-negative. This leads to the slightly differing optimization problem to be solved

$$\min_x \quad \sum_i \|D_i x\|_1 \tag{2.11a}$$

$$\text{s.t.} \quad y = Fx, \; x \geq 0 \tag{2.11b}$$

2.5 Generative Adversarial Network

A Generative Adversarial Network (GAN) is a type of ML framework whose aim is to generate new data with the same statistical distribution as the training set via unsupervised learning. The GAN was introduced by Goodfellow *et al.* in 2014 [12], and contains two distinct neural networks, the generator G_θ, and the discriminator D_ϕ, where θ and ϕ are the parameters of the respective networks.

The generator is a generative model whose goal is to map representations z from a low-dimensional latent space $\mathcal{Z}$, into a higher dimensional space $G_\theta(\mathcal{Z})$, such that for every training sample $x \sim \mathcal{X}$, there exists at least one latent representation z such that $G_\theta(z) \approx x$.

The discriminator is a discriminative model whose goal is to learn to classify whether a given image is real, or a fake image generated by the generator. The output of the discriminator is the probability of whether a given image belongs to the training dataset. This leads to a binary classification problem where the goal is to tune the parameters ϕ of the discriminator such that

$$\begin{cases} D_\phi(x) = 1, & x \sim \mathcal{X}, \\ D_\phi(x) = 0, & x \sim G_\theta(\mathcal{Z}). \end{cases} \tag{2.12}$$

See Figure 2.6 for a schematic of the training process of a GAN. Hence, in the train-

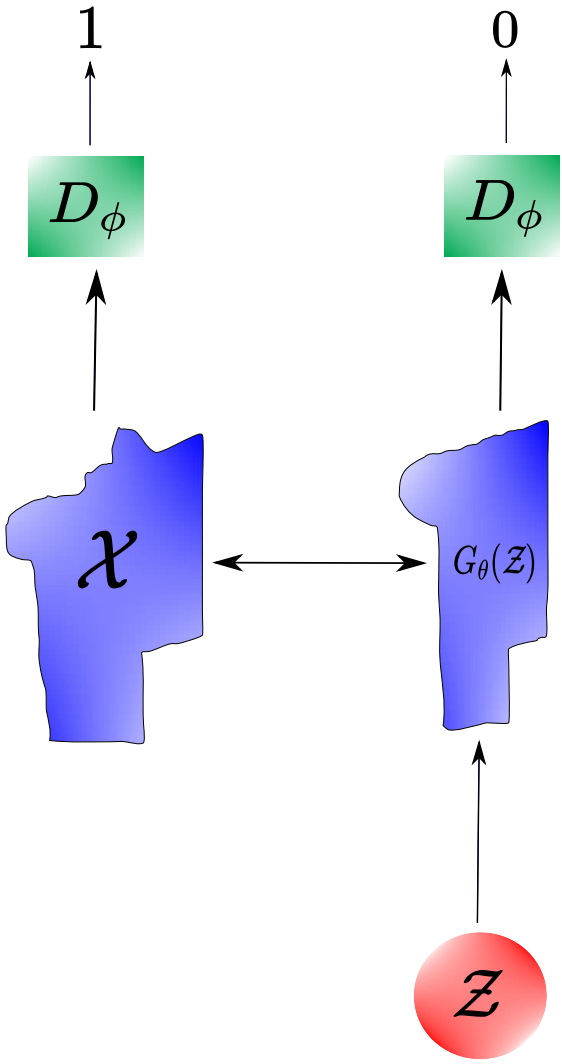


Figure 2.5: *The generator G_θ is a mapping from a low dimensional latent space $\mathcal{Z}$ to a more complicated space $G_\theta(\mathcal{Z})$ approximating the true data distribution $\mathcal{X}$. The goal of the discriminator is to differentiate between samples from the two distributions $\mathcal{X}$ and $G_\theta(\mathcal{Z})$.*

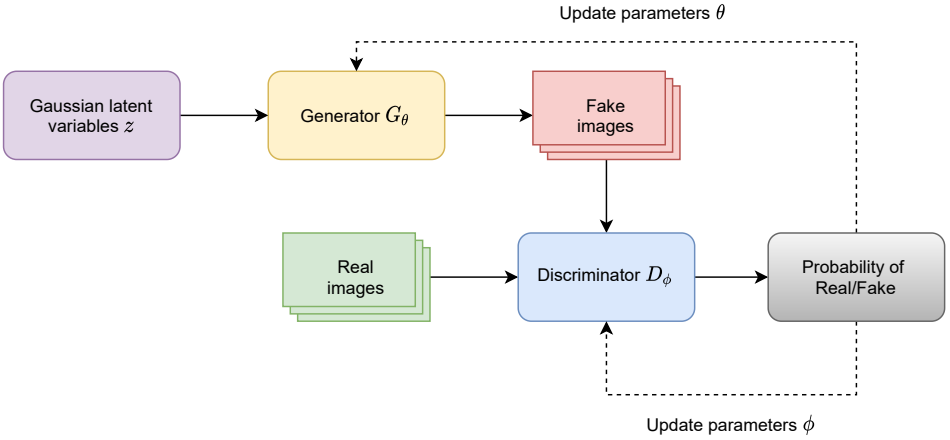


Figure 2.6: *A schematic of the training process of a GAN.*

ing of the GAN, the generator and discriminator can be seen as competing against each other. The generator tries to generate images that look real in order to trick the discriminator, and the discriminator tries to get better at distinguishing between real and fake images in order to not get tricked. Therefore the training of the two models are coupled, and seeks to find the optimal parameters (θ^*, ϕ^*) given by the solution to the min-max problem

$$(\theta^*, \phi^*) = \underset{\theta}{\operatorname{argmin}} \underset{\phi}{\operatorname{argmax}} \mathop{\mathbb{E}}_{x\sim\mathcal{X}}\Big[\ln D_\phi(x)\Big] + \mathop{\mathbb{E}}_{z\sim\mathcal{Z}}\Big[\ln(1 - D_\phi(G_\theta(z)))\Big], \tag{2.13}$$

where $\mathbb{E}[\,\cdot\,]$ denotes the expected value operator. The defined loss function in (2.13) may however prove difficult to utilize in practice. Most probably, the generator G_θ will perform poorly in generating real-looking images in the beginning of training. This leads to the discriminator D_ϕ having an easy time in labeling whether data is real or fake, in turn leading to that the minimization of $\ln(1 - D_\phi(G_\theta(z)))$ may saturate early on. The result of this are generator gradients vanishing, rendering the generator unable to improve itself.

However, with a simple change of perspective, the problem of vanishing of gradients early on in training may be eliminated. The minimization in (2.13) can be interpreted as the generator trying to minimize the probability of an image being perceived as fake. An equivalent interpretation, but with a flipped perspective would be for the generator trying to maximize the probability of an image being perceived as real. This leads to the so called non-saturating GAN-loss given by

$$(\theta^*, \phi^*) = \underset{\theta}{\operatorname{argmax}} \underset{\phi}{\operatorname{argmax}} \mathop{\mathbb{E}}_{x\sim\mathcal{X}}\Big[\ln D_\phi(x)\Big] + \mathop{\mathbb{E}}_{z\sim\mathcal{Z}}\Big[\ln(D_\phi(G_\theta(z)))\Big]. \tag{2.14}$$

Since its introduction in 2014, many different types of GAN consisting of varying network architectures and loss functions have been investigated.

2.5.1 Deep Convolutional Generative Adversarial Network

In 2016, Radford, Metz, and Chintala introduced the Deep Convolutional Generative Adversarial Network (DCGAN) [13]. The authors took inspiration from continued success utilizing Convolutional Neural Networks [14] (CNN) in supervised learning during the years prior. In DCGAN, the generator G_θ and discriminator D_ϕ are represented by CNN, and the authors provided architectural guidelines for how the two models were to be constructed.

The key components in the DCGAN architecture is the usage of convolutional and transposed convolutional layers in the discriminator and generator respectively, allowing the network to itself learn good up- and downsampling operations, leading to an increased quality in image synthesis. Furthermore, [13] proposes the usage of batch normalization [15], which showed to have a significant impact in avoiding a mode collapse during training of GAN, where the variance of the output images from the generator is notoriously low. Lastly, recommendations of layer activation functions were provided. The rectified linear unit (ReLU) is to be used in all generator layers, except for the output layer, where the authors recommend using the hyperbolic tangent. In the discriminator, the authors found success with using the leaky ReLU as an activation function after all layers.

2.5.2 Conditional Generative Adversarial Network

First introduced by Mirza and Osidenro in 2014, the Conditional Generative Adversarial Network (CGAN) allows for greater on control on the data generation process by conditioning on additional information, for example, class labels [16]. The additional information y is fed as an input to both the discriminator and generator, and the goal is for the generator to learn how to produce an output image x given the additional information y. An example for this could be the usage of class labels such as *"dog"* and *"cat"*. Given the additional information class label $y =$*"cat"*, the goal is for the generator to produce an image of a cat, and vice versa for $y =$*"dog"*. However, the additional information conditioned on does not necessarily have to be class labels. In [17], CS using CGAN is explored by Kim, Lee, and Yang. The authors studied whether conditioning on measurements $y = Fx$ as the additional information could improve the images given from the generator, with promising results on simpler datasets. The objective of CGAN is formulated similarly to the min-max objective in vanilla GAN (2.14), with the addition of conditioning on the auxiliary information y provided to the networks

$$(\theta^*,\ \phi^*) = \underset{\theta}{\operatorname{argmax}}\ \underset{\phi}{\operatorname{argmax}}\ \mathop{\mathbb{E}}_{x\sim\mathcal{X}}\Big[\ln D_\phi(x|y)\Big] + \mathop{\mathbb{E}}_{z\sim\mathcal{Z}}\Big[\ln(D_\phi(G_\theta(z|y)))\Big] \tag{2.15}$$

See Figure 2.7 for an illustration of the CGAN training process.

2.5.3 Boundary Equilibrium Generative Adversarial Network

Introduced by Berthelot, Schumm, and Metz in 2017 [18], the Boundary Equilibrium Generative Adversarial Network (BEGAN) provided yet another architecture

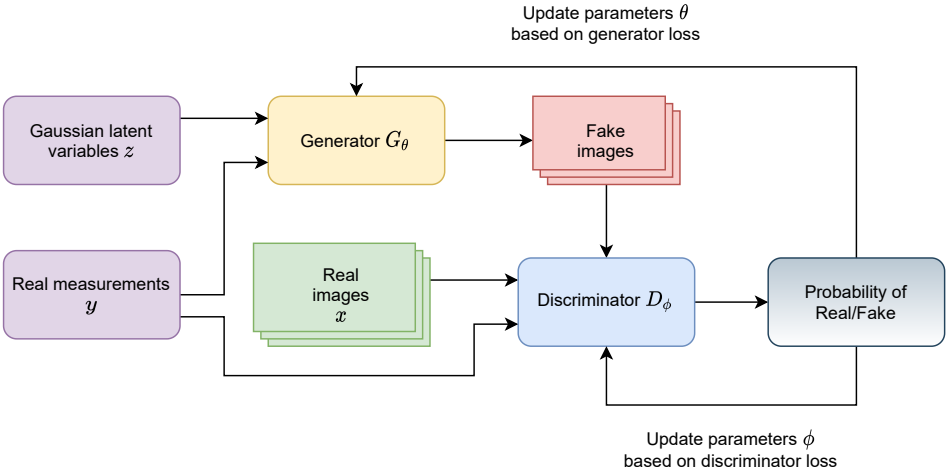


Figure 2.7: *A schematic of the training process of a CGAN. The auxiliary information y is fed to both the generator and discriminator.*

successful in generating diverse and real-looking images. In previously discussed GAN architectures, the discriminator D_ϕ is constructed as a classifier attempting to determine whether a given image is real or fake. BEGAN instead utilizes the idea of autoencoders in the construction of the discriminator. The theory of autoencoders will not be covered in this thesis, instead the reader is directed to [14]. The encoding part extracts latent features of the given input image, which is then reconstructed using the decoder part. This process captures essential features of the input image in a bottleneck between the two parts.

BEGAN objective

The generator G_θ works similarly as in previously discussed GAN. The goal is to map low-dimensional samples from the latent space $z \sim \mathcal{Z}$ such that for all training data $x \sim \mathcal{X}$, there exists a latent variable z such that $G_\theta(z) \approx x$, i.e., the generator should ideally represent the true data distribution $\mathcal{X}$ perfectly.

Since the discriminator D_ϕ now is an autoencoder, the output from the network is no longer a probability of a given input belonging to the real dataset, but rather an entire reconstructed image. The premise of the BEGAN is that matching the reconstruction loss distributions of real and fake images in turn will lead to the proper data distributions of real and fake images being matched. Let $Dec(Enc(x))$ denote a real image x that has passed through the discriminator, and thus have been encoded and decoded. The reconstruction error can then be defined as

$$R_{D_\phi(x)} = \|Dec(Enc(x)) - x\|_p, \tag{2.16}$$

where p denotes the l_p norm used for measuring the reconstruction error. Similarly, the reconstruction error of a fake image $G_\theta(z)$ given by the generator can be

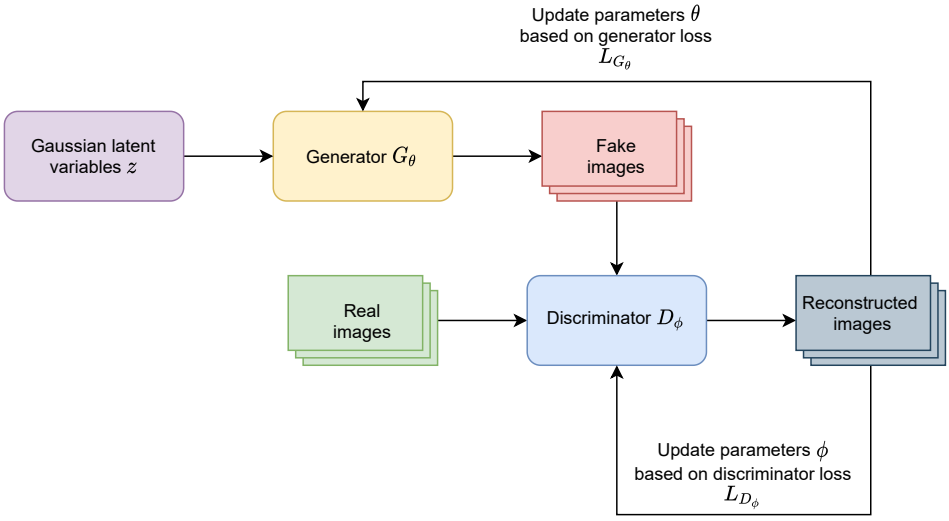


Figure 2.8: *A schematic of the training process of a* BEGAN.

defined as

$$R_{D_\phi(G_\theta(z))} = \|Dec(Enc(G_\theta(z))) - G_\theta(z)\|_p. \tag{2.17}$$

Similarly as in vanilla GAN, the discriminator and generator can be seen as competing against each other in a game. In BEGAN we seek to train the discriminator such that it autoencodes real images well and generated images poorly

$$R_{D_\phi(x)} \to 0$$
$$R_{D_\phi(G_\theta(z))} \to \infty,$$

leading to the following discriminator objective

$$L_{D_\phi} = \min_\phi [R_{D_\phi(x)} - R_{D_\phi(G_\theta(z))}]. \tag{2.18}$$

At the same time, we seek to train the generator such that the fake image reconstruction error is minimized, which means that the discriminator will perform well on generated images

$$L_{G_\theta} = \min_\theta R_{D_\phi(G_\theta(z))}. \tag{2.19}$$

A schematic of the BEGAN training process can be found in Figure 2.8.
In words, the discriminator can be seen as having two goals:

1. Being a good autoencoder, meaning that $R_{D_\phi(x)}$ should be as low as possible.

2. Being able to clearly distinguish between real and fake images.

These two goals are competing against each other, since improved autoencoding qualities on one hand lowers $R_{D_\phi(x)}$, but on the other hand provides better feedback to the generator on how to generate fake images. This in turn lowering $R_{D_\phi(G_\theta(z))}$ contradicting the objective function in (2.18).

A good balance between these two goals during training is desirable. Equilibrium between the discriminator and generator has been reached if

$$\mathbb{E}[R_{D_\phi(x)}] = \mathbb{E}[R_{D_\phi(G_\theta(z))}], \tag{2.20}$$

that is, the expected value of the reconstruction losses of real images is equal to the expected value of reconstruction losses of fake generated images. This means that the generator is generating fake images that are indistinguishable from the real samples for the current discriminator, and that the training will terminate.

Therefore, a slight modification is made to the loss function for the discriminator L_{D_ϕ} in (2.18). A parameter $k \in [0, 1]$ is defined via the equilibrium concept as

$$\frac{1}{k} = \frac{\mathbb{E}[R_{D_\phi(G_\theta(z))}]}{\mathbb{E}[R_{D_\phi(x)}]} \tag{2.21}$$

leading to the modified updated loss function for the discriminator

$$L_{D_\phi} = \min_{\phi}[R_{D_\phi(x)} - kR_{D_\phi(G_\theta(z))}]. \tag{2.22}$$

The introduction of the parameter k allows the discriminator to maintain a balance between its two goals. However, since the value of k may vary greatly from batch to batch, which is undesirable, the value of k is continuously updated via

$$L_{D_\phi} = \min_{\phi}[R_{D_\phi(x)} - k_t R_{D_\phi(G_\theta(z))}] \tag{2.23}$$

$$k_{t+1} = k_t + \lambda_k(R_{D_\phi(x)} - k_t R_{D_\phi(G_\theta(z))}). \tag{2.24}$$

This in order to maintain how much emphasis is to be put on $R_{D_\phi(G_\theta(z))}$ during training to keep the equilibrium. The parameter is initialized to $k_0 = 0$ in order to allow the discriminator to develop its autoencoding capability.

Finally, the concept of the diversity constant $\gamma \in [0, 1]$ is introduced, which is also defined via the equilibrium concept as

$$\gamma = \frac{\mathbb{E}[R_{D_\phi(G_\theta(z))}]}{\mathbb{E}[R_{D_\phi(x)}]}, \tag{2.25}$$

and it allows for a user designated bias in the discriminator goals discussed above. A lower value of γ could lead to greater image quality, but the cost is less diversity in the generated fake images. This due to the fact that a lower γ incentives

the discriminator to focus more on the goal of autoencoding real images. The value of γ is chosen by the user, and held constant throughout training.

To summarize, the final BEGAN objective utilizing all introduced parameters can be written as

$$L_{D_\phi} = \max_{\phi}[R_{D_\phi(x)} - k_t R_{D_\phi(G_\theta(z))}] \tag{2.26a}$$

$$L_{G_\theta} = \min_{\theta} R_{D_\phi(G_\theta(z))} \tag{2.26b}$$

$$k_{t+1} = k_t + \lambda_k(\gamma R_{D_\phi(x)} - R_{D_\phi(G_\theta(z))}). \tag{2.26c}$$

Convergence Measure

A big dilemma in training GAN is the lack of proper methodology for monitoring progress during training. A common method is simply visual inspection of the generator output. After each epoch, the user inspects the output of the generator, and given that the user has knowledge of how typical real images in the training set looks like, one can give some judgment on whether the network is improving. However, this methodology obviously has its flaws. The user may input its own biases and opinions since the method is strictly subjective. The fake image diversity can also be difficult to determine.

To remedy this, BEGAN provides a convergence measure $\mathcal{M}$ defined as

$$\mathcal{M} = R_{D_\phi(x)} + |\gamma R_{D_\phi(x)} - R_{D_\phi(G_\theta(z))}|, \tag{2.27}$$

and the introduction of the convergence measure allows for a clearer insight into how training progresses. The convergence measure is based on the reconstruction error of autoencoded real images, and the difference of the reconstruction errors of fake and real images. If the convergence measure has gone to zero, this would indicate that both real and fake images are reconstructed perfectly. Ideally, it is desired that the measure converges to the lowest value possible, since that indicates that reconstruction of both real and fake images are good.

3

Hyperspectral Image Reconstruction Methodology

The GAN based HSI reconstruction method is inspired by previous work by Bora *et al.* [4]. With the GAN based reconstruction approach, a learned prior is provided by the trained generator G_θ in a GAN, and the images to be reconstructed is assumed to lie near the range of G_θ. This differs from conventional CS, where a sparsity prior on the images in some basis is assumed. A learned prior allows for capturing the inherent structure of sparsity, but gives the possibility to capture other structures as well, structures that may not be determinable a priori by a human.

The method can essentially be broken down into two distinct steps. First a GAN is trained using simulated HSI training data x, and corresponding measurements y to obtain a mapping, the generator G_θ, from the low dimensional latent space $\mathcal{Z}$ to the higher dimensional image space $G_\theta(\mathcal{Z})$ approximating the true training data space $\mathcal{X}$. Secondly, the generator is used in an iterative optimization algorithm. Utilization of the generator in the optimization is possible due to the differentiability of the generator.

The goal of reconstruction is that the recovered HSI $\hat{x}$ is as close to the real HSI x as possible:

$$\|x - \hat{x}\| \approx 0. \tag{3.1}$$

However, typically only compressed measurements y of the real HSI x are available. The premise is that minimizing the norm between recovered measurements $\hat{y}$ and true measurements y leads to the norm between the recovered and real HSI being minimized

$$\|y - \hat{y}\| = \|Fx - F\hat{x}\| \to 0 \Rightarrow \|x - \hat{x}\| \to 0, \tag{3.2}$$

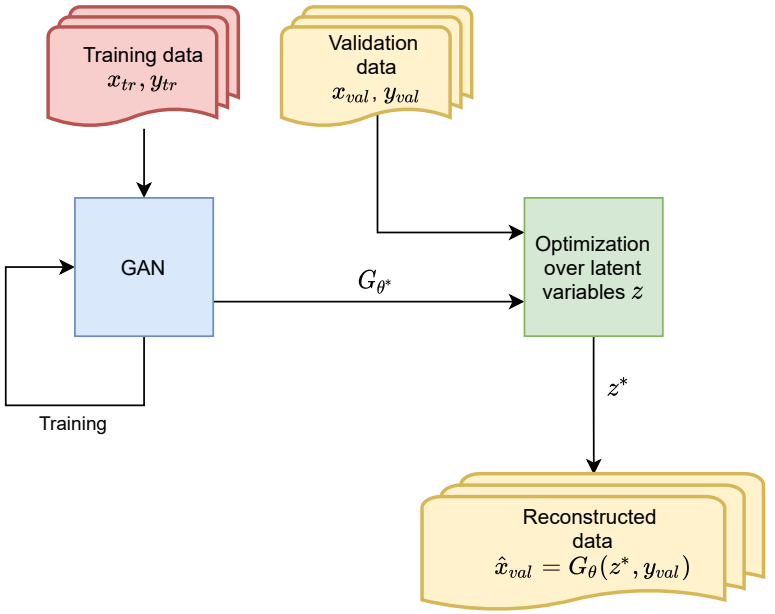


Figure 3.1: *An illustration of the complete reconstruction process.*

which holds provided that the measurement matrix F is full rank.

Hence, the optimization algorithm seeks to find the optimal latent variable values z^* such that the measurement error is minimized

$$z^* = \underset{z}{\operatorname{argmin}} \|y - FG_\theta(z|y)\|_2. \tag{3.3}$$

Finally the optimal latent variables z^* are used to retrieve the reconstructed image via the generator

$$\hat{x} = G_\theta(z^*|y). \tag{3.4}$$

A flowchart illustrating the complete reconstruction process can be found in Figure 3.1.

3.1 Data Generation

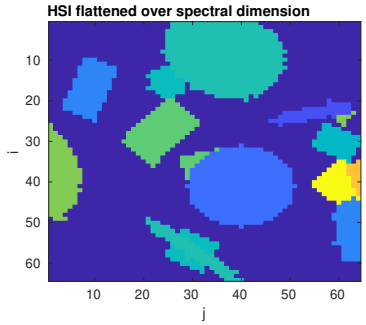
The training data consisted of pairs of generated HSI x of dimension $(i, j, \lambda) = (64, 64, 512)$, and their respective measurements $y = Fx$. The data was generated in MATLAB by placing rectangles and ellipses of random shape, size, and rotation onto a random spatial position. Each target shape was then assigned one Raman spectrum generated using (2.2). All spectra contained the wavelengths $\lambda \in [\lambda_{min}, \lambda_{max}] = [300, 700]$. The parameter values for generating the Raman spectra were sampled as follows:

- Number of peaks: $p \sim \mathcal{U}_d(1, 5)$
- Location for peak i: $\lambda_i \sim \mathcal{U}_d(\lambda_{min}, \lambda_{max})$
- Maximum intensity of peak i: $a_i \sim \mathcal{U}_c(0, 1)$
- Width of peak i: $b_i \sim 1 + 4\mathcal{U}_c(0, 1)$,

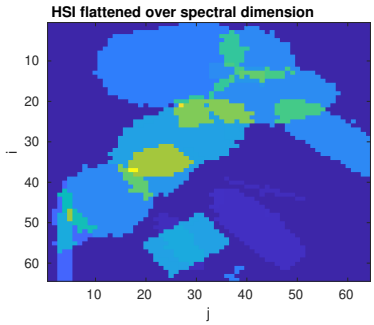
where $\mathcal{U}_d(a, b)$ denotes the discrete uniform distribution and and $\mathcal{U}_c(a, b)$ denotes continuous uniform distribution over the support $[a, b]$.

Two different datasets were created, both containing 3000 HSI, resulting in a total of $3000 \cdot 64 = 192000$ rows of dimension $(j, \lambda) = (64, 512)$ used as inputs to the GAN. In both datasets, the HSI contains 16 target shapes, each randomly assigned 1 out of 16 generated Raman spectra. The key difference between the datasets is whether the target shapes are allowed to overlap or not. Overlapping target shapes may lead to single spatial pixels containing two or more Raman spectra, simulating the real-life case of mixed substances. Examples of generated HSI from both datasets are shown in Figure 3.2.

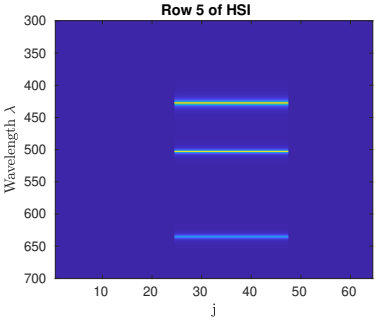
The corresponding measurements y to the simulated HSI x were acquired by the simplified linear model F previously described in section 2.3. The four different coded apertures that were used can be seen in Figure 3.3. The produced measurements contains $n_0 = K(j + \lambda - 1) = 4(64 + 512 - 1) = 2300$ elements, where K denotes the number of coded apertures used, and the rows contain $n_1 = j\lambda = 64 \cdot 512 = 32768$ elements. This results in a compression ratio of $n_0/n_1 \approx 7\%$.



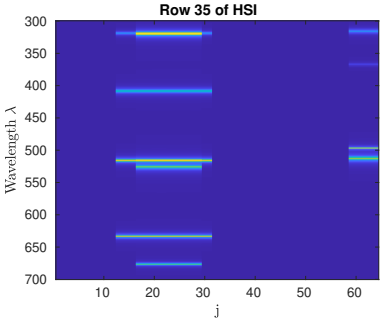
(a) *An example of simulated HSI sampled from the dataset not allowing target shape overlap.*



(b) *An example of a simulated HSI sampled from the dataset allowing target shape overlap.*



(c) *Row 5 ($i = 5$) of the HSI in (a). The image shows that the Raman spectrum belonging to the green ellipse have three peaks.*



(d) *Row 35 ($i = 35$) of the HSI in (b). Notice that several pixels around $j = 24$ contain two different Raman spectra.*

Figure 3.2: *The figures (a) and (b) show examples of data samples from the two different datasets. The color of the target shapes indicates the spectral intensity in the pixel, relative to the other pixels. Notice the lighter areas in (b), indicating a higher spectral intensity due to the pixels containing overlapping target shapes, and therefore several Raman spectra.*

Figure 3.3: *The four coded apertures used to capture the measurements of the HSI.*

3.2 Hyperspectral Image Generation using Conditional BEGAN

As previously mentioned, the goal is to be able to reconstruct HSI x from its compressed measurement $y = Fx$ by utilizing prior knowledge on some structure of the HSI. In TVAL3+, a sparsity prior on the gradient space of the HSI is assumed. However, by training a GAN with the training data described in section 3.1, a learned prior can be used to estimate the HSI from its measurement. By training the GAN, the generator learns a mapping from the low dimensional latent space $\mathcal{Z}$ to the more complex sample space $G_\theta(\mathcal{Z})$. Through the objective of the GAN, the generator is incited to generate new HSI reminiscent of the HSI in the training data set. Hence, the generator provides the learned prior, and captures the essence of what constitutes a HSI in this context.

By training a GAN, and extracting the generator, it is possible to generate new samples of HSI resembling the ones in the training data. However, the goal of this thesis is not to generate new samples of HSI similar to the training data, but rather to reconstruct a specific HSI x^* given its compressed measurement y. Since the compressed measurements are available as training data, they are exploitable during the training of the GAN in learning of its network parameters. The measurements y can be seen as linearly compressed signals of the HSI, thus y includes information about the HSI x^*. This is useful in order to provide further control on what samples the generator of the GAN outputs, and by providing the measurement y to the generator, the premise is that the generated sample $G_\theta(z|y)$ lies closer to the actual HSI x^* than what would be the case without the providing of y. Essentially, by utilizing the concept of CGAN, the learned prior in the form of the generator is refined by utilizing Gaussian distributed latent variables z in conjunction with the specific measurements y. Given a trained conditional generator containing the optimal network parameters G_{θ^*}, taking a latent variable z and measurement y as inputs, an estimation of the HSI x^* is given by:

$$\hat{x} = G_{\theta^*}(z|y). \tag{3.5}$$

The BEGAN framework previously presented in Section 2.5.3 provides for easy implementation of the conditioning on measurements y. The measurements y are simply concatenated to the latent variables fed to the generator, and to the latent variables in the bottleneck of the discriminator autoencoder. Figure 3.4 shows a diagram of the training and validation processes. The loss functions L_{D_ϕ} and L_{G_θ} found in the diagram are described in Section 3.2.1.

The introduction of conditioning on the measurements y in order to produce specific samples of HSI holds some similarity to the method studied in [3], which is benchmarked against later in this thesis. In [3], a decoder network is utilized to directly learn a mapping from the compressed measurements y to the HSI x. However, the utilization of a generative model comes with an advantage against

direct regression as in [3]. The fact that a generative model contains the latent variables z, sampled from the latent space $\mathcal{Z}$, provides a further way of improving the output $\hat{x}$ from the generator. Given the trained generator G_{θ^*}, we consider its network parameters locked in place. Since the generator is differentiable, the latent variables z can be optimized in order to find the optimal latent representation z^*, leading to the estimate reconstruction $\hat{x}$ lying closer to the true HSI x^*

$$\hat{x} = G_{\theta^*}(z^*|y). \tag{3.6}$$

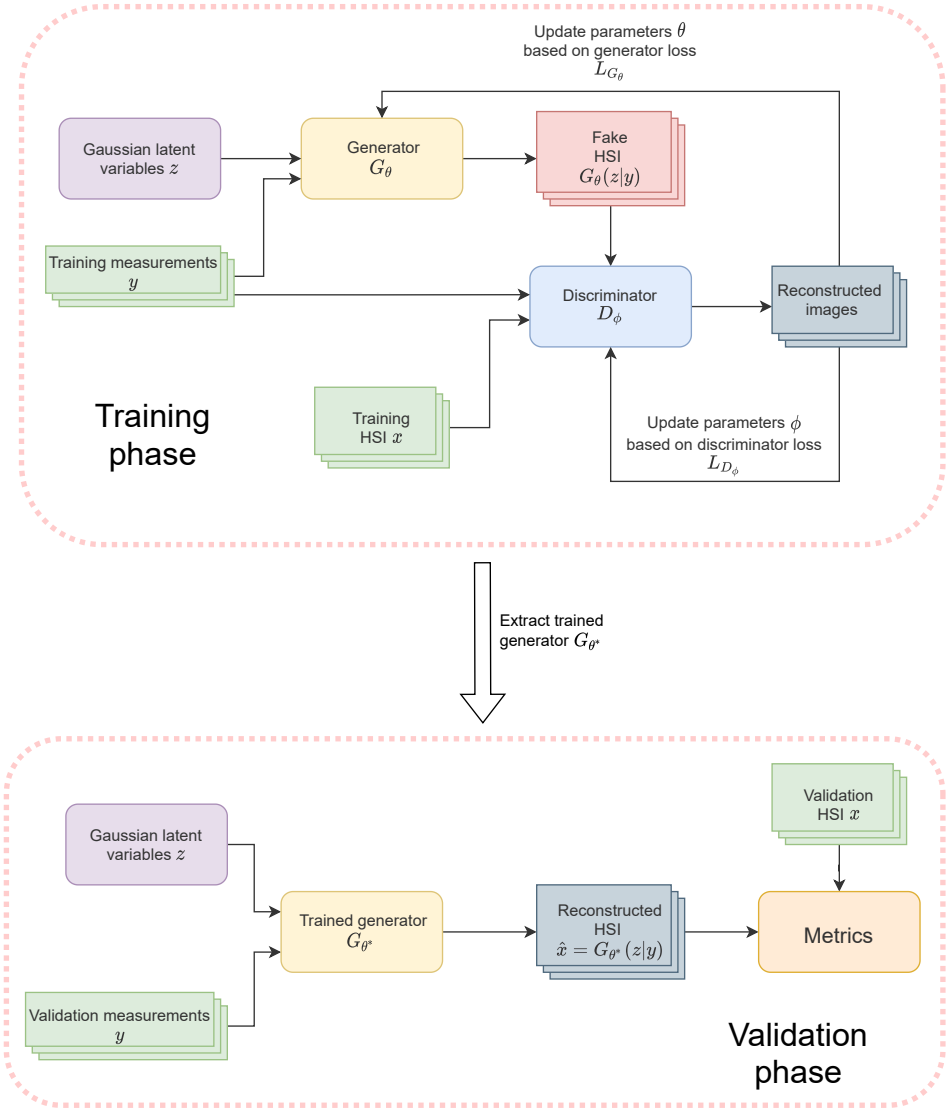


Figure 3.4: *Diagrams showing the training and validation phases.*

This latent variable optimization is presented in section 3.3.

Since the GAN utilized in this thesis is built upon the CGAN and BEGAN frameworks, it is henceforth dubbed CBEGAN. The following sections contains the CBEGAN objective and describes the training procedures. Details regarding the network architectures can be found in Appendix A.

3.2.1 Objective

The CBEGAN objective is similar to the BEGAN objective previously defined in (2.26), with the addition of the auxiliary information in form of measurements y being fed to the generator and discriminator

$$L_{D_\phi} = \max_{\phi}[R_{D_\phi(x|y)} - k_t R_{D_\phi(G_\theta(z|y))}] \tag{3.7a}$$

$$L_{G_\theta} = \min_{\theta} R_{D_\phi(G_\theta(z|y))} \tag{3.7b}$$

$$k_{t+1} = k_t + \lambda_k(\gamma R_{D_\phi(x|y)} - R_{D_\phi(G_\theta(z|y))}). \tag{3.7c}$$

During training of CBEGAN on the dataset allowing mix pixels, consistent mode collapse occured. Inspired by the work done by Chang et al. [19], an addition to the discriminator loss function in (3.7a) called the latent space constraint loss, L_c was introduced. The occurrence of mode collapse whilst using the BEGAN objective implies that all latent vectors z are encoded similarly in the discriminator. The L_c-loss puts a constraint on the norm between the input latent vector z and the output from the discriminator encoder $Enc(G_\theta(z|y))$, thus preventing all the different latent vectors z from being encoded similarly. The updated discriminator loss function including the new L_c-loss is given by

$$L_{D_\phi} = \max_{\phi}[R_{D_\phi(x|y)} - k_t R_{D_\phi(G_\theta(z|y))} + \alpha \cdot L_c] \tag{3.8a}$$

$$L_c = \|z - Enc(G_\theta(z|y))\|, \tag{3.8b}$$

where α is a parameter used for setting the relative importance of L_c as compared to the other terms of the discriminator loss.

3.2.2 Training

The training of the CBEGAN was conducted via a custom written training loop in Tensorflow, utilizing some parts of the Keras API. The network parameters (θ, ϕ) of the generator and discriminator respectively, were updated using the ADAM algorithm [20] configured by using a learning rate of $\eta = 0.0001$ and the Keras standard values for the hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The losses for the parameter updates were calculated using the CBEGAN objective previously defined in (3.7). The used batch size was 64, the CBEGAN objective diversity constant previously defined in (2.25) was set to $\gamma = 0.5$ and the proportionality constant defined in (2.21) was initialized to zero, $k_0 = 0$. All parameters and their respective values are listed in Table 3.1.

The datasets was normalized to $[-1, 1]$ to balance the training and in conjunction with using the hyperbolic tangent as the output layer activation function. Fifteen percent of the datasets was allocated as validation data, leading to the training data containing 2550 HSI, i.e., 163200 input rows, and the validation data containing 450 HSI i.e. 28800 input rows.

Training metrics such as loss values and the BEGAN convergence measure $\mathcal{M}$ defined in (2.27) were continuously logged using Tensorboard, allowing for monitoring of the training process and early termination in cases of non-satisfactory loss decrease. Additionally, a generated fake image was logged after the completion of each epoch, allowing for further user monitoring of the training process by visual inspection. This proved useful in cases where the losses were continuously decreasing, but the generated images were not in parity with the training data.

Table 3.1: *The used training parameters.*

Parameter	Value
η	0.0001
β_1	0.9
β_2	0.999
γ	0.5
k_0	0
Batch size	64
α	0.1

3.3 Latent Variable Optimization

Given an unknown HSI x^* we wish to reconstruct, and its corresponding available compressed measurement $y = Fx^*$, a generator G_{θ^*} obtained via training of CBEGAN produces a reconstruction $\hat{x}$ by

$$\hat{x} = G_{\theta^*}(z|y). \tag{3.9}$$

However, since the latent variables z are sampled from the traversable latent space $\mathcal{Z}$, there is an opportunity to fine tune the latent variables against some measure depending on the output of the generator.

The goal of the latent variable optimization is to find the optimal latent variables z^* such that the measurement error is minimized

$$z^* = \underset{z}{\operatorname{argmin}} \|y - FG_{\theta^*}(z|y)\|. \tag{3.10}$$

The optimal latent variables z^* are then taken to produce and improved reconstructed HSI $\hat{x}$ by

$$\hat{x} = G_{\theta^*}(z^*|y). \tag{3.11}$$

Due to the measurement matrix F used for collecting the measurements not having full rank, the implication in (3.2) does not hold. However, F is very close to full rank, and empirically the results showed that minimizing the measurement error in (3.10) and obtaining the optimal latent variables z^*, leads to the reconstruction error becoming smaller

$$\|x^* - G_\theta(z^*|y)\| < \|x^* - G_\theta(z|y)\|. \tag{3.12}$$

Since a major goal of the reconstruction process was for it to be fast-paced, an optimization algorithm with fast convergence was desired. According to Nocedal and Wright [21], the Gauss-Newton method is a good candidate, because of it not requiring costly calculations of Hessian matrices. Hence, the Gauss-Newton method was chosen for carrying out the latent variable optimization.

Figure 3.5 contains a block diagram illustrating the latent variable optimization procedure.

3.4 Final Reconstruction Algorithm

Since the HSI x are 3-dimensional signals of dimension $(i, j, \lambda) = (64, 64, 512)$, reconstructing entire HSI simultaneously was not feasible due to hardware limitations. Instead, the reconstruction is done row-wise by reconstructing the rows x^i one at a time in order, and later puzzled together to form the complete reconstructed HSI x.

Combining the learned prior from the trained generator G_{θ^*} with the Gauss-Newton method for fine-tuning the latent variables z, yields the algorithm used for reconstruction of HSI found in Algorithm 1.

The reconstruction algorithm is written using Tensorflow, utilizing its framework for easy and quick calculations of the Jacobians with respect to the residuals, and allowing for easy access to the generator during optimization.

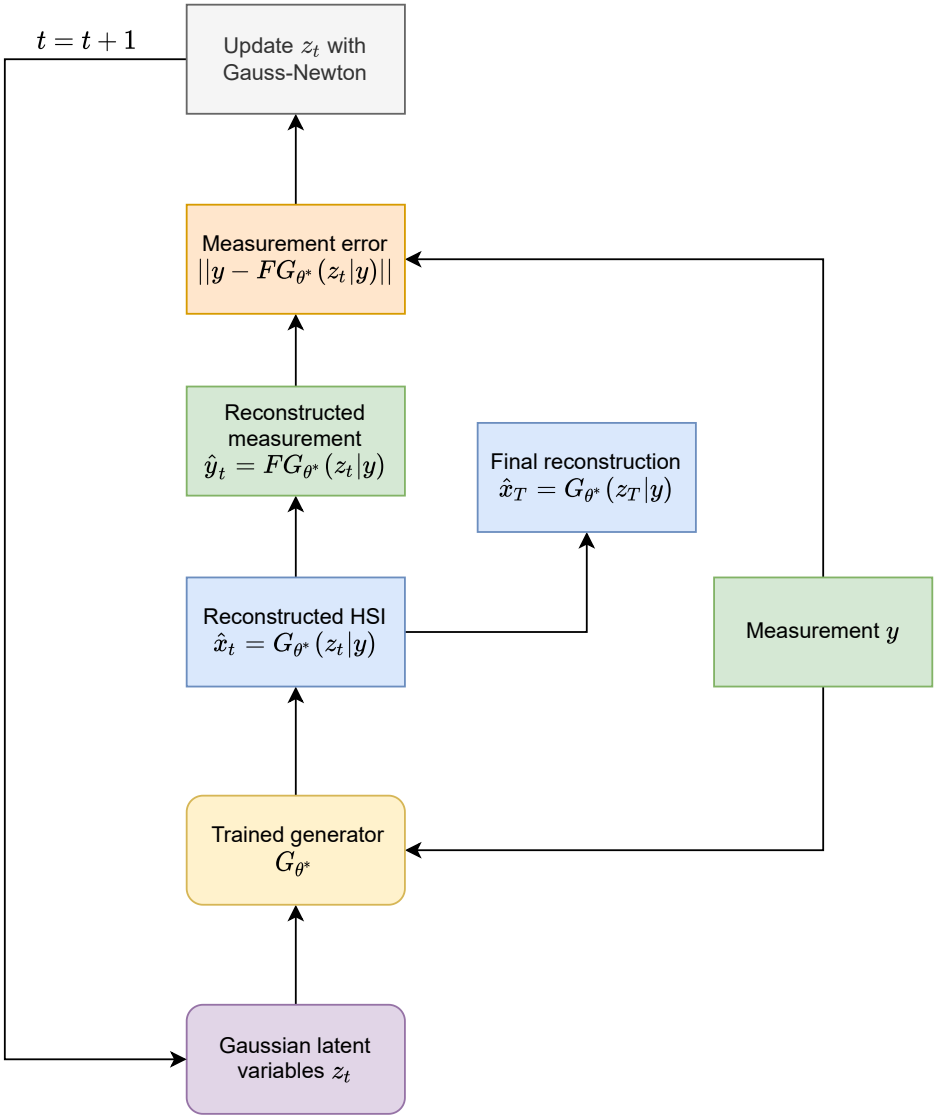


Figure 3.5: *Diagram illustrating the latent variable optimization.*

Algorithm 1: Hyperspectral image reconstruction using CBEGAN and the Gauss-Newton optimization method.

Input: Measurement matrix F, Trained generator G_{θ^*}, number of iterations T, true row measurements y^i;
Initialize: $\hat{x} = 0$;
for *i = 1 to 64* **do**
  Sample: $z_0 \sim N(0, 1)$;
  for *t = 0 to T-1* **do**
    $\hat{x}_t^i = G_{\theta^*}(z_t|y^i)$;
    $\hat{y}_t^i = F\hat{x}_t^i$;
    Loss = MeanSquaredError(y^i, $\hat{y}_t^i$) ;
    J = Jacobian(Loss, z_t) ;
    $z_{t+1} = z_t - (J^T J)^{-1} J^T \cdot \text{Loss}$;
  end
  $\hat{x}^i = G_{\theta^*}(z_T|y^i)$;
end
Output: Reconstructed hyperspectral image $\hat{x}$

4

Results

To evaluate the performance of the GAN-based HSI reconstruction method, the 450 HSI in the respective validation sets were benchmarked against reconstruction with TVAL3+, both in the case of no initial guess and the case where initial guesses from CBEGAN were provided. Additionally, the CBEGAN-Gauss-Newton method was benchmarked against the method presented in [3]. Due to the primary goal being fast reconstruction, the algorithms were compared by fixating the allowed run times of CBEGAN-Gauss-Newton and TVAL3+, and evaluating the reconstruction precision metrics. Since the method in [3] does not contain any fine tuning optimization steps, its reconstruction time can not be varied.

Additionally for completeness, TVAL3 was allowed to run until a convergence criterion was met. The TVAL3 convergence criterion checks whether the relative error of the reconstructed HSI $\hat{x}$ between subsequent iterations are less than some tolerance ϵ. That is, if

$$\frac{\|\hat{x}_{t-1} - \hat{x}_t\|_2}{\|\hat{x}_{t-1}\|_2} < \epsilon, \tag{4.1}$$

where $\epsilon = 10^{-6}$, and other algorithm parameters, was chosen according to their default values as specified by the authors [10]. The mean reconstruction time was noted and the result of CBEGAN-Gauss-Newton was evaluated at that same runtime.

Two metrics were used to evaluate the quality of reconstructed HSI. The first performance metric is the mean absolute error (MAE), defined as

$$MAE(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^{N} |x_i - \hat{x}_i|, \tag{4.2}$$

where N denotes the total number of voxels in the HSI. The second used performance metric is the cosine similarity, given by

$$S_C(x, \hat{x}) = \frac{x \cdot \hat{x}}{\|x\|_2 \|\hat{x}\|_2}. \tag{4.3}$$

The cosine similarity metric looks at the angle between two given vectors, and output a similarity value between zero and one. A value of zero indicates that the two vectors are orthogonal, and a value of one indicated that they are paralell. Hence, the absolute magnitudes are not affecting the similarity score, as is the case with MAE.

4.1 Robustness Test

A small test was conducted to test the robustness of the CBEGAN-Gauss-Newton reconstruction. This was done to verify that the initialization of the latent variables z was not severely affecting the quality of the reconstructions. Due to the validation sets containing 450 HSI, the collection of reconstruction results would prove quite time-consuming if robustness verification was to be conducted for all runs with different fixed execution times.

The robustness test was conducted by running CBEGAN-Gauss-Newton reconstruction on all 450 HSI in the validation set five consecutive times. The number of optimization iterations was fixed such that reconstruction of a HSI took ten seconds. The mean MAE of the reconstructed HSI was collected for each of the five runs, and the result can be seen in Figure 4.1.

As illustrated in Figure 4.1, the mean MAE score obtained after reconstruction of HSI in the validation set are consistent up to four decimals for all five independent runs, which indicates that the CBEGAN-Gauss-Newton is indeed robust. To save time, the collection of results for other fixed reconstructions times were only done once, with the knowledge of the probability of obtaining a non representative "lucky" run was small.

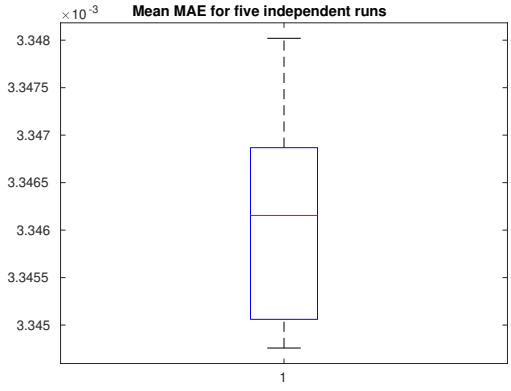
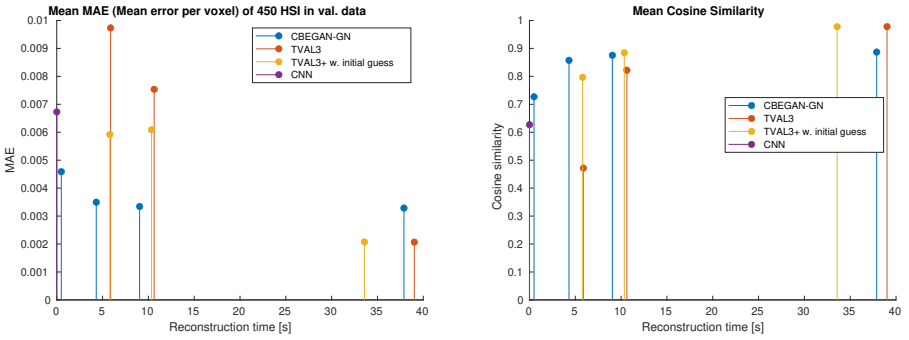


Figure 4.1: *Box chart showing mean MAE scores.*

4.2 Dataset not Allowing Mixed Substances

Figure 4.2 shows the metrics after reconstruction of all HSI in the validation set where only one Raman spectrum per pixel was allowed. Tables 4.1 to 4.4 contain all resulting numerical values of the chosen metrics, for approximate fixed run times of five and ten seconds, as well for the case when TVAL3+ was allowed to run until convergence. Reconstruction with CBEGAN only, i.e. without latent variable optimization with Gauss-Newton, and the CNN in [3] was also evaluated. Bold text is used to indicate the best result for the metrics.



(a) *(b)*

Figure 4.2: *Plots visualising how the reconstruction methods perform at different runtimes. The big gap in time is due to TVAL3+ taking approximately 40 seconds to converge.*

Table 4.1: *Reconstruction metrics of reconstruction with CBEGAN and CNN in [3].*

Method	Mean runtime [s]	Mean MAE	Mean S_C
CBEGAN	0.50	**0.004589**	**0.7268**
CNN in [3]	0.06	0.006727	0.6270

Table 4.2: *Reconstruction metrics of investigated methods where runtime was fixed to approx. 5s.*

Method	Mean runtime [s]	Mean MAE	Mean S_C
CBEGAN-Gauss-Newton	4.97	**0.003496**	**0.8574**
TVAL3+	5.47	0.009733	0.4717
TVAL3+ w. initial guess	5.50	0.005923	0.7966

The resulting reconstruction metrics show that for fixed low reconstruction times, CBEGAN-Gauss-Newton reconstruction is performing better than TVAL3+, with

Table 4.3: *Reconstruction metrics of investigated methods where runtime was fixed to approx. 10s.*

Method	Mean runtime [s]	Mean MAE	Mean S_C
CBEGAN-Gauss-Newton	10.18	**0.003345**	0.8754
TVAL3+	9.92	0.007558	0.8220
TVAL3+ w. initial guess	9.87	0.006079	**0.8845**

Table 4.4: *Reconstruction metrics of investigated methods where TVAL3+ ran until convergence.*

Method	Mean runtime [s]	Mean MAE	Mean S_C
CBEGAN-Gauss-Newton	39.13	0.003286	0.8869
TVAL3+	39.95	0.002072	**0.9779**
TVAL3+ w. initial guess	35.37	**0.002047**	0.9778

or without an initial guess, in terms of both metrics. The case where no optimization steps with Gauss-Newton are executed, i.e., the reconstruction is taken via inference of CBEGAN, is still better in terms of the precision metrics than TVAL3+ at fixed reconstruction times of five and ten seconds.

The method in [3] is by far the fastest reconstruction method. However, the quality of the reconstructed HSI is worse when compared to reconstruction with CBEGAN without latent optimization in terms of both metrics.

The results does also show that providing an initial guess to TVAL3+ significantly improves the quality of the output reconstructions for the lower fixed runtimes. However, CBEGAN-Gauss-Newton still outperforms TVAL3+ with initial guess in all metrics except for the cosine similarity at 10 second runtime. Providing an initial guess to TVAL3+ does also reduce the time to convergence with approximately 5 seconds.

Given adequate time, TVAL3+ will outperform CBEGAN-GN eventually. The improvements on reconstruction quality induced by the latent variable optimization with GN diminish fairly quickly. After five seconds of CBEGAN-Gauss-Newton the reconstruction quality barely improves and the method's performance clearly saturates.

Figure 4.3 shows magnitude images of a ground truth HSI and the reconstructed ditto with fixed runtimes of five seconds, and Figures 4.4 and 4.5 shows individual spectra in three different pixels. Clearly, the reconstruction achieved with CBEGAN-Gauss-Newton outperforms both versions of TVAL3+. The magnitude images show that CBEGAN-Gauss-Newton is superior in capturing the spatial structure of the ground truth HSI. Comparing the spectrum plots in we see that CBEGAN-Gauss-Newton reconstructs most peaks, and the single peak that it did

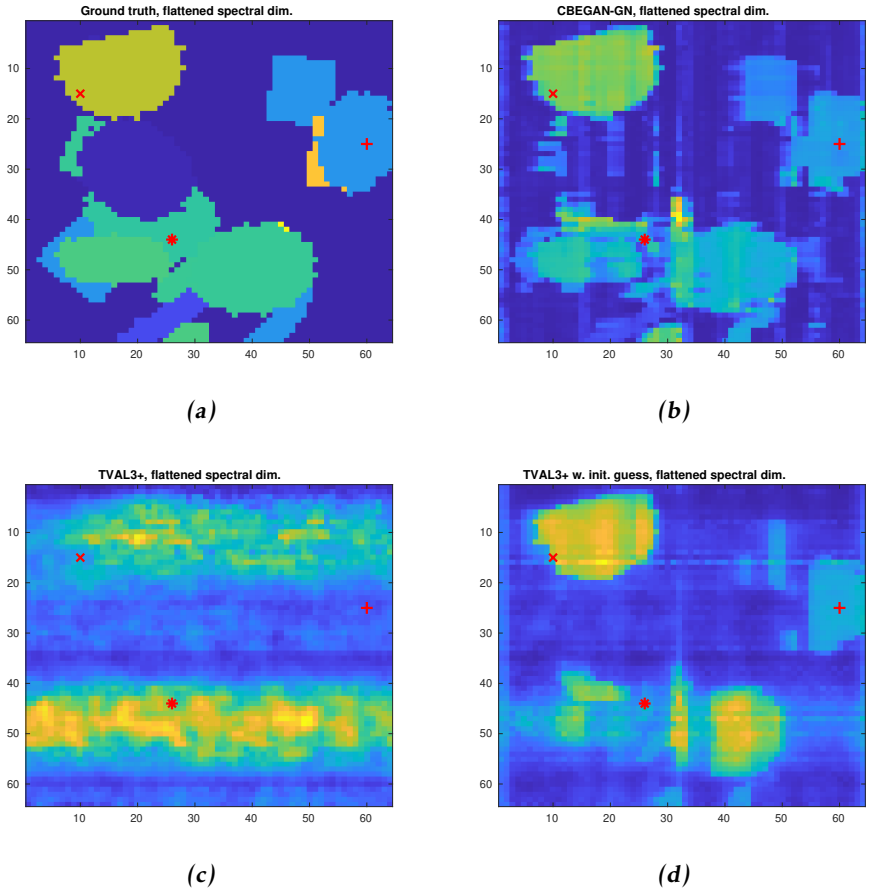


Figure 4.3: *Magnitude pictures of ground truth and reconstructed HSI using the different methods with an allowed runtime of five seconds.*

not reconstruct correctly, neither versions of TVAL3+ could reconstruct either. Compared to the spectrums of both versions of TVAL3+, the reconstructed spectra from CBEGAN-Gauss-Newton does not contain any ghost peaks.

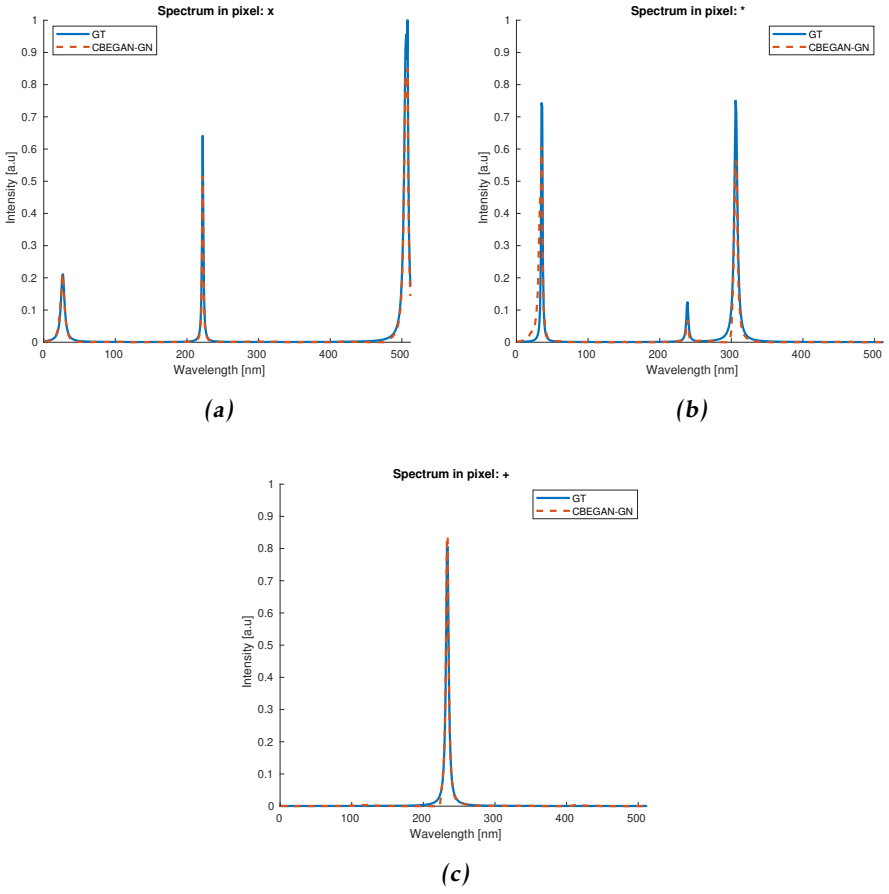


Figure 4.4: *Ground truth and reconstructed spectra.*

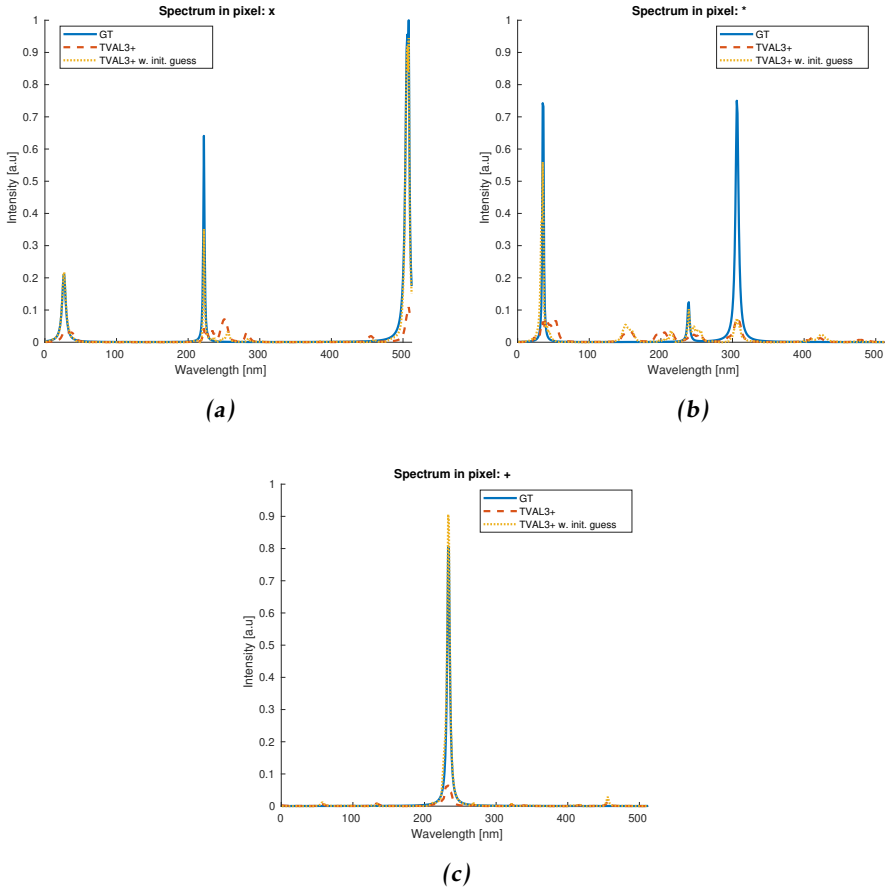
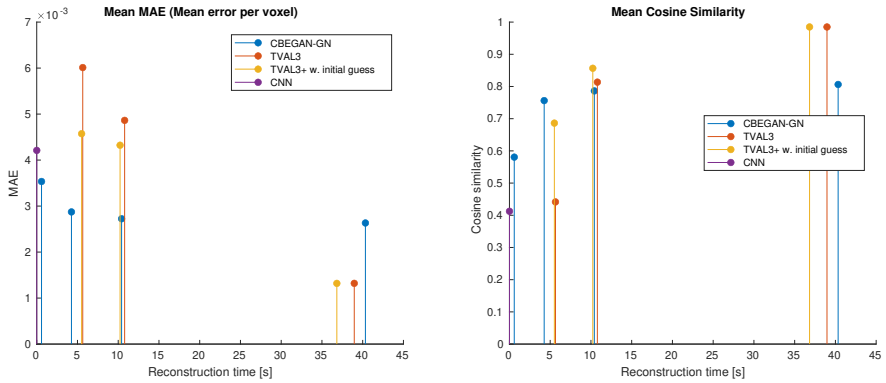


Figure 4.5: *Ground truth and reconstructed spectra.*

4.3 Dataset Allowing Mixed Substances

Figure 4.6 shows the metrics after reconstruction of all HSI in the validation set where several Raman spectrum per pixel was allowed. Tables 4.5 to 4.8 contain all resulting numerical values of the chosen metrics, for differing fixed runtimes. Bold text is used to indicate the best result for the metrics.



(a) *(b)*

Figure 4.6: *Plots visualising how the reconstruction methods perform at different runtimes.*

Table 4.5: *Reconstruction metrics of inference with CBEGAN and CNN in [3].*

Method	Mean runtime [s]	Mean MAE	Mean S_C
CBEGAN	0.55	**0.003535**	**0.5806**
CNN in [3]	0.05	0.004211	0.4124

Table 4.6: *Reconstruction metrics of investigated methods where runtime was fixed to approx. 5s.*

Method	Mean runtime [s]	Mean MAE	Mean S_C
CBEGAN-Gauss-Newton	4.26	**0.002873**	**0.7562**
TVAL3+	5.65	0.006012	0.4415
TVAL3+ w. initial guess	5.51	0.004573	0.6863

The resulting reconstruction metrics show similar results as for the previous dataset. Reconstruction with CBEGAN-Gauss-Newton produces a lower MAE and higher cosine similarity scores on average for low allowed runtimes. Given an allowed runtime of ten seconds, TVAL3+ produces higher cosine similarity scores than CBEGAN-GN, and in the case of allowing TVAL3+ to run until convergence,

Table 4.7: *Reconstruction metrics of investigated methods where runtime was fixed to approx. 10s.*

Method	Mean runtime [s]	Mean MAE	Mean S_C
CBEGAN-Gauss-Newton	10.42	**0.002725**	0.7863
TVAL3+	10.79	0.004864	0.8135
TVAL3+ w. initial guess	10.23	0.004323	**0.8566**

Table 4.8: *Reconstruction metrics of investigated methods where TVAL3+ ran until convergence.*

Method	Mean runtime [s]	Mean MAE	Mean S_C
CBEGAN-Gauss-Newton	40.33	0.002633	0.8063
TVAL3+	38.97	0.001321	0.9848
TVAL3+ w. initial guess	36.83	**0.001320**	**0.9849**

we acquire cosine similarity scores close to one.

We see a similar trend regarding the performance of CBEGAN-Gauss-Newton as for the previous dataset, namely that the performance saturates rather quickly, and that not a lot of performance gain is added by letting CBEGAN-Gauss-Newton run for more than 5 seconds.

Figure 4.7 shows magnitude images of a ground truth HSI and the reconstructed ditto with fixed runtimes of five seconds, and figures 4.8 and 4.9 shows individual spectra in three different pixels. It is noticeable that CBEGAN-Gauss-Newton is able to capture the spatial structure of the HSI than the two versions of TVAL3+. Given an initial guess from CBEGAN, the spatial structure in the reconstruction from TVAL3+ is significantly improved as compared to when no initial guess is provided. However still, the content in the magnitude pictures from TVAL3+ still looks a lot more smeared out as compared to CBEGAN-Gauss-Newton. The effect of this smearing can be seen in Figure 4.9a, where TVAL3+ has introduced spectral content, in a voxel with basically zero ground truth spectrum. This is probably due to TVAL3+ recognizing that there exists spectral content somewhere on the row, as can be seen in 4.7a. Since the objective is to minimize TV, the algorithm attempts to spread the spectral content to neighboring voxels creating a smooth surface containing small discrete gradients.

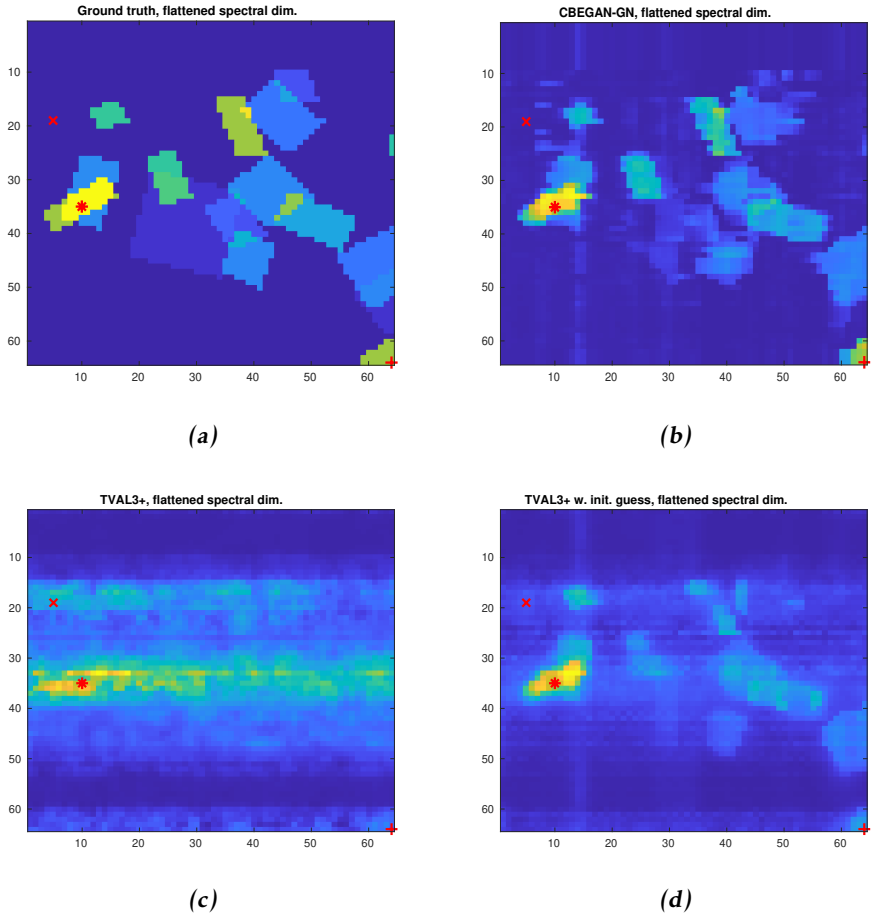


Figure 4.7: *Magnitude pictures of ground truth and reconstructed HSI using the different methods with approximate runtimes of five seconds.*

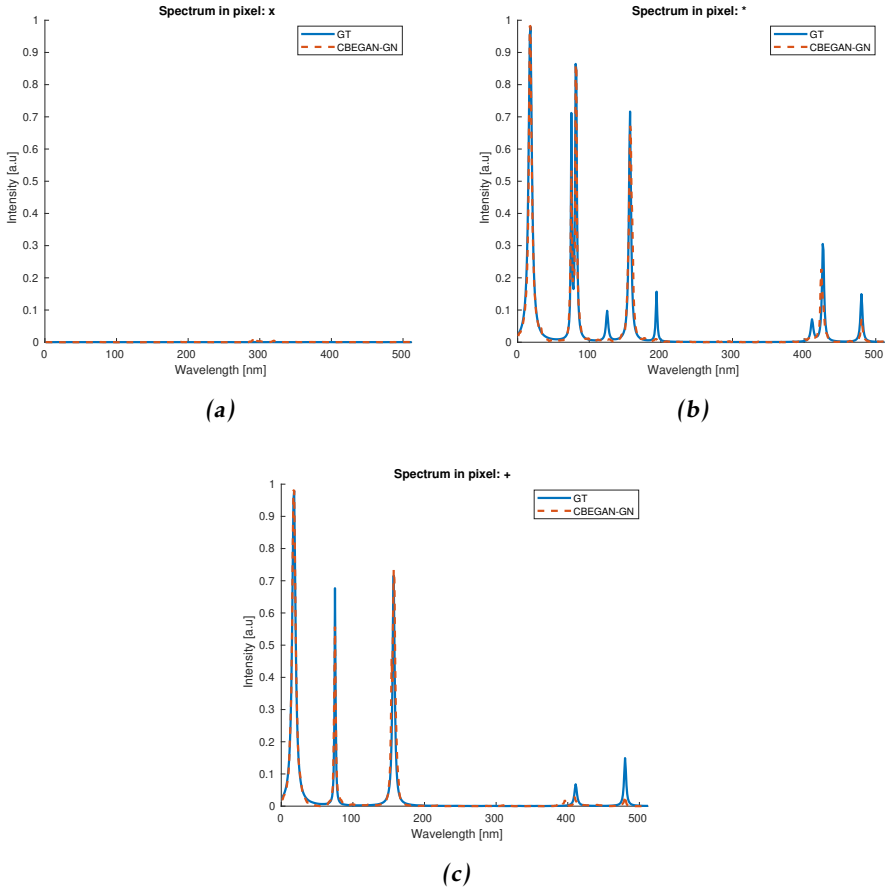


Figure 4.8: *Ground truth and reconstructed spectra from CBEGAN-GN*

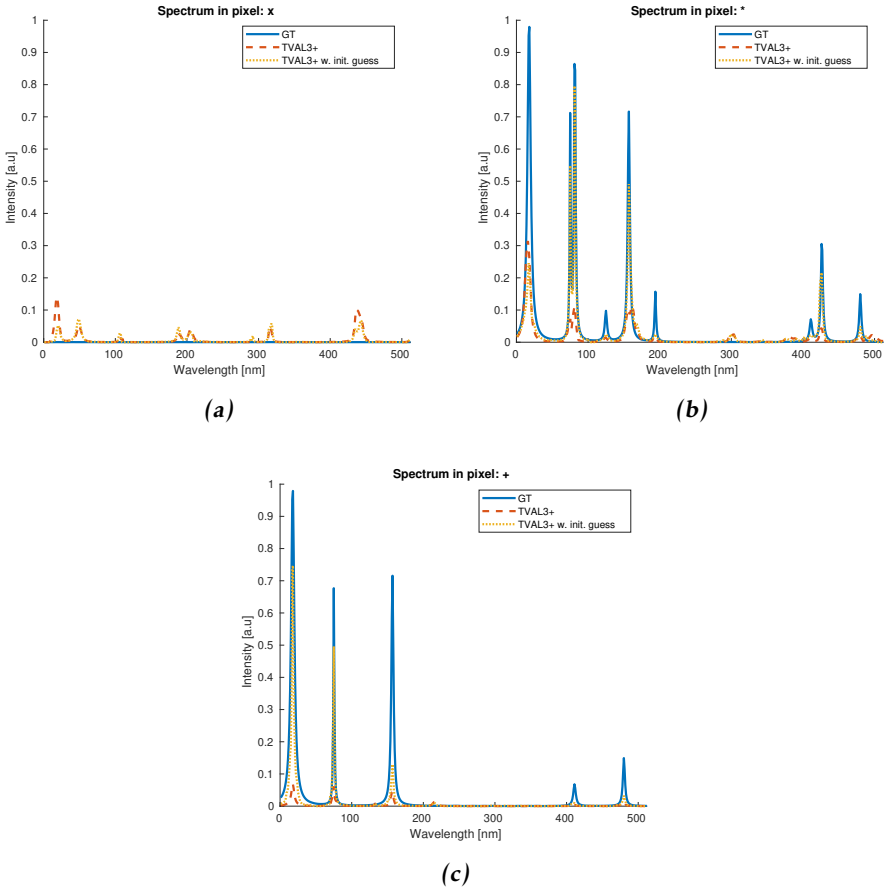


Figure 4.9: *Ground truth and reconstructed spectra from* TVAL3+.

5

Analysis and Conclusions

The work in this thesis shows that generative model based reconstruction method is indeed viable for reconstruction of HSI when fast reconstruction is of great importance. The CBEGAN-Gauss-Newton reconstruction method performs better than TVAL3+ in terms of reconstruction quality when using low reconstruction times. But the performance of the method saturates rather quickly, and therefore TVAL3+ is preferable if time is not the most essential parameter. As previously mentioned in the introduction, HSI can be measured in a matter of seconds using CASSI, and with CBEGAN-Gauss-Newton, good reconstructions of these HSI are now available in a matter of seconds as well.

The results show that given data samples with mixed substances, CBEGAN-Gauss-Newton is able to reconstruct the HSI fairly well. However, comparing the mean cosine similarity scores of CBEGAN-Gauss-Newton for the different data sets show that the reconstructions of data samples with mixed substances are not as successful as compared to the case when the substances are not mixed. This is probably the case because of the allowance of mixed substances simply makes the data distribution more complex, making it harder for the CBEGAN to capture it. Since similar CBEGAN with the same number of parameters were trained on the two data sets, it is probable that the CBEGAN had more difficulty learning the distribution of samples with mixed substances. A more flexible GAN, trained using data sets with higher cardinality than available in this thesis, should improve the reconstruction results.

The results does also show that providing TVAL3+ with initial guesses in form of output images from the trained CBEGAN does indeed improve upon the reconstruction quality. For fixed low algorithm run times, the quality metrics are significantly lower compared to reconstruction without initial guesses. Addition-

ally, when letting TVAL3+ run until convergence, the convergence time is lower when providing an initial guess.

As previously discussed, the performance of CBEGAN-Gauss-Newton saturates rather quickly. One theory as to why this is the case could be the fact that the number of latent variables are significantly lower than the number of elements in the measurements. Since both of these are provided as inputs to the GAN, it is possible that the latent variables simply does not play a big enough part in shaping the distribution of the weights in the generator. Possible future work would be to investigate how changes of the ratio between latent variables and measurements could affect the reconstructed HSI. Increasing the number of latent variables would allow the optimization over the latent variables to play a bigger part, and the decrease in the number of measurements means that less time has to be allocated into collecting those measurements.

The generator from the trained GAN provides a learned prior on the training samples, where different types of structures can be captured. In the CS literature, sparsity priors are most commonly assumed, such as in TVAL3+. In this case, the HSI are quite sparse, hence TVAL3+ performs well. A hypothesis is that CBEGAN-Gauss-Newton reconstruction should also perform fairly well when the data is not sparse. Additionally, the learned prior could provide insight into other properties of the data such as spatial context. Since the GAN outputs rows of the HSI the correlation between neighboring pixels row wise should be captured.

The data sets used in this thesis is unfortunately not totally realistic, due to the fact that any substances of the sixteen in total could neighbor each other or be mixed. In practice, it is more probable that some substances lie near to each other spatially, and other should never lie near each other. More realistic data sets would allow the learned prior to capture which substances typically lie near each other spatially, and which substances that do not. Even better, would be if the GAN was fed a couple of rows at a time, a small window of the HSI. This would allow for capturing of the correlation in both spatial directions. If the windows overlap somewhat, the learned prior would contain insight into what substances typically neighbor each other in both the vertical and horizontal directions.

Even though the focus of this thesis was not classification of Raman spectra per se, but rather reconstruction of HSI, it would be interesting to think about these two goals jointly whilst training the GAN. In classification of Raman spectra, there are two different aspects involved:

1. The location of the peaks of the reconstructed Raman spectra should coincide with the true Raman spectra.

2. In the case of the Raman spectra containing several peaks with different heights, the reconstructed peaks should be of the same relative height to each other as the true peak heights are relative to each other.

Since the absolute intensity of reconstructed peaks are not that important, but their relative heights are, cosine similarity could be a great candidate to involve in the GAN objective in a clever way.

Since it has been shown that TVAL3+ produces high quality reconstructions of HSI given adequate time, it would be interesting to investigate whether parts of which TVAL3+ can be incorporated into the GAN objective with success. One example could be to investigate whether a TV regularization term could be useful in either the generator loss, discriminator loss, or both. A TV regularization term in the generator would incentivise generation of synthetic images sparse in the gradient basis, and the incorporation of a TV term in the discriminator would provide an additional way for differentiating between real and synthetic images.

Appendix

A

CBEGAN Network Architectures

A.1 Generator Architecture

The generator network aims to produce a row of an HSI $\hat{x}^i$ given a measurement of the row y^i and Gaussian distributed latent variables z as inputs. The output $\hat{x}^i$ is of dimension $(i, j) = (64, 512)$, so to output an entire HSI $\hat{x}$, 64 outputs $\hat{x}^i$, $i \in [1, 64]$ from the generator are required

The generator takes the latent variables z and measurements y^i as inputs, and is built up by one fully connected layer followed by five transposed 2D convolutional layers. The fully connected layer reshapes the data into size (16, 2, 256). The convolutional filter kernels are of size (3, 3) and stride (2, 2), resulting in each convolutional layer upsampling the input data by a factor of two, until reaching the desired size of (512, 64). All layers in the generator are followed by batch normalization and Leaky ReLU activation, except for the last layer which uses the hyperbolic tangent as activation function.

The original BEGAN paper [18], proposes a generator architecture containing two convolutional layers prior to each upsampling, as well as skip connections from the input to the output of each convolutional layer. However, due to hardware limitations, the architecture was not feasible, and thus the scaled down DCGAN inspired variant found in Table A.1 was used. An illustration of the generator architecture can be found in Figure A.1.

Table A.1: *The generator architecture used in CBEGAN.*

Name	Layer	Filters	Output dimension
Input z	Input	-	(100,)
Input y	Input	-	(2300,)
Concatenate	Concatenation(z, y)	-	(2400,)
FC1	Dense-BN-LeakyReLU	-	(16, 2, 256)
Upsamp. 1	Conv2DTrans-BN-LeakyReLU	256	(32, 4, 256)
Upsamp. 2	Conv2DTrans-BN-LeakyReLU	128	(64, 8, 128)
Upsamp. 3	Conv2DTrans-BN-LeakyReLU	64	(128, 16, 64)
Upsamp. 4	Conv2DTrans-BN-LeakyReLU	32	(256, 32, 32)
Output	Conv2DTrans-tanh	1	(512, 64, 1)

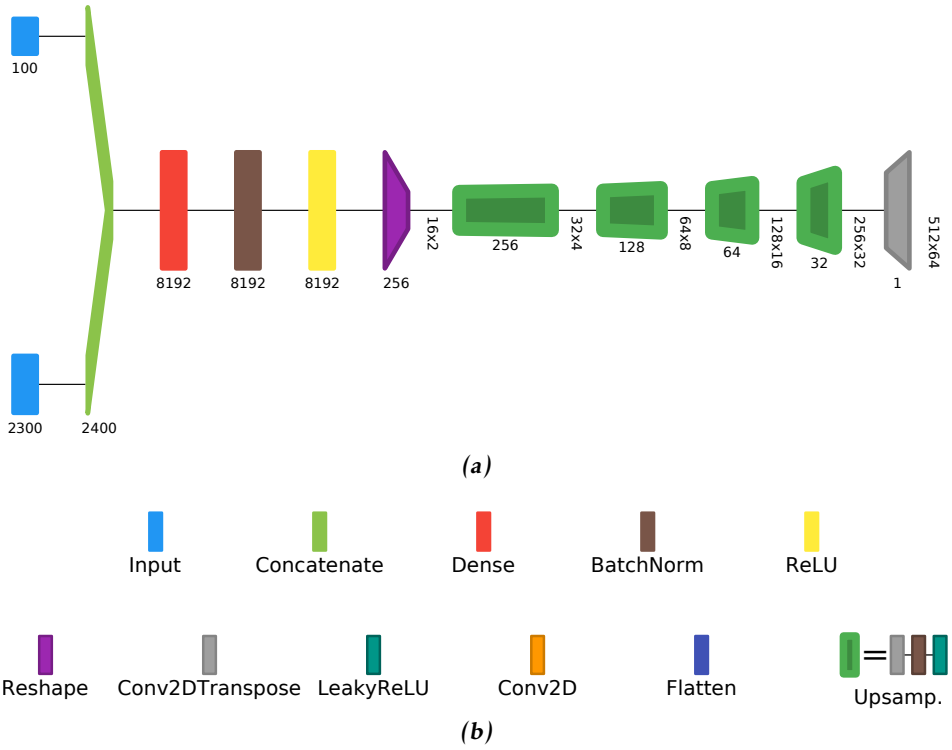


Figure A.1: *An illustration of the generator architecture in CBEGAN. Figures created by utilizing [22].*

A.2 Discriminator Architecture

As previously presented in Section 2.5.3, the discriminator used in the BEGAN framework is an autoencoder. The input of the discriminator is a real row of an HSI, or a fake row outputted from the generator, and the output of the discriminator is a reconstruction of the input, leading to the inputs and outputs having the same dimension of $(j, \lambda) = (64, 512)$.

The complete architecture of the discriminator is based on the architecture of the generator, where the decoding part of the discriminator is identical to the generator. The encoding part of the essentially mirrors the decoder part. The transposed convolutional layers are replaces with regular convolutional layers, resulting in the input being downsampled instead of upsampled.

The encoder input is, as previously mentioned, a real row from the training set, or a fake row from the generator, and it encodes the input into a a vector of the same dimensionality as the latent variable vector

$$Enc : [-1, 1]^{64x512} \rightarrow \mathbb{R}^{dim(z)} \tag{A.1}$$

The output from the encoder is then treated exactly like the input variables z in the generator. It is concatenated with a measurement y and then decoded to an output of the same dimensionality as the input

$$Dec : \mathbb{R}^{dim(z)+dim(y)} \rightarrow [-1, 1]^{64x512} \tag{A.2}$$

By trial and error, the utilization of dropout layers in the discriminator resulted in more real looking synthetic data output from the generator. The introduction of dropout layers in the decoder leads to the discriminator acting like a denoising autoencoder according to Goodfellow, Bengio, and Courville [14]. The encodings of denoising autoencoders are particularly useful when attempting to learn the underlying distribution of the input data, rather than learning the identity function copying the input to the output.

Table A.2 lists all layers used in the discriminator architecture, and Figure A.2 provides an illustration.

Table A.2: *The discriminator architecture used in* CBEGAN.

Name	Layer	Filters	Output dimension
Input x	Input	-	(512, 64, 1)
Downsamp. 1	Conv2D-LeakyReLU-Dropout	32	(256, 32, 32)
Downsamp. 2	Conv2D-LeakyReLU-Dropout	64	(128, 16, 64)
Downsamp. 3	Conv2D-LeakyReLU-Dropout	128	(64, 8, 128)
Downsamp. 4	Conv2D-LeakyReLU-Dropout	256	(32, 4, 256)
Downsamp. 5	Conv2D-LeakyReLU-Dropout	256	(16, 2, 256)
Flatten1	Flatten	-	$(16 \cdot 2 \cdot 256, 1)$
FC1	Dense	-	(100,)
Input y	Input	-	(2300,)
Concat1	Concatenation($FC1, y$)	-	(2400,)
FC2	Dense-BN-LeakyReLU	-	(16, 2, 256)
Upsamp. 1	Conv2DTrans-BN-LeakyReLU	256	(32, 4, 256)
Upsamp. 2	Conv2DTrans-BN-LeakyReLU	128	(64, 8, 128)
Upsamp. 3	Conv2DTrans-BN-LeakyReLU	64	(128, 16, 64)
Upsamp. 4	Conv2DTrans-BN-LeakyReLU	32	(256, 32, 32)
Output	Conv2DTrans-tanh	1	(512, 64, 1)

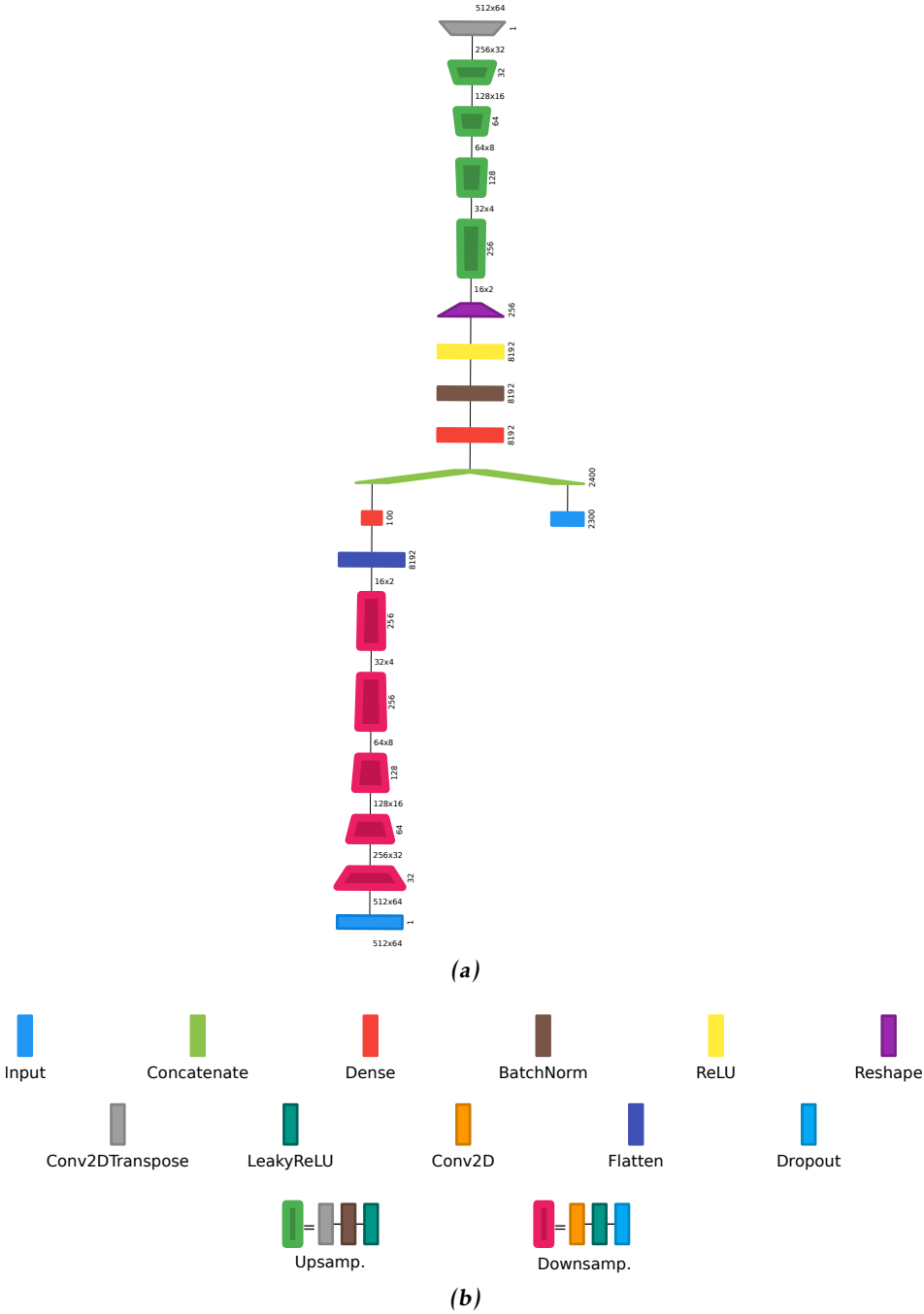


Figure A.2: *An illustration of the discriminator architecture in CBEGAN. Figures created by utilizing [22].*

Bibliography

[1] A. Brorsson, M. Nordberg, D. Gustafsson, and H. Östmark, "Raman shift coded aperture snapshot spectral imager," presented at Swedish Symposium on Image Analysis (SSBA), March 2021.

[2] M. Nordberg, L. Hollmann, L. Landström, and D. Gustafsson, "Reconstruction of compressed sensing Raman imaging using machine learning," in *Electro-Optical and Infrared Systems: Technology and Applications XVIII and Electro-Optical Remote Sensing XV*, vol. 11866, International Society for Optics and Photonics. SPIE, 2021, pp. 46 – 53.

[3] A. Brorsson, M. Nordberg, and D. Gustafsson, "Reconstruction of CASSI-Raman Images with Machine-Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 4388–4395.

[4] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, "Compressed Sensing using Generative Models," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, Aug. 2017, pp. 537–546.

[5] Y. Wu, M. Rosca, and T. P. Lillicrap, "Deep Compressed Sensing," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, ser. Proceedings of Machine Learning Research, vol. 97. Long Beach, CA, USA: PMLR, June 2019, pp. 6850–6860.

[6] J. Han, M. Kamber, and J. Pei, *Data Mining, Concepts and Techniques*, 3rd ed. Waltham, MA, USA: Elsevier/Morgan Kaufmann, 2012.

[7] P. Rostron and D. Gerber, "Raman Spectroscopy, a review," *International Journal of Engineering and Technical Research*, vol. 6, pp. 50–64, Sep. 2016.

[8] Z. Fang, Y. Tao, W. Wang, W. Zhang, L. Duan, Y. Liu, C. Yan, L. Qu, and C. Han, "Joint sparse representation and denoising method for Raman spectrum," *Journal of Raman Spectroscopy*, vol. 49, no. 12, pp. 1972–1977, 2018.

[9] I. Rish and G. Grabarnik, *Sparse Modeling: Theory, Algorithms, and Applications*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2014.

[10] C. Li, W. Yin, and Y. Zhang, "User's Guide for TVAL3: TV Minimization by Augmented Lagrangian and Alternating Direction Algorithms," 2010.

[11] A. Brorsson, "Compressive Sensing: Single Pixel SWIR Imaging of Natural Scenes," Master's thesis, Linköping University, Linköping, Sweden, 2018.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *Advances in Neural Information Processing Systems*, vol. 3, June 2014.

[13] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, San Juan, Puerto Rico, USA, May 2016.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[15] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. Lille, France: JMLR.org, July 2015, pp. 448–456.

[16] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," *CoRR*, vol. abs/1411.1784, 2014.

[17] K. Kim, J. H. Lee, and E. Yang, "Compressed Sensing via Measurement-Conditional Generative Models," *CoRR*, vol. abs/2007.00873, 2020.

[18] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary Equilibrium Generative Adversarial Networks," *CoRR*, vol. abs/1703.10717, 2017.

[19] C. Chang, C. H. Lin, C. Lee, D. Juan, W. Wei, and H. Chen, "Escaping from Collapsing Modes in a Constrained Space," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, ser. Lecture Notes in Computer Science, vol. 11211. Munich, Germany: Springer, Sep. 2018, pp. 212–227.

[20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, San Diego, CA, USA, May 2015.

[21] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[22] A. Bäuerle, C. van Onzenoodt, and T. Ropinski, "Net2Vis – A Visual Grammar for Automatically Generating Publication-Tailored CNN Architecture Visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 6, pp. 2980–2991, 2021.