FACULTY OF SCIENCE

UNIVERSITY OF COPENHAGEN

UNIVERSITY OF
COPENHAGEN

Master's Thesis in Physics
submitted by

ALINA SODE

2021

Alina Sode: *Automatic Detection of Foreign Objects in X-Ray Images*

This Master's Thesis has been carried out at

THE NIELS BOHR INSTITUTE, COPENHAGEN

under the supervision of

ASSISTANT PROFESSOR KENNETH SKOVHEDE
(Department of Computer Science, Copenhagen)

ACKNOWLEDGMENTS

First and foremost I wish to express my sincere appreciation to *Asst. Prof. Kenneth Skovhede* for taking on the role of being my supervisor. Thank you for supporting and guiding me through this road and until its end; and what a journey it has been.

I especially owe my deepest gratitude to *PhD fellow Carl-Johannes Johnsen* and *Research Assistant Aleksandar Topic* for their invaluable guidance, assistance and inputs during the work of this thesis. Without their persistent help and encouragement, the goal of this project would not have been realized.

Furthermore, I would also like to acknowledge the remaining professors, PhD fellows and master students in our joint office at the eScience research group at the Niels Bohr Institute for their continuous ideas for improvements, inspirations and discussions about this project — it is much appreciated. Additionally, I am thankful to all the individuals whom have proofread this document.

Moreover, I extend thanks to *Newtec Engineering A/S* for their collaboration and letting me borrow their area scintillator detector so I was able to work with high resolution X-ray images of customized samples. In particular, I would like to pay my special regards to *Bjarke Jørgensen, Head of Research*, for his correspondence and for scanning additional data for me in these trying COVID-19 pandemic times. This work would not have been possible without their assistance.

Copenhagen, May 2021

Alina Sode

ABSTRACT

In this thesis an adaptive and automated pipeline for static novelty detection of foreign objects and other defectives in food products using X-ray imaging is demonstrated. First, the fundamental principles underlying the interactions of radiation with matter are presented, whereas the primary focus is placed on the contrast mechanism in X-ray imaging. To gain the necessary knowledge required to understand feature extractions in images, deep learning with images and more specially convolutional neural networks are presented. The unsupervised convolutional autoencoder (CAE) and convolutional variational autoencoder (CVAE) networks are trained using only normal samples, and their abilities at reconstructing inputs are utilized to successfully distinguish between normal and anomalous samples. But, before data are fed to the neural networks it however needs to be prepared, and the field of computer vision provides many techniques to do so, such as contrast enchantment, noise reduction and image augmentations.

Collectively these frameworks lay the foundation for the proposed pipeline as summarized by three main steps: (1) training an unsupervised deep model able to reconstruct normal samples so precisely as possible; (2) computing statistical distributions based upon a chosen anomaly score and; (3) threshold selection. Various anomaly scores are examined and compared, whereas we learn that the particular choice of anomaly score has a large impact on the evaluation scores, and the anomaly score that works the best varies from dataset to dataset. Moreover, the anomaly ratio have also been showed to have an impact on the scores, as the precision decrease with decreasing ratio.

Identifying anomalous chocolate bar from normal ones are achieved with a top accuracy of 99% using a 50% anomaly ratio by utilizing the zero-mean normalized cross-correlation (ZNCC), operating with only 10 individual chocolate bars and 22 separate scans.

Using only 50 individual potatoes with 540 distinct scans, whereas 196 scans are of inserted needles and 156 scans of artificial created hollow hearts, we find that the models repeatedly have an easier time detecting needles compared to hollow hearts. Utilizing the structural similarity index measure (SSIM) and a 50% anomaly ratio yields the top accuracy of 89%.

Additionally, the novelty pipeline was tested using a different potato dataset consisting of only 45 unique perfect potatoes and 66 potatoes with natural hollow heart disease, also given a 50% anomaly ratio. This scan data are inherently different from the former datasets, and the generative model was found to have a more difficult time separating anomalous from normal samples, but by using the SSIM a top accuracy of 87% are achieved.

CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ADAM	Adaptive Moment Estimation
ANN	Artificial Neural Network
AUC	Area Under the Curve
AXIS	Adaptive X-Ray Inspection System
BP	Backpropagation
CAE	Convolutional Autoencoder
CCD	Charged Coupled Device
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
CVAE	Convolutional Variational Autoencoder
DAD	Deep Anomaly Detection
DUT	Device Under Test
ELBO	Evidence Lower Bound
FCNN	Fully Connected Neural Network
FN	False Negative
FP	False Positive
G-MEAN	Geometric-mean
GAN	Generative Adversarial Network
GD	Gradient Descent
GMSD	Gradient Magnitude Similarity Deviation
GPU	Graphics Processing Unit
IMED	IMage Euclidean Distance
KL	Kullback-Liebler
MLE	Maximum Likelihood Estimator

MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NN	Neural Network
PCA	Principal Component Analysis
PDF	Probability Density Function
ReLU	Rectified Linear Units
ROC	Receiver Operating Characteristic
SE	Squared Error
SGD	Stochastic Gradient Descent
SSIM	Structural Similarity Index Measure
TN	True Negative
TP	True Positive
ZNCC	Zero-mean Normalized Cross-Correlation

MATHEMATICAL NOTATION

Throughout this thesis, the following conventions regarding mathematical notation are being assumed, unless stated otherwise:

General

- The base-10 logarithmic scale is implicitly referred to as $\log = \log_{10}$. The natural logarithm, i.e. the base-e, is explicitly written as $\log_e = \ln$.
- The expectation or mean value of a variable X is denoted by $\mathbb{E}\big[X\big]$.

Sets

- Writing $x \in \mathcal{X} = \{x_1, x_2, ..., x_n\}$ means that x is an element of the set $\mathcal{X}$, and $\mathcal{X}$ consists of the listed elements $x_1, x_2, ..., x_n, n \geq 1$. Alternatively, the notation $\mathcal{X} = \{x^{(i)}\}_{i=1}^{n} = \{x^{(1)}, x^{(2)}, ..., x^{(n)}\}$ is an equivalent denotation.
- The closed interval from a to b is denoted by $[a, b]$, and consists of all real numbers $x \in \mathbb{R}$ satisfying $a \leq x \leq b$.
- The half-open interval from a to not including b, is denoted by $[a, b)$, and consists of all real $x \in \mathbb{R}$ satisfying $a \leq x < b$.

Vectorial Quantities

- Vectorial quantities are represented in non-italic and bold font, such as $\mathbf{v}$.
- The space of real-valued vectors of length n is denoted by $\mathbb{R}^n$, while the space of real $m \times n$ matrices is denoted $\mathbb{R}^{m \times n}$ given m rows and n columns.
- It is assumed that $\mathbf{x}^{n \times 1}$ is a column vector , i.e.;

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}.$$

The transpose of $\mathbf{x}$ is denoted by $\mathbf{x}^T$ and returns the row vector:

$$\mathbf{x}^T = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix},$$

i.e. $\mathbf{x}^{1\times n}$. It is also often written with parentheses as $\mathbf{x} = \left(x_1, ..., x_n\right)$.

- An identity matrix of size n is the $n \times n$ square matrix, $\mathbf{I}^{n\times n}$, with 1's on the main diagonal and zeros elsewhere, i.e.:

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

If the size of the identity matrix is immaterial and can be determined by the context it is simple denoted by $\mathbf{I}$.

Dot Product

- A dot $\cdot$ between vectors denotes a scalar product. The algebraic definition of the dot product of two vectors $\mathbf{a} = \left(a_1, a_2, ..., a_n\right)^T \in \mathbb{R}^n$ and $\mathbf{b} = \left(b_1, b_2, ..., b_n\right)^T \in \mathbb{R}^n$ is defined as;

$$\mathbf{a}^T \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + ... + a_n b_n,$$

where Σ denotes summation, n is the dimension of the vector space, and i the i^{th} component.

Hadamard Product

- A $\odot$ between two vectors denotes the Hadamard product (or Schur product), i.e. the element-wise product between two vector;

$$\mathbf{s} \odot \mathbf{t} = \begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix} \odot \begin{bmatrix} t_1 \\ \vdots \\ t_n \end{bmatrix} = \begin{bmatrix} s_1 t_1 \\ \vdots \\ s_n t_n \end{bmatrix},$$

such that $\left(s \odot t\right)_i = s_i t_i$, given $\mathbf{s}^{n\times 1}$ and $\mathbf{t}^{n\times 1}$.

- For two matrices of same dimension $\mathbf{A}^{m\times n}$ and $\mathbf{B}^{m\times n}$, it follows that the Hadamard product is a matrix of same dimension as the operands, with elements given by $\left(A \odot B\right)_{ij} = (A)_{ij}(B)_{ij}$, i.e.:

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \odot \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & \cdots & a_{1n}b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & \cdots & a_{mn}b_{mn} \end{bmatrix}.$$

Vector Norms

- The ℓ_2 norm is explicitly written as $\|\cdot\|_2$. For a vector $\mathbf{x} \in \mathbb{R}^n$ it is defined by:

$$\|\mathbf{x}\|_2 \equiv \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2},$$

  and is also called the Euclidean norm, as it corresponds to the distances between two points in Euclidean space.

Differentiation

- The Nabla operator ∇ is written as

$$\nabla = \begin{bmatrix} \dfrac{\partial}{\partial x_1} & \cdots & \dfrac{\partial}{\partial x_n} \end{bmatrix}^T,$$

  and its length depends on the number of inputs it is applied to. Component $\partial_i = \partial/\partial x_i$ denotes the partial derivative of quantity x_i.

- Given a $\alpha \in \mathbb{R}^n$ vector, $\partial\varphi$ denotes the gradient of scalar field φ in the n-dimensional parameter space;

$$\nabla\varphi(\alpha) = \begin{bmatrix} \dfrac{\partial\varphi}{\partial\alpha_1} \\ \vdots \\ \dfrac{\partial\varphi}{\partial\alpha_n} \end{bmatrix}.$$

  Here, $\partial\varphi/\partial\alpha_i$ represents the partial derivative of φ w.r.t. α_i. In particular, the following short-hand notation is defined

$$\nabla_{\alpha_i}\varphi \equiv \frac{\partial\varphi}{\partial\alpha_i}.$$

Other

- Given two functions $f : X \to Y$ and $g : Y \to Z$ the resulting function composition is denoted $g \circ f : X \to Z$ and defined by $(g \circ f)(x) = g\big(f(x)\big), \forall x \in X$.
- The convolution operation between two matrices is marked by a $*$.

Part I

SETTING THE STAGE

0 PROLOGUE

> *You kinda want to look for the anomalies. You don't actually want to look for the expected behaviour.*
>
> —Keith Rabois

Manufactured products undergoing testing are commonly known as *device under tests (DUTs)*. For every product line the interest typically lies in detecting DUTs that are defective, whereas a product can only either *pass* or *fail*. The DUT will likely pass if it is performing in accordance with the specific product specifications and fail otherwise.

In the *machine learning* terminology this falls under the problem of *binary classification*. However, training such a binomial classifier turns out to be impractical as (well-defined) samples from both normal and defective products are required. On one hand, defective products are typically hardly present in the specific product line. Furthermore, there might be many different types of malfunctions in a specific product that could lead to failure, thus leading to the easily violated problem of collecting enough samples of all possible variations within defective products. On the other hand, products that fulfill the particular specifications are typically better well-defined and easier to obtain.

Hence, it leads us to consider alternative models able to lean exclusively from DUTs that are to pass. Subsequently, in the *inference* stage the model should be able to reject defective samples that look immensely different compared to normal representations. Likewise, it is a requirement that samples living up to the specifications should pass. Nevertheless, a training paradigm for such a problem scenario is non-trivial, and till today's date there is a scarcity of *deep learning* oriented approaches utilized for (successfully) anomaly/novelty detections [12].

The work in this thesis is a contribution to a larger research collaboration termed *Adaptive X-Ray Inspection System (AXIS)* that consists of four different facilities. Each partner from the industry has their own part of the solution to deliver: *Magnatek Aps* are responsible for delivering the X-ray source; *Qtechnology A/S* for producing the camera; and *Newtec Engineering A/S* for building the machine and integrating the software. On the other hand, the Niels Bohr Institute is responsible for the software solution, which motivates the focus on robust and automate detections of defective DUTs, particularly in food inspection. Especially, there is much interest in achieving automated

solutions that are able to detect if there exists any type of foreign objects in specific products without knowing beforehand what objects they are searching after.

X-ray imaging are increasingly used in the food production industry for quality control inspection. X-ray provide a non-destructive testing as the radiation dosage does not permanently alter the inspected non-living product or objects. Therefore, X-rays makes it possible to quantitatively examine both internal and external factors of an DUT, hence locating mistakes before it is too late.

In some (rare) events foreign objects can be found inside food wrappers or the product themselves, and if sold on the market, such situations can possible result in bad customer experience or legal issues. Therefore, the detection of foreign objects is crucial, and the aim of this thesis is testing a possible deep learning scenario able to automate food control processes. Specifically, using X-ray images of particular chocolate bars and potatoes, the presence of foreign objects and other possible defects are predicted. These two food products are considered separately and their datasets have each of their challenges. The deep anomaly detection algorithms are trained using only data consisting of X-ray images of normal DUTs (without added foreign objects or other defects). Next, given a new unseen X-ray image, the algorithms predicts it either as normal or abnormal (with foreign objects or other malfunctions).

This thesis is roughly structured in two parts. In the first part the theoretical backgrounds needed to set the foundation for the rest of the thesis are reviewed. In chapter § 1 the basic principles of radiography and X-ray imaging are introduced. Chapter § 2 introduces the concepts of deep learning with special focus on computer vision tasks, in particular to *convolutional neural networks (CNNs)* that combines the two fields. Chapter § 3 serves as a formal definition to the terminology and methodology of detecting abnormalities in machine learning oriented tasks.

In the second part the pipeline behind possible automated solutions to novelty detection are demonstrated on the case studies — chocolate bars and potatoes. The theoretical framework behind the deep learning algorithms used for the novelty detections are examined in chapter § 4; the so-called *convolutional autoencoder (CAE)* and *convolutional variational autoencoder (CVAE).* Subsequently, the data of the case studies and the necessary theoretical background for preprocessing it are explored in chapter § 5. Having reviewed the theoretical background and prepared the data the novelty pipeline are implemented using the case studies, whereas the results are investigated in chapter § 6. Finally, chapter § 7 serves as a wrap-up on the work demonstrated in this thesis and possible future investigations.

The code related to this thesis is available at:

https://github.com/alinasode/novelty-detection-xray/

CONTENTS

X-RAY CONCEPTS

1

At present X-ray is utilized in a wide range of fields including physics and industry. In this chapter the physical principles of X-ray are introduced with special attention on X-ray imaging. In section § 1.1 the X-ray energy range are introduced. Following in § 1.2 the behaviour of X-rays when they interact with matter is outlined as it is the most relevant concept needed to understand contrast differences in X-ray images. Subsequently, two non-destructive X-ray detector types typically used for X-ray imaging in inspection of food products and industrial products are sparsely described in section § 1.3.

1.1 LIGHT AS PHOTONS

In order to capture an image the particular camera requires some sort of measurable energy, specifically *electromagnetic waves* or "light". In this context "light" represents electromagnetic radiations in all part of the *electromagnetic spectrum*, cf. Fig. 1.2.

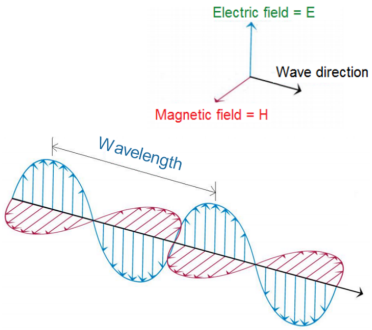


Figure 1.1: Light as an electromagnetic wave. The electric field, magnetic field and wave propagation direction are all perpendicular to each other. Image adapted from: *Wavelength, Frequency, Amplitude and phase – defining Waves! Techplayon (2017).*

Electromagnetic waves are waves from electric and magnetic fields that oscillate perpendicular to each other and to the direction of propagation [31], cf. Fig. 1.1. In the perspective of quantum mechanics, electromagnetic waves are described as a massless entity known as the *photon*. The electric and magnetic fields oscillates sinusoidally and the distance that the electromagnetic wave propagates during one cycle is known as the *wavelength*, λ_γ. The wavelength of the incident photon is connected to its *frequency*, ν_γ, by [54]

$$\lambda_\gamma = \frac{c_0}{\nu_\gamma}, \qquad c_0 \simeq 2.998 \cdot 10^8 \,\mathrm{m\,s^{-1}}, \tag{1.1}$$

where c_0 is the speed of light in vacuum. Furthermore, the energy associated with a single photon is

$$E_\gamma = \frac{hc_0}{\lambda_\gamma} = h\nu_\gamma, \tag{1.2}$$

where h denotes *Planck's constant*, $h \simeq 6.626{\cdot}10^{-34}$ J s. The energy is measured in electron volt, equal to 1 eV $\simeq 1.602 \cdot 10^{-19}$ J.

From eq. (1.2) it can be deduced that $E_\gamma \propto \lambda_\gamma^{-1} \propto \nu_\gamma$, which means the shorter the wavelength, the higher the frequency and the higher the energy. Consider the electromagnetic spectrum in Fig. 1.2. X-rays have high frequency, hence

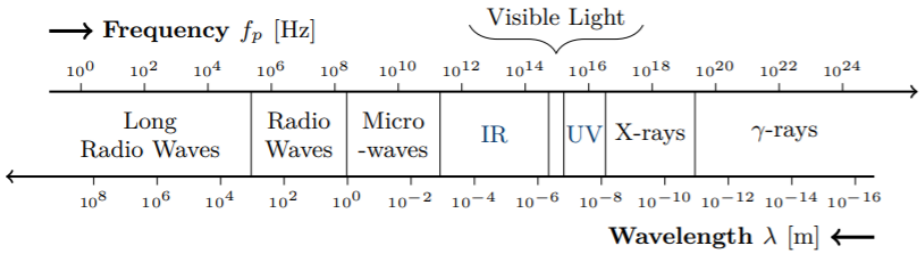


Figure 1.2: The electromagnetic spectrum given different frequencies and wavelengths that are related by eq. (1.1). The X-rays lies in the the domain of $[0.01, 10]$ nm. Image from [54].

short wavelength, and in particular the wavelength of X-rays lies in the range of 0.01 nm up to 10 nm, which approximately corresponds to the energy range of 124 keV down to 124 eV. Due to their high penetrating ability, the *hard X-ray region* with energies typically spanning in $[10, 124]$ keV is utilized in industrial and medical radiography [54].

1.2 INTERACTIONS OF PHOTONS WITH MATTER

When X-rays propagates though matter they lose a certain amount of energy dependent on the absorption behaviour of the specific media. This section serves to explaining the relation between the energy of the photon and the absorption in the materials since such reductions of energy is the main principle of X-ray imaging. Sections § 1.2.1-§ 1.2.4 summarises the major mechanics that cause attenuation of an incident photon beam, whereas in § 1.2.5 the attenuation contrast in X-ray imaging is explained.

1.2.1 *Photoelectric Absorption*

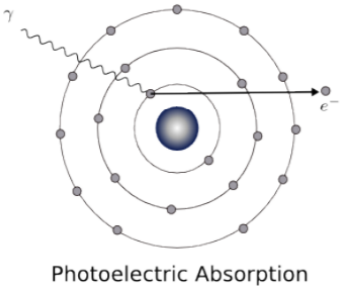


Figure 1.3: In the photoelectric absorption a photon, γ, is fully absorbed by an electron, e^-, resulting in the ejection of the electron, leaving the atom in an ionized state. Image from [54].

When an incident photon is absorbed completely by a tightly bound orbital electron (i.e. when $E_b \lesssim E_\gamma$) that results in the electron being ejected with kinetic energy $E_{e^-} = E_\gamma - E_b$, an *photoelectric absorption* is said to have occurred, cf. Fig. 1.3 [43, 59]. Here, E_b denotes the electron shell binding energy. This process results in the atom being ionized, and by subsequently emission of a characteristic radiation the vacancy in the inner shell can be filled by an outer shell electron, such that the atom can return to its neutral state.

These interactions are non-trivial, but a rough approximation for the probability of photoelectric absorption per atom is proportional to $Z^n/E_\gamma^{3.4}$, where Z is the *atomic number* and $n \in [4, 5]$ [43]. From this relationship it follows that the photoelectric process is more dominant for lower energies, and absorber materials of high atomic numbers absorbs photons to a larger degree.

1.2.2 Incoherent Compton Scattering

Compton scattering occurs when an incident photon is deflected from its original path by an interaction with a loosely bound orbital electron, i.e. when $E_b \ll E_\gamma$, see Fig. 1.4. Due to the interaction the electron gains kinetic energy and is ejected from the atom, while the photon loses energy but continues to travel through the material along an altered path with a longer wavelength [54]. The event is known as an *incoherent scattering* as the energy transfer depends on the angle of scattering. Furthermore, the probability of occurrence of Compton scattering depends on the number of free electrons available, thus the probability increases with increasing atomic number [43].

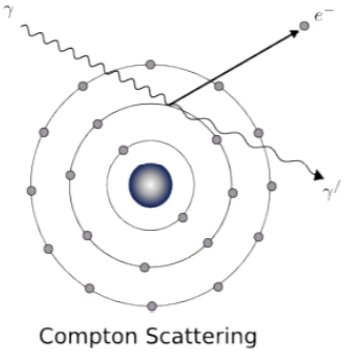


Figure 1.4: Compton scattering. The incident photon, γ, transfer a portion of its energy to the recoil electron, e^-, whereas the path of the scattered photon, γ', with energy $h\nu_{\gamma'} < h\nu_\gamma$ is deflected by some angle. Image from [54].

1.2.3 Pair Production

In order to produce a positron and an electron the energy of the incident photon must exceed $E_\gamma \geq 2m_e c_0^2 = 1.02\,\text{MeV}$ as the rest energy of an electron and positron is $m_e c_0^2 = 0.51\,\text{MeV}$ [43, 61]. Such process is known as a *pair production* in the nuclear field, and with the annihilation of the photon a electron-positron pair is created, see Fig. 1.5. The effect of pair production only becomes significant at even larger energies — well outside the scope of the X-ray range.

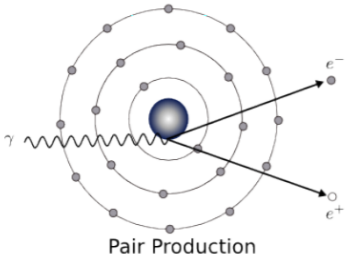


Figure 1.5: Pair production in the nuclear field. An incident photon, γ, interacts with the nucleus and its total energy is transformed into and electron, e^-, and a positron, e^+. Image from [54].

1.2.4 Coherent Rayleigh Scattering

Rayleigh scattering is a coherent process that is caused by an interaction of an incident photon with bounded orbital electrons, and typically occurs only for low photon energies [43, 54]. The process neither excites nor ionizes the atom, but the photon simply changes its direction with no change in internal energy, cf. Fig. 1.6. Compared to that of the photoelectric effect the contribution of coherent scattering is negligible for X-ray imaging [61].

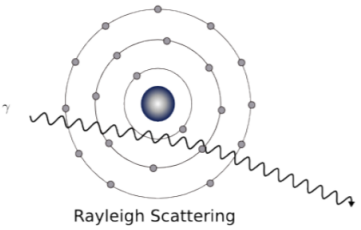


Figure 1.6: Rayleigh scattering. An incident photon, γ, interacts with the atom and changes its direction without any loss of energy. Image from [54].

1.2.5 Attenuation Contrast

Given the intensity of an incident monoenergetic photon beam, I_0, the attenuation of radiation after the beam penetrates a layer of material with mass thickness x is described by *Lambert-Beer's law* [54, 59]

LAMBERT-BEER'S LAW

$$I(x) = I_0 e^{-\mu x}. \tag{1.3}$$

Here, the *linear attenuation coefficient* is material dependent and consists of

all possible different types of absorption and scattering interactions, i.e. the total sum [61]

$$\mu = \mu_{\text{pe}} + \mu_{\text{comp}} + \mu_{\text{pp}} + \mu_{\text{cs}}, \tag{1.4}$$

where μ_{pe}, μ_{comp}, μ_{pp} and μ_{cs} are the contributions from the photoelectric effect, Compton scattering, pair production and coherent scattering, respectively. Hence, attenuation is simply the reduction of the number of photons that arrive at the detector. It follows that μ decreases with increasing photon energy, and μ increases with increasing atomic number and increasing density of the absorbing material, ρ.

The attenuation coefficients can be used for selecting an optimal radiation energy that will produce the largest contrast between specific materials in X-ray imaging. Consider for example Fig. 1.7 where the linear attenuation coefficients of pure iron and carbon have been plotted. The energy at the largest difference in attenuation between the two materials will result in the greatest contrast in the X-ray image.

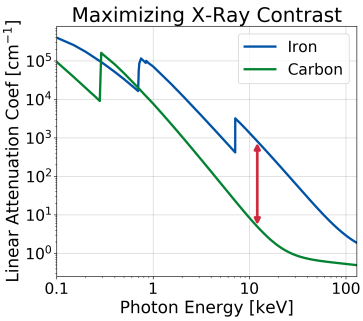


Figure 1.7: Log-log plot of the linear attenuation coefficient versus photon energy between pure carbon ($Z = 6$) and pure iron ($Z = 26$). The absolute maximum seperation between the two curves occurs around 12 keV. Values from the database compiled by [23] have been used.

Since μ is dependent on the density of the medium the normalized density independent *mass attenuation coefficient* is often used for convenience [43]

$$\mu_m = \frac{\mu}{\rho}. \tag{1.5}$$

As for the linear attenuation coefficient described by (1.4), the total mass attenuation coefficient also consists of separate mass attenuation coefficients for the different processes outlined in § 1.2.1-§ 1.2.4. Depending on the photon energy and the absorber material the partial contributions varies.

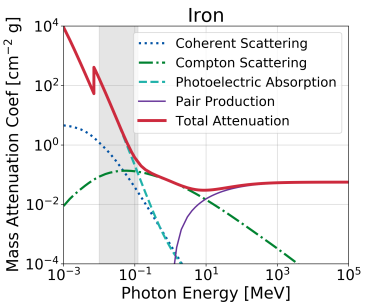


Figure 1.8: Log-log plot of the mass attenuation coefficients for pure iron ($Z = 26$) with contributions sources of attenuation. The gray shaded area spans the hard X-ray region of $[10, 124]$ keV. Values from the XCOM (Berger and Hubbell, 2010) database have been used.

Consider for example the mass attenuation coefficients for iron plotted as a function of photon energy in Fig. 1.8. It can be seen from the plot that at relative low photon energies (< 100 keV) the photoelectric effect dominates. At intermediate energies (100 keV - 10 MeV) the Compton scattering contributes the most. In general, the contribution from Rayleigh scattering is small compared to the other processes. At even larger energies (> 10 MeV) the pair production dominates, however, since the industrial X-ray radiography is typically done in the hard X-ray region, the contribution from pair production becomes irrelevant. Hence, the photoelectric absorption and Compton scattering accounts for the majority of attenuation encountered in the application of food inspection.

Obvious food products does not consists of a single chemical element. Nevertheless, the mass attenuation coefficient of mixtures or compounds of elements can be obtained according to the simple additivity [43]

$$(\mu_m)_{\text{c}} = \sum_i w_i (\mu_m)_i, \tag{1.6}$$

where w_i is the weight fraction of the i^{th} atomic element.

1.3 X-RAY IMAGING

The images described in section § 5 have been obtained either by an *area scintillator detector* or a *line detector*, and the intuitive working mechanic behind them are sparsely described in § 1.3.3 and § 1.3.4, respectively. Common for both detector based methods are that incoming X-rays are converted into visible light though the detectors by use of a specific detector material introduced in § 1.3.2. Nevertheless, first and foremost the principle behind how X-ray are produced and how they are characterized are described in § 1.3.1.

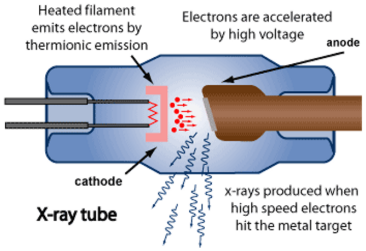


Figure 1.9: Schematic illustration of an X-ray tube. In order to direct the produced X-rays the anode is tilted by a certain angle. Image source: *X-rays. ARPANSA.*

1.3.1 Generation of X-Rays

An *X-ray tube* is a vacuum tube whose main components are a negatively charged *cathode* and a positively charged *anode* [54], see Fig. 1.9. The electrons emitted by the cathode are accelerated through a potential difference towards the anode whereas they interacts with this target material. There are two types of interactions that produces X-ray radiation.

An incident electron may collide with an electron within an atom and dislodging it, hence leaving a vacancy that is filled by another orbital electron from a higher energy level. This process results in the production of a *characteristic X-ray* photon whose energy depends on the atomic number of the target material. This interaction can only occur if the incoming electron has kinetic energy larger than the binding energy of the orbital electron it collides with.

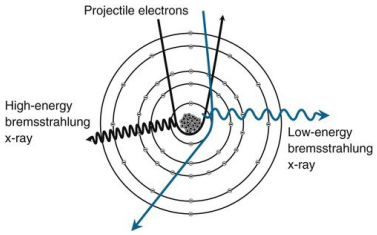


Figure 1.10: Bremsstrahlung. Incident electrons are subjected to the columnb field of an atomic nucleus, resulting in the emission of a photon. Image source: *Bremsstrahlung Radiation. PhysicsOpenLab (2017).*

As incoming electrons penetrate the anode material and pass close to an atomic nucleus they are deflected and deaccelerated, resulting in a loss of kinetic energy that is converted into a photon. This type of electromagnetic radiation is known as *bremsstrahlung* since the path of charged particles are being bended. As illustrated on Fig. 1.10, the amount of energy lost can vary from zero to the total incident energy [54]. Electrons striking near the center are subjected to a greater interaction resulting in the producing of photons with higher energy, whereas electrons hitting further away produces lower-energy photons. The probability of electrons hitting the outer nuclear field zones is largest, and so the emission of lower-energy photons are dominating.

The metallic anode plate is typically a materiel of high atomic number and high melting point, such as tungsten with $Z = 74$. Fig. 1.11 shows simulated X-ray emission spectra for pure tungsten. The sharp peaks at certain energies corresponds to the characteristic X-rays associated with tungsten, whereas the continuous X-rays are due to bremsstrahlung. The lower-energy photons have been filtered out by applying a thin metal plate, which entails that the average energy of the emitted spectra has lowered — such change is known as *beam hardening*. The soft X-rays (< 10 keV) are easily absorbed, reflected,

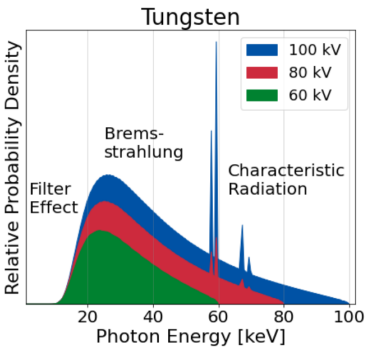


Figure 1.11: Simulated X-ray emission spectra at different tube voltages for pure tungsten given 55° target angle and filtered by 1 mm aluminium. The bremsstrahlung spectrum has a maximum photon energy that corresponds to the energy of the incident electrons. Simulations generated with `SpekPy`.

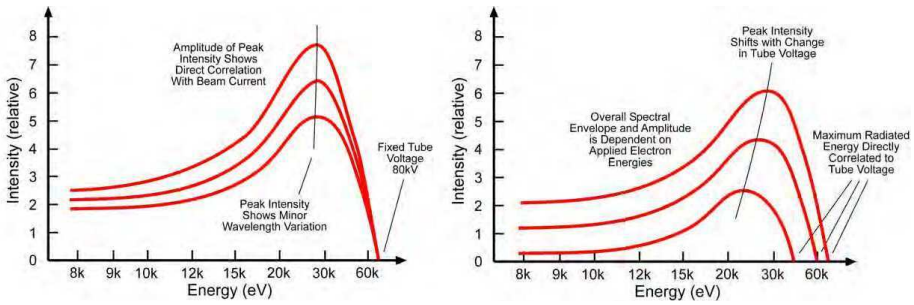


Figure 1.12: Behaviour of the bremsstrahlung spectrum emitted by an X-ray tube when modifying its current while keeping the voltage constant (left) and voltage while keeping the current constant (right). Images adapted from [82].

or scattered, but cannot penetrate the samples for imaging purposes; hence they are being absorbed by such an pre-filtering in order to increase the quality of recorded images. Increasing the *tube voltage*, which electrons are accelerated though, influence the production of characteristic X-ray and the peak intensities. As the voltage difference between the anode and cathode limits the maximal energy of created X-rays, the peak value of the emission spectrum implicitly corresponds to the tube voltage.

Consider the illustrations in Fig. 1.12 for which the effects of varying the *tube current* and voltage is shown. The overall energy of an X-ray beam is refereed to as the *quality*. The photon distribution, and thus the quality, is affected by a number of parameters such as the atomic number of the target material, changes in the filtration, and most importantly of all, the tube voltage. As the tube voltage determines the penetrating ability of the X-ray beam, increasing the voltage results in the X-ray beam having higher mean energy, causing a beam hardening effect that shifts the energy spectrum.

On the other hand, the *quantity* refers to the number of photons produced during an exposure, or the intensity. A number of factors influences the quantity, however the two most important ones are the tube current and the *exposure time* — but not the voltage. The exposure time is simply the duration of X-ray production. Hence, as either the current or exposure time increases the number of electrons striking the target increases, thus the photons generated in the X-ray tube increases. I.e. in summary the voltage controls the contrast and the current the amount of X-rays.

1.3.2 Scintillators

Scintillator is a material that convert high photon energy into optical light. The scintillator completely absorbs the energy of the incident photon, resulting in exciting atoms of the material. Subsequently, when the atoms de-excites, photons with energy within the optical region of the electromagnetic spectrum are emitted. Hence, in this way the incident X-ray has been converted into visible light.

Scintillators are employed in *indirect X-ray detection methods*. First, the

incident photons are converted to optical light using a scintillator. However, the emitted light is far to weak to be detectable, and thus needs to pass though a *charged coupled device (CCD)*. Here, the optical light is converted into current which in turn can be transferred into digital signals that can form the X-ray images.

1.3.3 Area Scintillator Detectors

One dataset consisting of specific chocolate bars, which is further presented in section § 5.2, have been acquired by an X-ray set-up with a 2D area scintillator detector borrowed from *Newtec Engineering A/S*. The set-up requires scanning objects to be motionless and stationary during the process. It however provides easy real-time access to adjust the settings for the voltage and current, reducing the time needed for calibrating. After manually calibrating for optimal settings, the machine are capable of producing images of high quality and resolution. The machine are capable at operating in the tube voltage range $[35, 105]$ kV and tube current range $[0.35, 7.50]$ mA, and given a maximal tube power 525 W.

1.3.4 Line Detectors

Besides area scintillator detector, industries frequently employs *line detectors* due to their fast image acquisition and data processing.

In the typically set-up utilized in food inspections, the scanning object moves past a stationary placed line-camera by means of an in-line conveyor belt under a constant velocity. In fast succession, the camera only captures one image line at a time, finally producing a continuous 2D image by "stitching" them together, cf. Fig. 1.13.

However, it is all a matter of the right settings. The *line rate* of the camera, i.e. the number of rows sampled each second, should be adjusted to match the current speed of the conveyor belt. If they are synchronized, precise and meaningful 2D analysis of the images can be made. If not, e.g. if the line rate is fixed but the speed of the conveyor belt varies, the object image will either be elongated or compressed [36].

ButeoX

The second dataset presented in section § 5.2 consists of potatoes with and without needles and holes. A former Master student at Niels Bohr Institute have recorded this data on the X-ray prototype-machine known as *ButeoX*

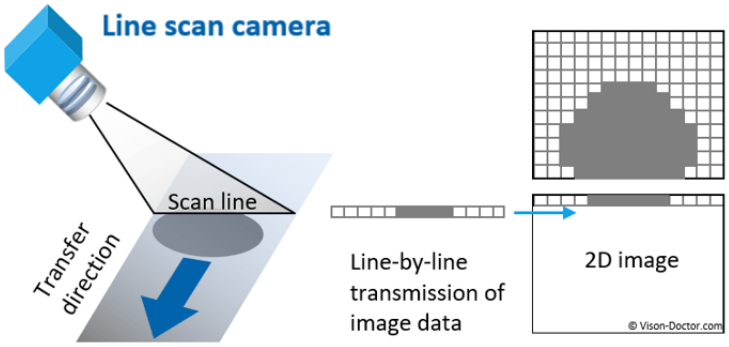


Figure 1.13: Schematic illustration of the principle of image acquisition using line-scan cameras. Image source: *Vision Doctor, Line scan camera basics (2019).*

which was located at the H.C. Ørsted Institute in Copenhagen. The ButeoX is a line scanner, whose main components are the conveyor belt, X-ray source and line detector.

The set-up consist of the conveyor belt which is mounted inside a steel case. In combination with a scintillator, the line detector is placed in the area in between the belt layers, i.e. beneath the scanning samples. The X-ray source produces X-rays with a Tungsten anode tube and the unit is placed above the conveyor belt.

The installed X-ray source operates within voltages $[40, 80]$ kV and with a maximal power of 100 W. As for the camera during the scanning, the line rate was set to 1000 Hz.

The ButeoX always returns images with a width of 410 pixels, but have arbitrary lengths that naturally varies depending on the scanning object. The pixel size of the line detector is 0.8 mm. The speed of the conveyor belt can then be calculated by taking the product of the resolution and line rate, i.e. the maximum speed during the scanning is $0.8\,\text{mm} \cdot 1000\,\text{Hz} = 0.8\,\text{m}\,\text{s}^{-1}$.

1.4 CASE STUDIES: POTATOES AND CHOCOLATE

As briefly mentioned, X-ray images of potatoes and a brand-name chocolate bar are the objectives considered in this thesis. During the X-ray scanning the tube voltage and current needs to be explicitly chosen, with primary focus on increasing the contrast between food product and foreign objects. And although raw images have been forwarded to be examined, it is still considerable to explicitly investigate the absorption of the primary elements found in the organic products.

Ideally, eq. (1.6) should be used to investigate the expected spectrum from a potato or chocolate bar. However, since neither food products have been chemically analysed, the actual chemical compositions are not known.

Nevertheless, the primary compounds of potatoes are starch ($C_6H_{10}O_5$), water

(H_2O), and a negligible amount of protein, and where the typically starch content of the total potato mass is around $15 - 25\%$. To the potatoes are added foreign objects mainly composed of iron ($Z = 26$). Fig. 1.14 shows the mass attenuation for starch and water within the relevant domain for the ButeoX. Depending on the starch content the attenuation of potatoes will change slightly, and carving holes in the potatoes reduces the thickness of the potato. Nonetheless, the significant higher mass attenuation coefficient for iron makes it easy to detect foreign objects inside the food product, while the hollow hearts are a bit more problematic but can still be detected.

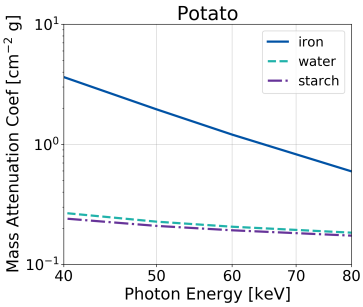


Figure 1.14: Log-log plot of the mass attenuation coefficients for pure iron ($Z = 26$) and compounds of water and starch. Values from the XCOM (Berger and Hubbell, 2010) database.

The chocolate bars primary consists of milk chocolate ($\sim 66\%$), and its remaining content mainly of wheat flour and sugar. The chemical composition of these factory made chocolate bars are way harder to analyse compared to potatoes, however its primary elements are carbon ($Z = 6$), hydrogen ($Z = 1$), nitrogen ($Z = 7$) and oxygen ($Z = 8$). The foreign objects added to the chocolate bars are mainly composed of iron, lead ($Z = 82$) and aluminium ($Z = 13$). On Fig. 1.15 the mass coefficients for each of these elements are plotted within the range relevant for the area scintillator detector, cf. section § 1.3.3. Again, it is worth noticing the distances between the elements in the organic chocolate bar and the elements the non-organic foreign objects are composed of.

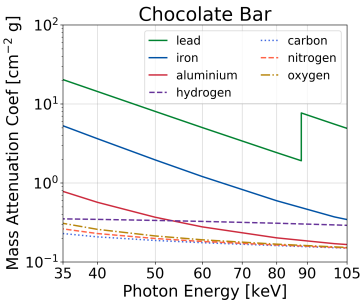


Figure 1.15: Log-log plot of the mass attenuation coefficients for pure iron ($Z = 26$), lead ($Z = 82$), aluminium ($Z = 13$), carbon ($Z = 6$), hydrogen ($Z = 1$), nitrogen ($Z = 7$), and oxygen ($Z = 8$). Values from the XCOM (Berger and Hubbell, 2010) database.

Hence, the attenuation coefficient explains why X-rays pass freely though materials of low atomic number, while objects of much higher atomic numbers absorbs the incident photons. That is, the very large attenuation coefficients of the elements found in foreign objects, leads to photon starvation as most photons are absorbed and only an insufficient number of them can be measured; this is essentially what gives the contrast difference in the X-ray imaging. Hence, dense areas, such as foreign objects, will appear darker on the X-ray images compared to the food products which will appear in shades of gray. Air, that have even lower densities than the scan objects, will show up as almost completely white.

CONTENTS

DEEP LEARNING FOR COMPUTER VISION

2

This chapter summarizes some of the concepts in computer vision, deep learning and convolutional neural network required to understand the later chapters of this thesis. It touches upon the intuitions, concepts as well as specific tools used in the algorithms described in chapter § 4 and forward.

Most considerations in this chapter follow two great textbooks, one by M. A. Nielsen: *Neural Networks and Deep Learning* [56] and another by I. Goodfellow, Y. Bengio, and A. Courville: *Deep Learning* [27].

Section § 2.1 presents an introduction to the field of deep learning, and showcases the fundamental ideas, concepts and architecture behind neural networks.

In section § 2.2 the fields of computer vision and deep learning are combined, such that the concept and architecture of convolutional neural networks can be introduced.

2.1 INTRODUCTION TO DEEP LEARNING

Deep learning is one of the most popular subfield of machine learning that deals with different types of *artificial neural networks (ANNs)*. Deep learning can perform task such as image classification, object detection, image segmentation and image restoration — common for all such tasks is that deep learning uses large neural networks to teach machines to automate tasks.

A dataset is split into subsets called *training set*, *testing set* and *validation set*. During training, the model is given the training set and the aim is for the model to gradually learn from those training examples. The goal of training is to learn enough on the training data so that the model can make predictions about the testing data. The validation set is typically used to provide an unbiased evaluation of a given model fit.

Many techniques and practices are applied in training neural networks and this section seeks to give an overview about them. The main concepts of ANNs are introduced in § 2.1.1, while some types of most popular activation functions are outlined in § 2.1.2. Back-propagation algorithms are used during training of neural networks and an overview is outlined in § 2.1.3. Loss functions and optimizers are necessary for compiling the model and are discussed in § 2.1.4 and § 2.1.5, respectively.

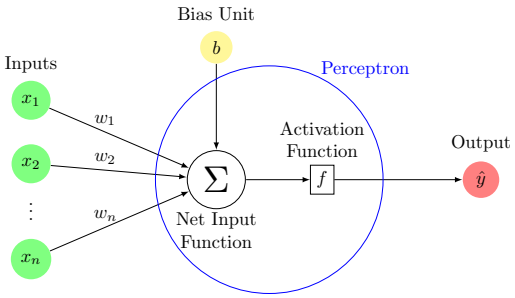


Figure 2.2: The concept in ANN. The perception is one single neuron that takes n weighted inputs plus a bias and produces one output. In this general and particular case the activation function is fed with the linear representation $f(\mathbf{w}^T \cdot \mathbf{x} + b) = f(x_1 w_1 + x_2 w_2 + ... + x_n w_n + b)$.

2.1.1 *Artificial Neural Networks*

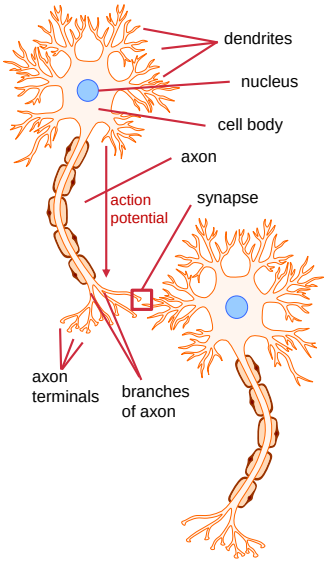


Figure 2.1: Illustration of how two biological neurons might connect: The terminal axon of the first neuron connects to the dendtries of the second target neuron and communatates togehter through the synapse. Inputs (stimuli) are taken through the dendrites, and upon firing of an action potential (which fires above some certain threshold), the pre-synaptic ("sending") neuron sends a signal down its axon to its terminal axons. Here, the signal is transmitted and outputted to the post-synaptic ("receiving") neuron, which in turn decides whether or not to fire its own action potential [1, 45]. Illustration modified from *Motifolio*.

ANNs are computational models inspired by animal's central nervous systems (in particular the brain) and their functionalities. Inspired by biological neural networks, ANNs consists of artificial neurons (processing units), which are interconnected by edges. Imitating how two biological neurons might connect through a synapse (cf. Fig. 2.1), the processing units of the ANN are made up of inputs and outputs units. Based on some internal weighting system, the input unit receive various forms and structures of information, and using an assigned *activation function* the *neural network (NN)* attempts to learn about the information presented in order to produce an output unit [56].

Consider the mathematical model shown in Fig. 2.2. The simplest and the basis forming unit of an ANN is the so-called *perceptron*, which simply can be understood as anything that takes multiple inputs and produces one output. Terminology following Fig. 2.1, the perceptron can be thought of as the cell body. Based on the strength at a synapse (w_i), the signals that travel along the axons (x_i) from a pre-synaptic neuron and onto the dendrites of a post-synaptic neuron interact multiplicatively ($w_i x_i$). The dendrites carry the signal to the cell body where they all get summed. Given some activation function, neurons can then fire along its axon, whereas the output $\hat{y}$ represents the output axon [71]. ANNs are thus inspired by mechanisms of biological neural networks, but they are in no way identical and their similarities are very far fetched.

As mentioned above, when inputs are transmitted between neurons, respective weights are applied to them. The weights are essentially reflecting how important an input is, and the larger the value, the more influence the particular input has on the neuron's output. Typically, an optimal *bias* is added to the product of inputs and weights for a perceptron. The bias is simply a constant value, and depending on the assigned weight it is utilized to offset the result. With other words; the bias can be thought of as how flexible the perceptron is.

Hence, the activation function takes the sum of weighted input plus the bias term, b, as arguments and returns the predicted output, $\hat{y}$. Thus, the

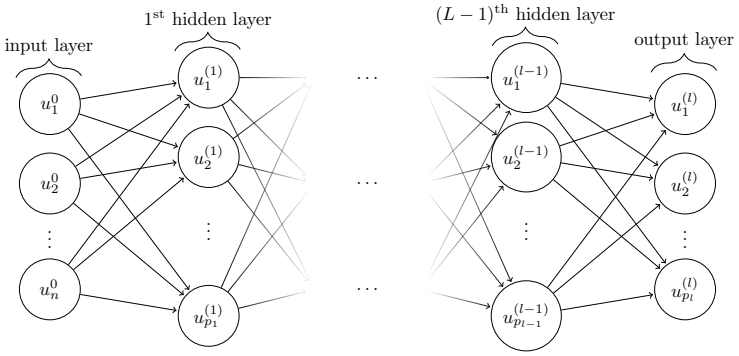


Figure 2.3: Architecture of fully connected network of a L-layer perceptron given n input units. Each layer contains p (hidden) units. Do notice that the input layer does not contribute in the count of layers, whiles the output layer does, as weights only are learn-able to nodes of the following layer in the feed-forward NNs [45, 81].

activation function is fed with the following linear term [1, 56]

$$\hat{y} = f\left(\mathbf{x} \cdot \mathbf{w}^T + b\right) = f\left(\sum_{i=1}^{n} w_i x_i + b\right), \tag{2.1}$$

where $\mathbf{x} = \left(x_1, ..., x_n\right)^T$ is the feature input vector and $\mathbf{w} = \left(w_1, ..., w_n\right)^T$ the weights. There are multiple activation functions that can be used in order to make a computation non-linear, and a discussion follows in § 2.1.2.

So far, only a single perceptron has been considered for an ANN. Subsequently, a *multi-layer perceptron (MLP)* consists of multiple interlinked layers, called *hidden layers*, stacked between the *input layer* and the *output layer* [66], see Fig. 2.3. Each layer is made up of a set of nodes, where each node is fully connected to all nodes in the previous layer. Furthermore, the nodes in a single layer are completely independent and do not share any connections to each others.

It is now possible to transform eq. (2.1) into a generalized formula for a *fully connected neural network (FCNN)*, [56]. Let $\mathbf{u}^l = \left(u_1^l, ..., u_{p_l}^l\right)$ denote the outputs for layer l for p_l number of nodes. Furthermore, let the input layer be initialized by setting $u_j^0 = x_j, \forall j \in \{1, 2, ..., n\}$, where j denotes a node. For each neuron j in layer l, the *weighted input*, i.e. the intermediate quantity, is calculated according to

$$z_j^l = \mathbf{w}_j^l \cdot \mathbf{u}^{l-1} + b_j^l, \quad \forall j \in \{1, 2, ..., p_l\}, \tag{2.2}$$

where b_j^l denotes the bias and $\mathbf{w}_j^l = \left(w_{j1}^l, ..., w_{jp_{l-1}}^l\right)$ the weights for layer l and node j. I.e. the notation for the weight $w_{jp_{l-1}}^l$ should be read as the connection from the p^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer. Then, an activation function $f_l(\cdot)$ is applied and the neuron outputs the following entity to the subsequent nodes

$$u_j^l = f_l\left(z_j^l\right), \quad \forall j \in \{1, 2, ..., p_l\}. \tag{2.3}$$

This transformation occurs at all nodes in the $(L-1)$ hidden layers of the FCNN, and the output layer will thereafter yield $\mathbf{u}^L$.

2.1.2 Activation Functions

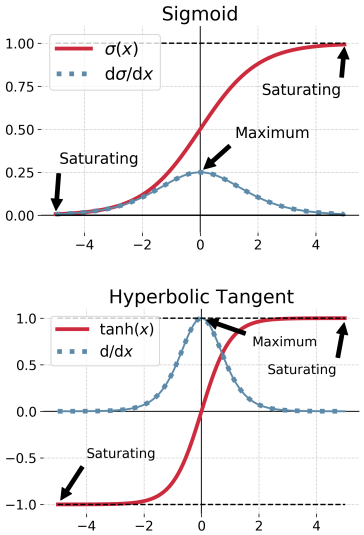


Figure 2.4: The logistic sigmoid and hyerbolic tangent functions and their derivatives.

The single layer perceptron is effectively simply a dot product between an input layer and a set of learning weights, which means that it is actually just a linear transformation, cf. eq. (2.1). In order to make complex problems easier to solve it is therefore important to use non-linear activation functions. This is owned by the fact that any linear combination of linear functions collapses down to be a linear function [81].

Depending on the problem at hand, there exists a wide variety of different activation functions that can be used. In principle, the activation function decides if an entire node gets to send data or not, and therefore the particular choice of activation type is crucial. Moreover, they have a significant impact on the learning speed of the network. The non-linear activation functions that have been used in the implementations in § 6 are the *sigmoid*, *hyperbolic tangent*, *rectified linear units (ReLU)* and *leaky ReLU* activations, cf. Figs. 2.4-2.5.

LOGISTIC SIGMOID ACTIVATION FUNCTION

All types of sigmoid functions saturates at either ends. Specifically, in machine learning it is the *logistic sigmoid function* that is referred to, and this non-linear transformation is described by [66, 71]

$$\begin{aligned} \sigma(x) &= \frac{e^x}{e^x + 1} = \frac{1}{1 + e^{-x}}, \\ \frac{d}{dx}\big(\sigma(x)\big) &= \sigma(x)\big(1 - \sigma(x)\big). \end{aligned} \tag{2.4}$$

This function takes in any real-valued input and will render outputs in $[0, 1]$. In the limit $x \to \infty$ the sigmoid function converges to 1, and towards 0 in the case of x tending towards $-\infty$, and so its values saturate at either end. This yields the undesirable property that its gradients rapidly converges towards and reaches zero[1]. This makes networks slow to train because the gradients control how rapidly the network changes during training (as elaborated in § 2.1.3). But, if gradients are so small (or zero), such that the change a weight undergoes is negligible, and hence making the network stuck at a particular measure of loss, then little to no training can take place. This issue is commonly known as the *vanishing gradient problem*, and leads to poor network performance.

1. The sigmoid function is differentiable $\forall x$. Evaluating the limits for the gradient of the sigmoid function yields: $\lim_{x\to\infty} \sigma(x)\big(1 - \sigma(x)\big) = \lim_{x\to-\infty} \sigma(x)\big(1 - \sigma(x)\big) = 0$. Furthermore, simply by looking at its graph in Fig. 2.4, moving away from the origin $x = 0$ at either sides returns smaller and smaller values for its gradient at a fast rate.

HYPERBOLIC TANGENT ACTIVATION FUNCTION

The hyperbolic tangent activation is a non-linear continuous function that produces outputs in the scale of $[-1, 1]$ for all $x \in \mathbb{R}$, as it is given by [71]

$$\begin{aligned} \tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}}, \\ \frac{d}{dx}\big(\tanh(x)\big) &= 1 - \tanh^2(x). \end{aligned} \tag{2.5}$$

Compared to eq. (2.4), the derivatives of the hyperbolic tangent are steeper, however this activation also suffers from the vanishing gradient problem.

Another commonly used activation, that does not saturate in $\mathbb{R}^+$, is ReLU. The ReLU is simply a max operation between zero and an input. Hence, compared to the sigmoid function, which has negative x values in addition to positive ones, the ReLU function only has positive x values [66, 71]:

$$f(x) = \begin{cases} 0, & \text{for } x \leq 0 \\ x, & \text{for } x > 0 \end{cases} = \max(0, x),$$
$$\frac{\mathrm{d}}{\mathrm{d}x}\big(f(x)\big) = \begin{cases} 0, & \text{for } x \leq 0 \\ 1, & \text{for } x > 0 \end{cases} = \max(0, 1). \tag{2.6}$$

RELU ACTIVATION FUNCTION

The ReLU has a constant gradient of 1 for positive x, however on the other end for negative x it has zero gradients. Since the activation is constant in the region $\mathbb{R}^+$ it is therefore more resilient over for the vanishing of gradients. However, its zero gradient in the region $\mathbb{R}^-$ poses another problem known as the *zero gradient problem*, whereas data will belong to "dead" ReLU nodes that will not contribute to updates of the weights during training.

In order to compensate for this problem a small positive linear term in x is often added to give a non-zero slope at all points. This way, the leaky ReLU activation remedy the problem by adding $\alpha = 0.01$ [71]:

$$g(\alpha, x) = \begin{cases} \alpha x, & \text{for } x \leq 0 \\ x, & \text{for } x > 0 \end{cases} = \max(\alpha x, x),$$
$$\frac{\mathrm{d}}{\mathrm{d}x}\big(g(\alpha, x)\big) = \begin{cases} \alpha, & \text{for } x \leq 0 \\ 1, & \text{for } x > 1 \end{cases} = \max(\alpha, 1). \tag{2.7}$$

LEAKY RELU ACTIVATION FUNCTION

Leaky ReLU thus entails some leakage for $x \in \mathbb{R}^-$, which counteracts nodes being inactive in this region, as was the problem with the ReLU. This leads to its gradient being non-zero in $\mathbb{R}$.

ReLU and leaky ReLU are usually employed in the intermediate layers of a FCNN. ReLU is cheap to compute, and in [46] it was found that compared to using sigmoid or tangent activations, using ReLU activations greatly accelerates the convergence of the optimizer. The characteristic S-shape of the sigmoid makes it useful at converting a real-value into one that can be interpreted as a probabilistic score in the interval $[0, 1]$. The hyperbolic tangent squeezes outputs in the domain $[-1, 1]$ and is zero-centred. Because of these properties and depending on what the NN should return, the sigmoid or tangent usually serves as the last activation function in a FCNN [27].

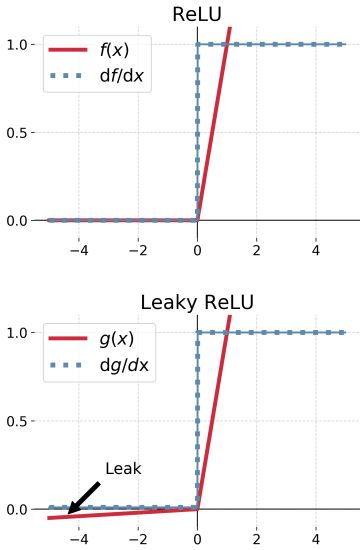


Figure 2.5: The ReLU and leaky ReLU (given $\alpha = 0.01$) functions and their derivatives.

2.1.3 Backpropagation

The data flow of the processes explained until now in § 2.1.1, where the output is computed by the input, is know as *forward propagation*. Upon initialization of a NN, the estimated output is typically far away from the desired output

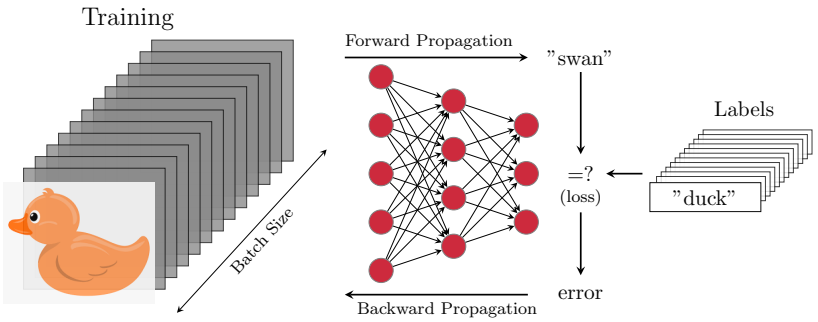


Figure 2.6: Given training data divided into mini-batches, the NN is executed in the following order. In the forward propagation, features are extracted and, in this example, output labels are predicted. Evaluating some given loss function, errors are then calculated. Subsequently, in the backward propagation, the influence of each parameter w.r.t. the current loss is found and weights are updated accordingly. Illustration inspired by [81].

and so the weights of the model will be far from ideal. In order to reduce this error, the weights and biases are adjusted and updated based on the impact each had on the current error. Such process is know as *backward propagation* [81], where the data flow has been reversed (going from output unit to input units), so that the weights and biases are recalculated for each activated node.

Henceforth, the NN is executed in two distinct modes during the training phase, see Fig. 2.6. A *loss function*, whose role is elaborated in § 2.1.4, is defined in the forward propagation; one best fit for evaluating the specific model for what it should achieve. A *backpropagation (BP)* algorithm works by evaluating the loss at the output and then propagating it back into the network, for which weights are updated in order to minimize the error.

In order to get successful results from a FCNN, some large amounts of training data is usually required. Accordingly, in practice the dataset is typically divided into smaller pieces, called *mini-batches*. During training, each mini-batch will in turn be loaded and fed to the network, where the BP algorithm will calculate the gradients and update its weights [81]. This procedure is repeated until all samples in the training set have been used once; then one *epoch* of training is said to have been executed.

Under the BP algorithm, gradients are calculated of each parameter w.r.t. to the differentiable loss function. This is what makes the NN "learn" by combining it with a gradient-based optimizer (that are introduced in § 2.1.5). But why gradients? Gradients (rate of changes) gives how one quantity changes in relation to another quantity. Hence, using the chain-rule from calculus[2] the influence of a certain input, on systems that are composed of multiple functions, can be found. As an example illustrating such gradient flow, consider Fig. 2.7 [66]. The forward pass (indicated by green arrows) calculates z as a function $f(x, y)$ using the input variables x and y. In turn, in the backward pass (indicated by red arrows) the following operations occurs: By receiving the gradient of the loss function, w.r.t. z, i.e. $\partial\mathcal{L}/\partial z$, then the gradients of x and y on $\mathcal{L}$ can be calculated using the chain-rule as indicated in the figure.

Following this logic, the error in each layer of a FCNN can be described and

2. The single-variable chain rule written in nested form is given by: $z' = (f \circ g)'(x) = f'(g(x))g'(x), \forall x$, where f and g are differentiable functions [57]. Alternatively, by introducing an intermediate variable $y = g(x)$ it can instead be written in Leibniz's notation: $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$. That is; variable z is dependent on variable y, which in turn is dependent on variable x, and hence z depends on both x as well as y.

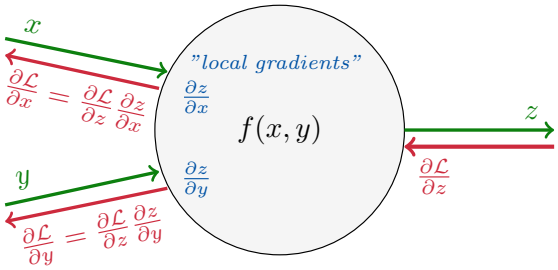


Figure 2.7: Gradient flow during back-propagation.

related to its weights and biases [7, 56]. Suppose the error in layer l and neuron j is defined by

$$\delta_j^l \equiv \frac{\partial \mathcal{L}}{\partial z_j^l}. \tag{2.8}$$

Here $\delta^l = (\delta_1^l, ..., \delta_{p_l}^l)$ denotes the vector of errors associated with layer l. If the layer in question is the output layer, s.t. $l = L$, eq. (2.8) can be re-expressed using the chain-rule and eq. (2.3) according to

$$\delta_j^L = \frac{\partial \mathcal{L}}{\partial u_j^L}\frac{\partial u_j^L}{\partial z_j^L} = \frac{\partial \mathcal{L}}{\partial u_j^L}\frac{\partial f_L(z_j^L)}{\partial z_j^L}, \tag{2.9}$$

for an activation function $f_L(\cdot)$ at layer L. The first term of the right-hand side is a measure of how the loss change w.r.t. the j^{th} output, whiles the second term is a measure of how fast the activation function changes w.r.t. the weighted input to the node. Eq. (2.9) can easily be rewritten in vector-based form as

$$\delta^L = \nabla_u \mathcal{L} \odot \frac{\partial f_L(z^L)}{\partial z^L}, \tag{2.10}$$

with $\nabla_u \mathcal{L} = (\partial \mathcal{L}/\partial u_1^L, ..., \partial \mathcal{L}/\partial u_{p_L}^L)$. By back-propagating δ^L through the FCNN, the error for any arbitrary layer l can be derived according to

$$\delta^l = \left(\left(\mathbf{w}^{l+1}\right)^T \cdot \delta^{l+1}\right) \odot \frac{\partial f_l(z^l)}{\partial z^l}, \tag{2.11}$$

where $\mathbf{w}^{l+1}$ is the weight matrix for the $(l+1)^{\text{th}}$ layer, interconnecting layers $l \rightarrow (l+1)$. I.e. given the error δ^{l+1} and applying the transpose weight matrix, the error are moved backward through the network, see Fig. 2.8. By then taking the element-wise product with the node in layer l, it moves the error through the activation function, finally resulting in δ^l. This way, by computing δ^L using eq. (2.10), and subsequently applying eq. (2.11) continuously, errors are successfully computed all the way through the network: $\delta^L \rightarrow \delta^{L-1} \rightarrow ... \rightarrow \delta^0$.

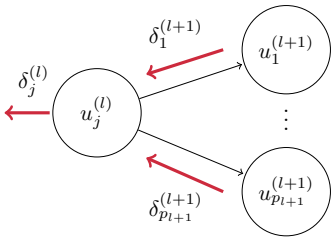


Figure 2.8: Backpropagation of errors through the network. Once the forward pass is evaluated for all output units, the errors $\delta_k^{(l+1)}, \forall k \in \{1, ..., p_{l+1}\}$ can be propagated backwards. $\delta^{(l+1)}$ corresponds to the errors that are propagated back from layer $(l+1)$ to layer l.

In particular, the error quantities for the parameters of the network are characterised by the gradient of the loss, s.t. component-wise, $\forall j \in \{1, ..., p_l\}$ and $\forall k \in \{1, ..., p_{l-1}\}$, they are described by [7, 56]

$$\frac{\partial \mathcal{L}}{\partial b_j^l} = \delta_j^l \qquad \text{and} \qquad \frac{\partial \mathcal{L}}{\partial w_{jk}^l} = u_k^{l-1} \delta_j^l. \tag{2.12}$$

2.1.4 Loss Functions

During the learning process the purpose is to minimize the error for each epoch and this is done using some optimization strategy, which is introduced in cf. § 2.1.5, aiming to minimize the *cost function*. Compared to the loss function, that is used for a single training example, the cost function is the *average loss* over the entire training dataset [27]. That is, the error comes from the loss function, which essentially is an objective function that can inform the network how well it is doing during training. Hence, it acts as a guide and tells the optimizer if it is moving in the right direction to reach the global minimum. Therefore, depending on what an ANN should achieve, the specific choice of loss function has to be very selective in order to gain the optimum result.

The *Kullback-Liebler (KL)* divergence is a measure of how a probability distribution differs from another distribution over the same variable, i.e. it measures how similar two distributions are. Let $P(x)$ and $Q(x)$ be two probability distributions of a discrete random variable x defined on the same probability space $\mathcal{X}$. That is, both $P(x)$ and $Q(x)$ sums up to 1, also $P(x) > 0$ and $Q(x) > 0$ for any $x \in \mathcal{X}$. Then, the KL-divergence of P from Q is defined by [15]

DISCRETE KULLBACK-LIEBLER DIVERGENCE

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}. \tag{2.13}$$

With other words; it is the expectation of logarithmic difference between P and Q w.r.t. P, i.e.[3]:

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}\big[\log P(x) - \log Q(x)\big]. \tag{2.14}$$

A KL-divergence of zero indicates that the distributions are identical, and the larger the value, the more the distributions differs from each others. Do note that the KL-divergence satisfies $D_{\mathrm{KL}}(P\|Q) \geq 0$, and only $D_{\mathrm{KL}}(P\|Q) = 0$ if, and only if, $P(x) = Q(x)$ [7].

The KL-divergence has its origins in information theory [47], and so it is a way to quantify exactly how much information is lost when approximating a

3. Recall the definition of the expected value for any function. Let Ω be the finite state space of all possible outcomes and let $g : \Omega \to \mathbb{R}$ be a probability distribution defined on Ω. Then, for any function $f : \Omega \to \mathbb{R}$, the expected value of f under distribution g is computed as $\mathbb{E}[f] = \mathbb{E}_{x\sim g}[f(x)] = \sum_{x\in\Omega} g(x)f(x)$. Hence, setting $g(x) = P(x)$ and $f(x) = \log P(x) - \log Q(x)$, then for some fixed $Q(x)$ eq. (2.13) can be rewritten in terms of expectations.

distribution. By using the definition of information entropy[4], eq. (2.13) can be re-expressed as the difference between two types of entropies:

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log P(x) - \sum_{x\in\mathcal{X}} P(x)\log Q(x) = H(P,Q) - H(P), \quad (2.15)$$

where $H(P)$ is the entropy of P and $H(P,Q)$ the cross-entropy of P and Q. Hence, the KL-divergence can be thought as a measure of entropy increase when $Q(x)$ is used to approximate the true distribution $P(x)$.

The KL-divergence is non-symmetric in the sense that $D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$. In deep learning the objective is to approximate the true distribution P of the target variables w.r.t. the input features, given some approximate distribution Q. Accordingly, since the KL-divergence is non-symmetric it can be done in two ways; by minimizing the *forward KL-divergence* ($D_{\mathrm{KL}}(P\|Q)$), or minimizing the *backward KL-divergence* ($D_{\mathrm{KL}}(Q\|P)$). For instance, in section § 4.2 the KL-divergence is used as part of the loss function in order to measure how closely two probability densities are related.

4. The information entropy of a discrete set of probabilities p is $H = -\sum_{i=1} p(i)\log p(i)$; also known as *Shannon entropy* [67].

2.1.5 Optimizers

In BP algorithms, differentiable loss functions are evaluated, and it is the optimizers job to update the network's weights in such a way that that the training loss is going to be minimized. The most common optimization techniques lies in the family of *gradient descent (GD)*. Below the GD, *stochastic gradient descent (SGD)* and *adaptive moment estimation (Adam)* techniques are introduced. Moreover, the derivation leading to eqs. (2.18) and (2.20) heavily follows [56].

Before thinking in terms of the neural network context, then suppose the loss function $\mathcal{L}(\mathbf{v})$ has parameters given by $\mathbf{v} \in \mathbb{R}^M$ in the M-dimensional parameter space. For purpose of visualization, it helps to imagine $\mathcal{L}$ as a function of only two variables, and Fig. 2.9 shows an example of a minimizing trajectory on the level curves for $\mathbf{v} \in \mathbb{R}^2$.

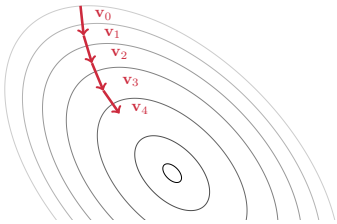


Figure 2.9: Visulaisation of level curves for a simple convex loss function in $\mathbf{v} \in \mathbb{R}^2$. Given the initial parameters $\mathbf{v}_0$, as the parameter position moves downhill in the level curves, the parameters are then iteratively updated untill it reaches the extrema — in turn succesfully minimizing $\mathcal{L}(\mathbf{v})$.

In such context the objective behind GD is therefore to find the set $\mathbf{v} \in \mathbb{R}^M$ that minimizes $\mathcal{L}$, and topologically it corresponds to finding the deepest valley point of the loss landscape. Starting off by defining some initial guess for the parameters, the local shape around the current position is found and the current position is moved to the next in the direction that reduces the loss the most given a step size. Such iteratively process is repeated until either a *local* or *global* minima is reached.

Let a change in the parameters be denoted as $\Delta\mathbf{v} \equiv (\Delta v_1, \Delta v_2, ..., \Delta v_M)^T$ and the first partial derivatives of the loss function defined by $\nabla\mathcal{L}(\mathbf{v}) \equiv (\partial\mathcal{L}/\partial v_1, \partial\mathcal{L}/\partial v_2, ..., \partial\mathcal{L}/\partial v_M)^T$. From calculus, it then follows that a change in $\mathcal{L}$ produced by a small change $\Delta\mathbf{v}$ is described by the approximation

$$\Delta\mathcal{L}(\mathbf{v}) \approx \nabla\mathcal{L}(\mathbf{v}) \cdot \Delta\mathbf{v} = \sum_{m=1}^{M} \frac{\partial\mathcal{L}}{\partial v_m}\Delta v_m. \tag{2.16}$$

The gradient $\nabla\mathcal{L}$ indicates the direction of maximum increase, but in the minimization problem the aim is to go downhill, and so it is necessary to insure that the gradient is negative and points in the direction in which the function decreases most rapidly. One method in guaranteeing that the direction for which the loss is reduced is to set

$$\Delta\mathbf{v} = -\eta\nabla\mathcal{L}, \tag{2.17}$$

where η is a small, positive parameter refereed to as the *learning rate*. Then, according to eq. (2.16) it yields that $\Delta\mathcal{L} \approx -\eta\nabla\mathcal{L} \cdot \nabla\mathcal{L} = -\eta\|\nabla\mathcal{L}\|^2$, where $\|\nabla\mathcal{L}\|^2 \geq 0$ such that $\Delta\mathcal{L} \leq 0$. Hence, in such way $\mathcal{L}$ will always decrease by an alternation of $\mathbf{v}$, but will never increase. Furthermore, by the displacement of $\mathbf{v}$, it follows component-wise that $v_i \longrightarrow v_{i+1} = v_i + \Delta v_i$, and by insertion of eq. (2.17) the parameters are iteratively updated according to

$$v_{i+1} = v_i - \eta\nabla\mathcal{L}(v_i).$$

In the NN setting, the loss function are parametrized for weights and biases, i.e. $\mathbf{v} = \{\mathbf{w}, b\}$. Consequently, the corresponding GD update rules are;

GRADIENT DESCENT

$$\begin{aligned} w_{i+1} &= w_i - \eta\nabla_{w_i}\mathcal{L} = w_i - \eta\frac{\partial\mathcal{L}}{\partial w_i}, \\ b_{i+1} &= b_i - \eta\nabla_{b_i}\mathcal{L} = b_i - \eta\frac{\partial\mathcal{L}}{\partial b_i}. \end{aligned} \tag{2.18}$$

I.e. until convergence, GD changes the weights and bias proportional to the negative of the gradient of the loss function times a step size w.r.t. the preceding weight or bias.

GD utilizes the entire training data before it updates. When the size of the training set is large, then one iteration of GD involves computations of a considerable number of gradients, which is computationally expensive [27]. To resolve this issue, the SGD method may be applied instead. It simply chooses a mini-batch consisting of S samples $\{x_1, x_2, ..., x_S\}$ that are randomly drawn from the data, and performs an update for each of those mine-batches. The overall gradient of the loss function is computed by averaging over the gradient of each training input, i.e. the estimation

$$\nabla\mathcal{L}(\mathbf{w}, b) \approx \frac{1}{S}\sum_{s=1}^{S} \nabla\mathcal{L}_{x_s}(\mathbf{w}, b), \quad \forall x_s \in \{x_1, ..., x_S\}. \tag{2.19}$$

Hence, the network parameters are component-wise updated according to

$$w_{i+1} = w_i - \frac{\eta}{S}\sum_{s=1}^{S} \nabla_{w_i} \mathcal{L}_{x_s} = w_i - \frac{\eta}{S}\sum_{s=1}^{S} \frac{\partial \mathcal{L}_{x_s}}{\partial w_i},$$
$$b_{i+1} = b_i - \frac{\eta}{S}\sum_{s=1}^{S} \nabla_{b_i} \mathcal{L}_{x_s} = b_i - \frac{\eta}{S}\sum_{s=1}^{S} \frac{\partial \mathcal{L}_{x_s}}{\partial b_i}, \tag{2.20}$$

STOCHASTIC GRADIENT DESCENT

where the sums are over all the training samples x_s in the current mini-batch. Subsequently a new mini-batch is randomly chosen and the procedure is repeated until the training inputs has been exhausted. This executes one epoch and a new training epoch can thereupon begin.

The calculated gradients for each mini-batches are but noise estimators of the true gradient of the whole training set [81]. But by repeatedly obtaining small noisy updates, the algorithm will eventually converge to a close enough good minima of the loss function. Small batch sizes usually require small learning rates in order to maintain stability, hence resulting in a long runtime [27]. Conversely, using larger batch sizes provides more accurate estimates of the true gradient and allows for larger learning rate, but the trade-off is that more memory is required during training.

Optimizers have configurable hyperparameters, and the learning rate is a crucial one. As one may have noticed from eqs. (2.18) and (2.20), η is proportional to the slope of $\mathcal{L}$, and it thus controls how fast the optimizer tries to minimize the loss function. Consider for instance Fig. 2.10. If η is set to small, then the algorithm will converge very slowly and it can happen that it get trapped in some bad local minima or get stuck at some flat region where the gradient is zero [27, 81]. On the other hand; if η is set to large, problems for converging to a minima at all will arise. Another aspect is that as training progresses and the error drops, the initial chosen value for the learning rate might become to large such that the algorithm might start overshooting the minimum.

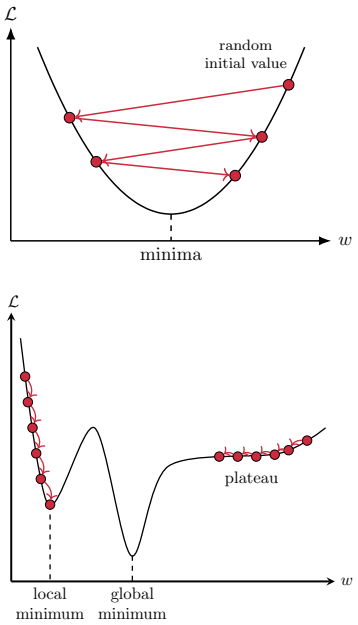


Figure 2.10: Incremental steps for random inital values are shown on a convex (upper) and a non-convex (lower) functions. If η is too large, the algortihm might just jump across the valley and possible even end up at a point higher than the intial one, making the algorithm diverge and failing to find a good solution. If η is to low, the algorithm has to go thorugh many iterations to converge, and it might not converge to the global minima but get trapped or stuck at other places.

The non-linearity of a NN typically causes the loss function being non-convex [27], which requires extra techniques during training. In order to address such issues, the adaptive optimizer Adam [39] is briefly mentioned. Adam is an extension to the SGD method, and has been designed so that the learning rate of the model automatically adjusts and reduces during the training phase. In particular, Adam has shown to be an efficient choice in deep learning scenarios [13], and consequently it has been chosen as the optimizer of the implementations in chapter § 6. Additionally, Adam is regarded to be fairly robust to the choice of hyperparameters [27], and a method for choosing the initial learning rate for the implementations follows in section § 6.2.1.

Besides utilizing such *adaptive learning rates* in order update the network weight more efficiently, Adam also employs *momentum* in favour to converge faster. Learning using the SGD strategy can be slow, but by introducing the

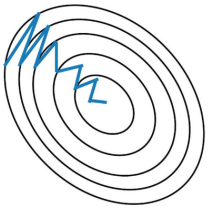
method of momentum, it helps accelerating the learning in the relevant direction and damping oscillations [27]. The effect of the momentum scheme is illustrated in Fig. 2.11. In order for the learning rate to adapt over time, ADAM employs two exponentially decaying averages of gradients, and computes bias-corrected estimates of them. Specifically, writing $\theta = \{\mathbf{w}, b\}$, the update rules of ADAM is [39]

ADAPTIVE MOMENT ESTIMATION

$$\begin{aligned} m_{i+1} &= \beta_1 m_i + (1-\beta_1)\nabla_\theta \mathcal{L}(\theta_i), \\ v_{i+1} &= \beta_2 v_i + (1-\beta_2)\nabla_\theta \mathcal{L}(\theta_i)^2, \\ \hat{m}_{i+1} &= m_{i+1}/(1-\beta_1^{i+1}), \\ \hat{v}_{i+1} &= v_{i+1}/(1-\beta_2^{i+1}), \\ \theta_{i+1} &= \theta_i - \alpha_i \hat{m}_{i+1}/(\sqrt{\hat{v}_{i+1}} + \varepsilon), \end{aligned} \tag{2.21}$$

where α is the learning rate, $\beta_1, \beta_2 \in [0, 1)$ the exponentially decay rates for the momentum estimates, $\varepsilon = 10^{-8}$ the default numerical stability parameter, and $m_0 = 0$, $v_0 = 0$ the initial 1^{th} and 2^{th} moment vectors, respectively.

Stochastic Gradient Descent **withhout** Momentum



Stochastic Gradient Descent **with** Momentum

Figure 2.11: Illustartion of SGD with and without the method of momentum. Momemtum helps to accelerate the convergence. Image source: *Gradient Descent Optimizers for Neural Net Training. D. Chang (2020).*

2.1.6 *Batch Normalization*

Batch normalization is an important concept first proposed in [37] and acts as a regularization technique. It has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks. The technique is implemented during training, where it calculates the mean and standard derivation of each input variable per batch, subsequently using these statistical values to perform normalization. I.e. letting $\mathcal{B}$ denote a mini-batch of size S of the entire training set, $\mathcal{B} = \{x_1, ..., x_S\}$, the mean and variance of $\mathcal{B}$ are respectively calculated as

$$\mu_{\mathcal{B}} = \frac{1}{S}\sum_{i=1}^{S} x_i, \qquad \sigma_{\mathcal{B}}^2 = \frac{1}{S}\sum_{i=1}^{S}\left(x_i - \mu_{\mathcal{B}}\right)^2. \tag{2.22}$$

Given a d-dimensional layer input, $\mathbf{x} = \left(x^{(1)}, ..., x^{(d)}\right)$, normalization is applied to each activation $x^{(k)}$ separately. That is

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_{\mathcal{B}}^{(k)}}{\sqrt{\sigma_{\mathcal{B}}^{(k)^2} + \varepsilon}}, \quad k \in \{1, ..., d\}, i \in \{1, ..., S\}, \tag{2.23}$$

where ε in the denominator is an arbitrarily small constant added for numerical stability. Finally, the following scale and shift transformation is applied in order to restore the representative power of the network

BATCH NORMALIZATION

$$\mathrm{BN}_{\gamma,\beta}\left(x_i^{(k)}\right) \equiv y_i^{(k)} = \gamma^{(k)}\hat{x}_i^{(k)} + \beta^{(k)}, \tag{2.24}$$

where the parameters β and γ are subsequently learned in the optimization process. As can be deduces from the equations above, batch normalization is unstable when using small batch sizes [37].

Consequently, each training iteration in the NN will be slower because of the extra normalization calculation during the forward pass and the additional hyperparameters to train during backpropagation. However, the benefit of the batch normalization is that the network will converge more quickly and thus overall training is faster [37]. Furthermore, batch normalization allows for higher learning rates without risk of divergence and reduces the need for dropout, which concept is introduced in section § 2.2.4.

2.2 CONVOLUTIONAL NEURAL NETWORK

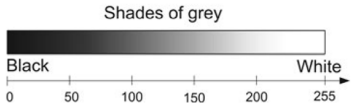


Figure 2.12: 8-bit pixel intensities falls between values 0 and 255. Usually 0 represents black and 255 white. Image source: *Hack Till Dawn. T. Mulc (2017).*

In this section, concepts from the field of *computer vision* are combined with practices from § 2.1 in order to introduce a NN specially designed to handle data with a grid-like topology, such as images. *Convolutional neural networks (CNNs)* falls under *supervised learning* as it takes an input image, process it, and classifies it under a certain category. § 2.2.1 serves as a motivation behind and overview of the architecture of a classical CNN. In § 2.2.2-§ 2.2.5 each individual layer and their operations will in turn be introduced and finally § 2.2.6 presents the concept of transpose convolution. But, before dealing with CNNs it is considerably to shortly present exactly what an image is and how computers interpret them.

Computers sees images as arrays of pixels and depends on the image *resolution*. Based on the resolution, it will percept a two-dimensional image as $I \in \mathbb{R}^{H \times W \times C}$, where H is the height, W the width, and C the number of color channels. If $C = 1$, there is talk about an grayscale image with pixel values usually given in 8-bit byte format, such that values spans in $[0, 255]$ [48]. The pixel value is simply a scalar that represents the brightness of the pixel, cf. Fig. 2.12. RGB (red, green, blue) images are represented by $C = 3$ and its pixel value is a vector combined by its three color planes [66], see Fig. 2.13.

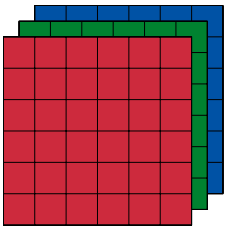


Figure 2.13: RGB images has $C = 3$ color chanels, and all 2D maps have same height and width. Each map represents the pixel intensities for its representive primary color. Each element in the full RGB grid are represented by a pixel, which is a combination of the three intensities of (red, green, blue).

2.2.1 *Overview: General Structure of a CNN*

CNNs are very similar to regular NNs from § 2.1 in the sense that they still have learnable weight and biases, activations functions and a loss function. The essential change is the assumption that inputs are images, which in turn allows the network to encode certain properties into the architecture.

Hence, it leads one to wonder how the human brain process images? Accordingly, it is convenient to develop an intuition for how the human visual

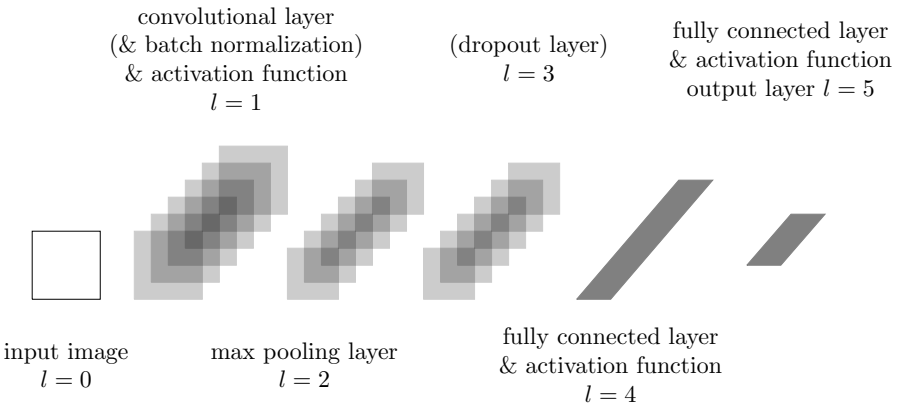


Figure 2.14: A visualization over the architecture of layers in a common CNN. Batch normalization and dropout layers are optional. Instead of max pooling, another subsampling method may be performed. The depth of the convolutional layer corresponds to the number of feature maps, and in this particular illustration $F = 6$.

processing system works. Processing in the brain is mostly hierarchical, and the visual cortex of the brain contains a wealth of neurons which are tuned to detect very specific stimuli [50]. Following the hierarchical organization, the brain detect increasingly more complex visual features. The artificial CNN is build upon this foundation, and mathematically *convolution* is used (as explained in cf. § 2.2.2). In order to capture increasingly abstract features, lower level features are convoluted to produce *feature maps* (or *activation maps*) in the next layer. It is such subsequent passing of input through various layers that extracts higher level patterns.

CNNs are typically composed of three types of layers (or building blocks) that are stacked together: convolution, pooling, and fully connected layers [66, 81]. The first two layers, convolution and pooling, performs the feature extractions, whereas a fully connected layer maps the extracted features into a final output, such as class scores. Additionally, dropout regularization is commonplace [75], and such layer follows after the pooling layer. Moreover, batch normalization can be performed in between the convolution and its subsequently activation function. Fig. 2.14 shows an illustrative example of such architecture.

The depth of the network corresponds to the number of filters, F, used for the convolution operation. It is the number of filters that controls the number of features that a convolutional layer will look for [81]. Hence, each feature map will extract and reflect one image characteristic.

2.2.2 Convolutional Layer

It is impractical to connect all neurons in a fully connected manner as is done in regular NNs, since images are high-dimensional inputs, hence containing many parameters. Instead, neurons are locally connected to a small region of the layer before it, such that the amount of parameters in the network is vastly reduced in the forward propagation [81]. The spatial extent of this

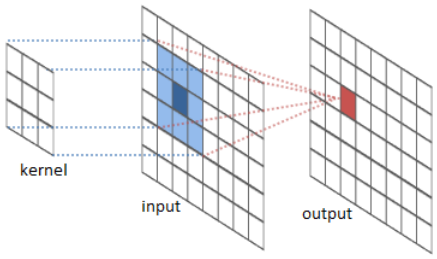


Figure 2.15: Visualization of a 2D discrete convolution between the kernel and input, producing the output that is the feature map. Each output pixel is computed by sweeping the kernel over the input and computing the weighted sum of its neighbors that lie within some specific window. Image source: *the River Trail documentation.*

connectivity is called the *local receptive field* for an rendered neuron in the feature map [56]. Similar to the recognition process on how human gazes at an object and recognizes it, the concept of locality is an important concept to images. Convolution is used to sweeping the kernel, and each connection learns a shared weight and an overall bias. While features are extracted on local perceptions, subsequent convolution layers will be comprised of a combination of the receptive fields of the earlier layers, enabling an expansion in the overall receptive field which gradually converts local features to global features.

The convolution operation is a linear operation that merges two signals. Given an *input* $x(i)$ and a *kernel* $w(a)$, discrete convolution is performed according to [27]

$$S(i) = (x * w)(i) = \sum_{a=-\infty}^{\infty} x(a)w(i-a), \tag{2.25}$$

resulting in a feature map $S(i)$. Following the approach of the local receptive field, then given a two-dimensional input, such as an image $I \in \mathbb{R}^{H\times W\times C}$, discrete convolution is performed with a two-dimensional kernel $K \in \mathbb{R}^{k_1\times k_2\times C}$ and bias $b \in \mathbb{R}^1$ according to [56]

$$S(i,j) = (K * I)(i,j) = b + \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} K_{m,n} I_{i-m,j-n}, \tag{2.26}$$

where a grayscale image is supposed for ease of notation. It follows from eq. (2.26) that the resulting feature map fulfills $S \in \mathbb{R}^{(H-k_1+1)\times(W-k_2+1)}$ given $I \in \mathbb{R}^{H\times W}$ and $K \in \mathbb{R}^{k_1\times k_2}$. The input could be either the input image or an input feature map. Fig. 2.15 visualizes how convolution is performed between the weight kernel and input for a small scale example. Flipping the kernel, such that the kernel is rotated 180°, yields the subsequent form of eq. (2.26)

$$S(i,j) = (K * I)(i,j) = b + \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} K_{m,n} I_{i+m,j+n}, \tag{2.27}$$

called the *cross-correlation*. Cross-correlation is conveniently more straightforward to implement numerically compared to convolution [27].

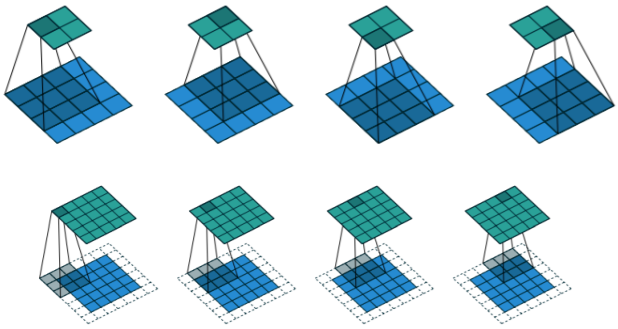


Figure 2.16: Upper: Convolving a weight kernel $K \in \mathbb{R}^{3\times3}$ over an $I \in \mathbb{R}^{4\times4}$ input using strides $S_t = 1$ and no padding, $P = 0$, results in a $S \in \mathbb{R}^{2\times2}$ feature map. Lower: Half padding, $P = 1$, and $S_t = 1$ convolution results in $S \in \mathbb{R}^{5\times5}$ given $K \in \mathbb{R}^{3\times3}$ and $I \in \mathbb{R}^{5\times5}$. Images from [22].

The convolution process (with flipped kernel) can be generalized in a neural network implementation as following. Let u_{ij} denote the outputs for layer l and let $u^0_{ij} = I$ be the elements of the input image. Then the weighted input to layer l is derived by [56]

$$z^l_{ij} = b^l + \sum_m \sum_n w^l_{m,n} u^{l-1}_{i+m,j+n}, \tag{2.28}$$

where $w^l_{m,n}$ are weight filters connecting layer l and $l-1$, and b^l the shared bias for layer l. Then, the weighted output at locations i, j for layer l is

$$u^l_{ij} = f_l(z^l_{ij}), \tag{2.29}$$

given an activation function $f_l(\cdot)$ in the l^{th} layer. Successive convolutional layers consists of a combination of the earlier layers local receptive fields. Hence, the deeper layers into the CNN will extract more abstract image features, i.e. patterns recognized by the convolution kernels becomes more complex and sparse.

Strides, S_t, indicates the number of pixels the kernel window shifts over the input at each step of convolution. E.g. setting $S_t = 1$ would ensure that no locations are missed in an image, as the kernel would successively move one index to derive the next element in the feature map.

Consider Fig. 2.16. In the upper case, the feature map resulting from the convolution is smaller in size compared to the input image I. However, such a dimension reducing might not be the sought one for the resulting feature map. Instead, a *zero padding* can be applied to the input, such that I is altered by padding with zeros around its border, yielding $I \longrightarrow \tilde{I}$. In the lower case in Fig. 2.16, a zero padding around the input has been applied such that the spatial dimension of the output has been preserved.

Formally, let P denote the zero padded pixel thickness around the original input. If given symmetric square input and kernel so $W = H$ and $k = k_1 = k_2$, then the spatial dimension for any output feature map $S \in \mathbb{R}^{O\times O}$ is

$$O = \frac{(W - k + 2P)}{s} + 1, \tag{2.30}$$

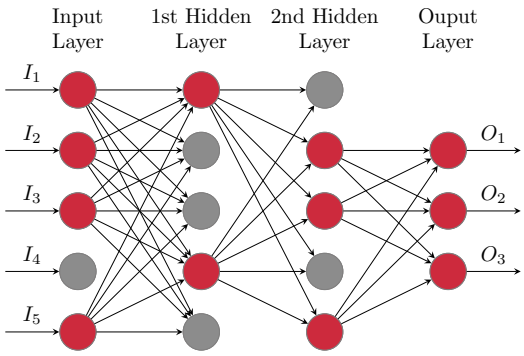


Figure 2.18: An example of what a feedforward FCNN could look like with dropout. Red circles represents activated neurons while gray circles represents deactivated neurons.

given same integer strides along both axes $s = s_1 = s_2$ so $S_t \in \mathbb{R}^{s \times s}$ [22].

2.2.3 Max Pooling Layer

A *pooling* layer may be periodically inserted in-between successive convolutional layers. Pooling is a downsampling from a convolutional layer, and its function is to reduce the amount of parameters while preserving the most prominent features. Pooling operations spatially reduce the size of feature maps by using some *pooling function* to summarize subregions. Hence, provided a kernel and a stride, pooling works by sliding a window across the input and feeding the content of each frame to a predefined operation, resulting in a reducing of information [22].

The most common downsampling operation is *max pooling* [56]. This type of pooling simply outputs the maximum activation of each disjoint subset. These dominant entities are what is propagated to the next layer in the CNN. Fig. 2.17 shows an illustrative example of a max pooling operation.

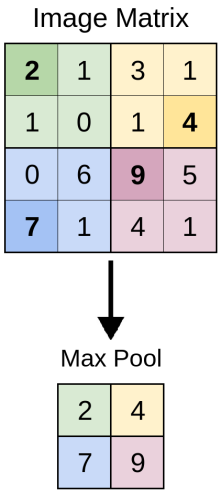


Figure 2.17: Illustative example of max pooling with 2×2 filters and strides $S_t = 2$. Image source: *Convolutional Neural Networks to Classify Sentences. A. G. Walters (2019).*

Pooling layers can be gotten rid of in favour of architectures that only consists of repeated convolutional layers with very few to no pooling layers [69]. Instead, using larger strides of the convolutional layers can precede the spatial dimensionality reduction accordingly. In particular, [60] has found that discarding pooling layers helps training good generative models, which practices have be applied to the implementations in § 6.

2.2.4 Dropout Layer

Dropout is a regularization method presented in [35]. The dropout neural network aim to increase a NN's ability to generalize properly and perform well with unseen data, hence preventing *overfitting*. The *dropout neural network* works by randomly dropping out a percentage of units for each training input, and in that way samples a *thinned* version of the original

network that it trains on. Hence, for a neural network with n units, there are in turn 2^n unique thinned neural networks over which weights are shared [70].

Given that a unit is retained with probability p during training, such a feedforward dropout network is formally formed by setting[5]

$$\begin{aligned} r_j^l &\sim \text{Bernoulli}(p), \quad 0 \leq p \leq 1,\ r_j^l \in \{0,1\}, \\ \tilde{\mathbf{u}}^l &= \mathbf{r}^l \odot \mathbf{u}^l, \\ z_j^{l+1} &= \mathbf{w}_j^{l+1} \cdot \tilde{\mathbf{u}}^l + b_j^{l+1}, \\ u_j^{l+1} &= f_j(z_j^{l+1}), \end{aligned} \tag{2.31}$$

5. A *Bernoulli distribution* takes the value of 1 with probability p and the value 0 with probability $q = 1 - p$, i.e. a set of possible outcomes of either 1 or 0.

using the notations from § 2.1.1. Neurons of the network is rendered *activate* by $r_j^l = 1$ and consequently *inactivate* by $r_j^l = 0$. A randomly thinned version of a feedforward FCNN is shown in Fig. 2.18. In [70] it has been shown that applying dropout layers in a CNN improves the generalization performance, and can outperform ordinary CNNs.

2.2.5 Fully Connected Layer

After multiple convolutional, subsampling (such as max pooling) and possible dropout layers, a fully connected layer completes the network [66]. The last feature map matrix will be flattened and converted to a vector and fed into the fully connected layer. In a classification task, the final output layer would be the probabilities of the inputs being in a particular class.

2.2.6 Transpose Convolution

Transpose convolution is not part of the regular CNN algorithm, nevertheless it is convenient to introduce, as it is a layer used in *generative modeling*. The transpose convolutional layer is usually carried out for *upsampling*, i.e. to generate an enlarged output feature map which has has spatial dimensions greater than that of the input.

Just like the standard convolutional layer, transposed convolutional layer is also defined by zero padding and stride. Fig. 2.19 visualizes the transpose convolutions to the convolutions in Fig. 2.16. The transposed convolutions are given inputs that are the results of direct convolutions applied on some initial input. The operation of the transpose convolution reconstructs the spatial dimensions of the initial input, however it does not guarantee to recover the input itself, as it does not act as the inverse to the convolutional operand [22].

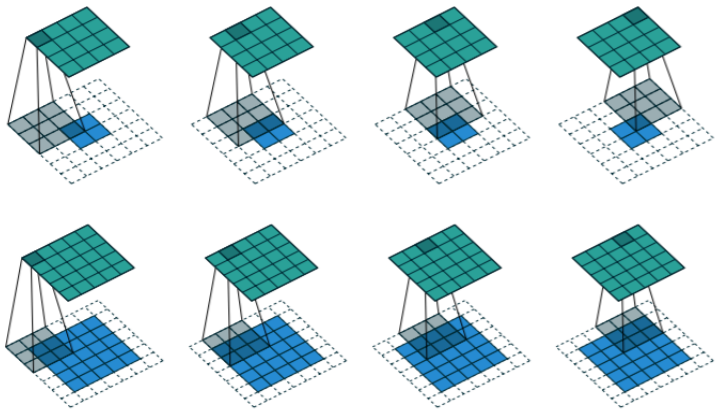


Figure 2.19: The original convolutions are seen in Fig. 2.16, respectively. Upper: The transpose convolution is applied on a $P = 2$ zero padded input $I' \in \mathbb{R}^{2\times2}$, given a kernel $K \in \mathbb{R}^{3\times3}$ using $S_t = 1$, resulting in a $S' \in \mathbb{R}^{4\times4}$ feature map. Lower: The transpose convolution using $S_t = 1$ and $P = 1$ given $I' \in \mathbb{R}^{5\times5}$ and $K \in \mathbb{R}^{3\times3}$ produces $S' \in \mathbb{R}^{5\times5}$. Images from [22].

Only the *fractional stride* $S_t = 1$ has been considered above. Conversely to belief, $S_t > 1$ involves inserting $z = s - 1$ zeros between each rows and columns of the input before applying a possible zero padding. The associated output feature map $S' \in \mathbb{R}^{O'\times O'}$ of the transposed convolution given input $I' \in \mathbb{R}^{H'\times W'}$ with $W' = H'$, is then described by [22]

$$O' = s\big(W' - 1\big) + k - 2P, \tag{2.32}$$

using a square kernel $K \in \mathbb{R}^{k\times k}$ and stride $S_t \in \mathbb{R}^{s\times s}$

To quantify eq. (2.32), consider the example in Fig. 2.20. Here, strides $S_t = 2$ has been considered given an kernel $K' \in \mathbb{R}^{3\times3}$ and input $I' \in \mathbb{R}^{3\times3}$ zero padded by $P = 1$. I.e. explicitly evaluating eq. (2.32) gives $O' = 2 \cdot (3 - 1) + 3 - (2 \cdot 1) = 5$, hence the output is a $S' \in \mathbb{R}^{5\times5}$ feature map. Notice, an input $I' \in \mathbb{R}^{3\times3}$ was given, but when adding the zeros in between it instead yields an input $I' \in \mathbb{R}^{5\times5}$. This input is hereafter zero padded by $P = 1$. Still utilizing the kernel $K' \in \mathbb{R}^{3\times3}$, but instead using strides $S_t = 1$, it would likewise result in an $S' \in \mathbb{R}^{5\times5}$ output as $O' = 1 \cdot (5 - 1) + 3 - (2 \cdot 1) = 5$.

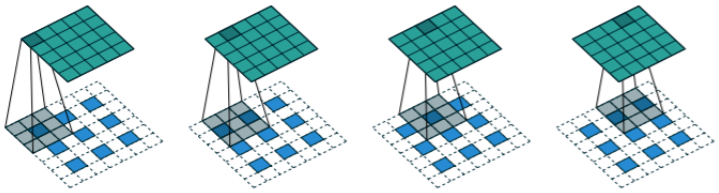


Figure 2.20: Given an $I' \in \mathbb{R}^{3\times3}$ input, zeros are inserted between each row and column, whereas $P = 1$ zero padding has been applied. The transpose convolution is then applied using $S_t = 1$ and kernel $K' \in \mathbb{R}^{3\times3}$, resulting in the output $S' \in \mathbb{R}^{5\times5}$. Image from [22].

CONTENTS

ABNORMALITY DETECTION SCENARIOS

3

In the literature a mix-up exists between the terminology and problem statements of the terms *outliers*, *anomalies* and *novelties*. The following definitions of the terms and their distinctions in § 3.1 are proposed in the survey [11]. Additionally, an overview of how each of their detection techniques are related to machine learning tasks is included.

3.1 WHAT ARE OUTLIERS, ANOMALIES AND NOVELTIES?

As known from the statistics literature, outliers are data points expected to be present in the dataset, cf. Fig. 3.1. Usually their occurrences are caused by unavoidable random errors or from systematic errors relating to how the data was sampled, such as human mistakes [1].

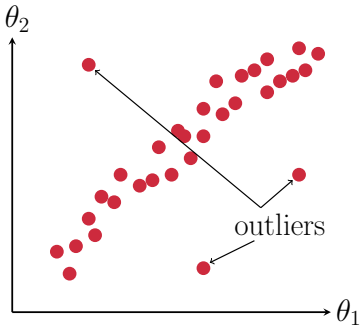


Figure 3.1: Outliers are deviated instances in a dataset without any clear pattern. In machine learning, a problem like this would be attacked as an unsupervised classification tasking to find and isolate outliers.

In a machine learning perspective an outlier detection task can be formalised as an unsupervised classification problem. Given a dataset, the objective is detecting the events that (highly) deviates from others and without a clear pattern.

Anomalies are data instances that stands out as being dissimilar to all others data observations and do not follow the rest of the pattern. They are outliers or other values that are not expected to exist [1]. In that sense, taking into account all normal data, anomalies have a very low probability of occurrence.

According to [11], an anomaly detection is given a (highly) *unbalanced* training dataset, consisting of two categories: normal, $\mathcal{N}$, and anomaly, $\mathcal{A}$. The model is then trained as a supervised classification problem, and learns to recognise anomalies by teaching it what a normal instance looks like compared to what an anomaly looks like. However, since anomalies are so scarce it poses a problem for this detection technique due to the lacking samples of $\mathcal{A}$. During training the algorithm never gets sufficient look at this underlying class, hence the results will not be optimized for the unbalanced class.

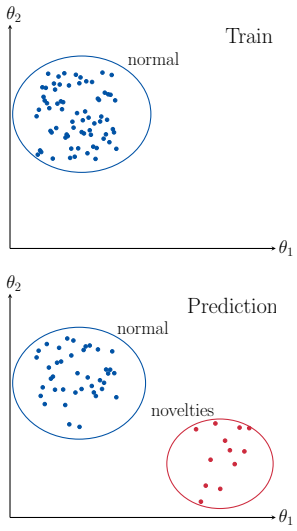


Figure 3.2: Example of static novelty detection. A supervised classification with only one class for training — the normal instances. In the prediction stage the model will recieve new samples, whereas instances are classified as either normal or novel. Some sort of decision boundary can for example be used in order to isolate the behaviors.

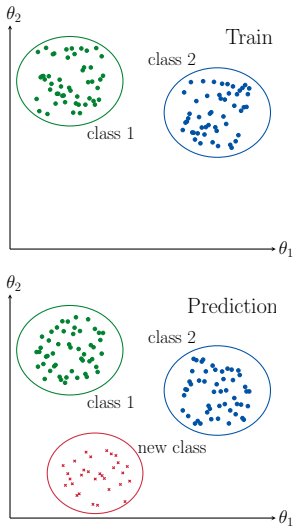


Figure 3.3: Example of dynamic novelty detection. The model is trained given some known number of classes. In the prediction stage, new instances will either be classified among known classes or as novel, for which it is kept in a buffer and may emerge as a new class.

Like outliers and anomalies, novelties are also a form of abnormalities. But compared to an anomaly detection, where the model learns to find patterns that do not adhere to what is considered as normal using two classes, in novelty detection the model learns using a dataset that contains only one class. The survey [11] distinguish between two different types of novelty detection problems in machine learning. They differentiate between what they call *static* and *dynamic* novelty learning scenarios.

In a static novelty detection, the model learns from a dataset composed singularity by the normal data $\mathcal{N}$, see Fig. 3.2. In that sense, it can be thought of as a supervised classification with only one class represented in the training data. The objective is to determine whether observations it has never seen before fit within the exposed dataset, such that new instances can be predicted either as normal (expected patterns) or novel (unexpected patterns). For instance, as elaborated in § 6.3.3, it will be seen that novelties can be identified and classified by computing a so-called *anomaly score*. Based upon this metric, some appropriate threshold can be set, which in turn is used to determine what is suspicious and whatnot.

Alternatively, a dynamic novelty detection is a supervised classification problem that has unknown number of classes for which the class variable dynamically changes during the classification process. The objective is to discover new emerging classes, and when new instances is received the predictor has to either classify them among the current classes or, if instances are not similar to any current known class, store them in a *buffer* that is considered a candidate for a potential new class [11]. A new class then may emerge if the buffer is considered "full". Hence, the learning scenario of a dynamic novelty detection is given a settled number of classes in its training phase, after which new classes may emerge, disappear, reappear and drift in the prediction stage, cf. Fig. 3.3.

Specifically, the key difference between what is called anomaly detection and novelty detection lays in the methods for which they are utilized in training. In anomaly detection the training dataset contains both normal and abnormal instances, whereas in novelty detection the data only consists of normal samples. In particular, the terms for "anomaly" and "novelty" are interchangeable and mixed-up in the literature, and bottom line is that both types are in fact abnormal observations. Therefore, in the remaining thesis these two terms will be used interchangeable and undifferentiated. But, bear in mind that in reality the applications found in § 6 all practices what is termed novelty detections in [11].

Part II

AUTOMATIC ANOMALY DETECTION

CONTENTS

DEEP ANOMALY DETECTION TECHNIQUES

4

> *The key is to let computers do what they are good at, which is trawling these massive data sets for something that is mathematically odd. And that makes it easier for humans to do what they are good at — explaining those anomalies.*
>
> —Daniel Bruhl

This chapter serves as an introduction to the algorithms applied in chapter § 6 using the data described in chapter § 5. *Deep anomaly detection (DAD)* techniques, as described in the survey [12], includes some common operation procedures and faced challenges. They aim to learn hierarchical discriminative features from the training data, and subsequently some well-defined boundary between normal and anomalous behaviours needs to be selected.

Possible deep learning techniques that are able to detect anomalies using images are the *convolutional autoencoder (CAE)* and the generative model *convolutional variational autoencoder (CVAE)*. The common idea behind these two approaches is having the *encoder* accept the input data, compress it into an lower dimensional *latent space* representation, and letting the *decoder* reconstruct the input data from that space. By letting the neural network train on a single specific class of data, then when applying anomalous images as test images the reconstructed images are expected to have a larger error since the network have never seen such kind of sample data before. By defining an *anomaly score*, that is described in section § 6.3.3, some threshold are determined in order to distinguish between normal images and anomaly images. Hence, it is possible to detect abnormal behaviours using such pipelines.

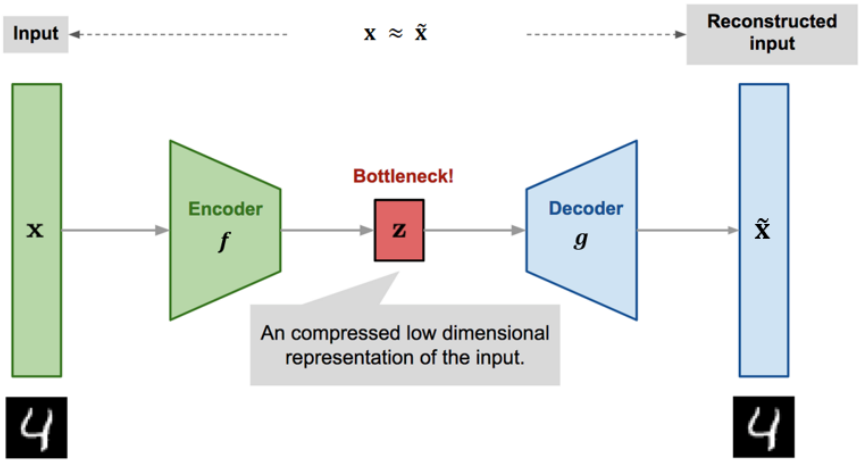


Figure 4.1: Architecture of the CAE model. The encoder takes an original image sample, returns a single data point in the latent space, which is then passed into the decoder that reconstruct the image. Image source: *Autoencoder to Beta-VAE. Weng, L. (2018).*

4.1 CONVOLUTIONAL AUTOENCODERS

The fundamental and most common unsupervised deep architectures used in DADs are the autoencoders [12]. This section thereby serves as an introduction to the framework of the classic CAE.

An autoencoder is a neural network that is trained in an unsupervised way and attempts to reconstruct the original input as its output, such that $\mathbf{x} \mapsto \tilde{\mathbf{x}}$. It consists of an *encoder* network that compress the input image into a lower dimensional *latent code*, $f : \mathbb{R}^n \to \mathbb{R}^p$, $f(\mathbf{x}) \to \mathbf{z}$. Subsequently, a *decoder* network use this representation to recover a reconstruction with same dimension as the input, $g : \mathbb{R}^p \to \mathbb{R}^n$, $g(\mathbf{z}) \to \tilde{\mathbf{x}}$. Specially, to learn a hierarchical feature representation, the encoder is constructed upon convolutional layers as for the CNN (as outlined in section § 2.2), whilst the decoder follows an inverse architecture of the CNN. The schematic architecture of a CAE is visualized in Fig. 4.1.

Hence, the encoder network are structured to take an input image and compress it into a different feature representation; an lower-dimensional code, or embedding, of the input. In the process useful properties and features of the input are extracted and learned without the need for the training data to be labelled. Subsequently, the decoder network aims to reconstruct the original input as precisely as possible by decompressing the coded representation. The aim of the encoder is to keep maximum information of the input, and the decoder to reconstruct with minimum error.

As formally defined in [3], the objective is to learn the functions that minimize the loss function, which for the CAE typically is the *mean squared error (MSE)* between the input and output entities:

LOSS FUNCTION FOR AUTOENCODER

$$\mathcal{L}_{\text{AE}} = \mathcal{L}\big(\mathbf{x}, g \circ f(\mathbf{x})\big) = \frac{1}{N} \sum_{i=1}^{N} \Big(x^{(i)} - g\big(f(x^{(i)})\big)\Big)^2, \tag{4.1}$$

where $g\big(f(\mathbf{x})\big) = \tilde{\mathbf{x}}$. The CAE is supposed to learn an approximation to the identity map on the data, and since 2D input images can be transformed to 1D vectors of length N, and as the output has the same dimension as the input, the "distance" between the original and reconstructed sample can be quantified by the MSE metric. Hence, the MSE is a well-suited loss function for the CAE as it is a measure of dissimilarity between input and output.

CAEs are often used for dimensionality reduction purposes in order to discover more efficient and compressed representations of the data. In the special case where a CAE is trained using the MSE criterion and is comprised of a single fully-connected hidden layer, as well where all non-liner operations are dropped in favour for linear ones, then the latent representation of the CAE is nearly equivalent to the *principal component analysis (PCA)* [58]. Because, in this case the CAE is searching for the best linear subspace to project data with as little information loss as possible, just as the PCA does. Hence, the CAE is a generalization of PCA, but while PCAs are restricted to linear dimensionality reduction, CAEs can enable both linear and non-linear transformations [4].

The deterministic CAEs are not designed to reconstruct input images exactly, but only approximately [27]. No matter how the latent space is organized, the CAE is solely trained to minimize the loss function. Hence, the learned latent space of an CAE is not imposed to be organized following some predefined distribution and its structure is not regular enough for the model being able to producing new data by a generative process [81]. That is, the latent space lacks continuity which prevents interpolations between training points, thus preventing its generative ability.

The CVAE is topologically similar to the CAE, but instead of learning a function that solely represent the training data, it is a generative model that aims to learn parameters of a predefined probability distribution that generates the input data. If trained successfully, new data that resembles the training data can be generated by sampling new points from the learned latent distribution and feeding a point though the decoder part of the network which produces an new artificial image. CVAEs thus models the latent space probabilistically, enabling the production of new content, since new samples that looks like the ones in the training data can be created.

4.2 CONVOLUTIONAL VARIATIONAL AUTOENCODERS

A CVAE is essentially a stochastic generalization of the CAE [40]. CVAEs combines the research areas of deep learning and *probabilistic modeling*, hence it falls under the domain of probabilistic generative models. In general, it is known as a *latent variable model*.

The generative process of the CVAE is reviewed in § 4.2.1. The simple Gaus-

sian case is defined in § 4.2.2, and a trick necessary for training the model is introduced in § 4.2.3. Having explained the probabilistic model interpretation of the CVAE, its deep learning interpretation is summarized in § 4.2.4.

4.2.1 Probabilistic Model Interpretation

From a probabilistic perspective the CVAE is described by two stochastic entities; the observable set $X = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ and unobservable latent set $Z = \{\mathbf{z}^{(i)}\}_{i=1}^{N}$ consisting of N independent and identically distributed samples of variable $\mathbf{x} \in \mathbb{R}$ [40].

The CVAE generates $\mathbf{x}$ by sampling from the distribution of $\mathbf{z}$, as shown partly by the solid lines in Fig. 4.2. The joint distribution expressed by this model is given as[1]

$$p_{\theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z}). \tag{4.2}$$

I.e. each local latent variable is related to its corresponding local observation though the *likelihood* $p_{\theta}(\mathbf{x}|\mathbf{z})$, and for that reason it is also known as the *probabilistic decoder*. Latent variables are sampled from a *prior density* $p_{\theta}(\mathbf{z})$, which describes the original distribution of $\mathbf{z}$. In a typical instance of the CVAE, there is only a single layer of latent variables, and its distribution is usually an isotropic centred Gaussian, i.e. $p_{\theta}(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ [40].

Accordingly, this framework describes a generative process using the following procedure

$$\begin{aligned} &\text{Draw } \mathbf{z}^{(\mathbf{i})} \sim p_{\theta}(\mathbf{z}), \\ &\text{Draw } \mathbf{x}^{(\mathbf{i})} \sim p_{\theta}(\mathbf{x}|\mathbf{z}^{(\mathbf{i})}). \end{aligned}$$

Hence, the distribution of observed variables are generated and governed by the hidden lower-dimensional latent variables. Consequently, the true parameter values, as well as the hidden lower-dimensional latent variables, are unknown. Specifically, the *maximum likelihood estimator (MLE)* is usually used to estimate the *generative parameters*, $\boldsymbol{\theta}$ [40].

Having specified the generative process, approximate inference is performed regarding the hidden latent variables and generative parameters [40]. Now, given an observed data example $\mathbf{x}$ the goal is to understand what possible values of the hidden variable $\mathbf{z}$ were responsible for it — i.e. the task is manifested into a probabilistic *inverse problem*.

1. If $p(x, y)$ is the value of the joint probability distribution of the discrete random variables $\mathcal{X}$ and $\mathcal{Y}$ at (x, y), and $p(y)$ is the value of the marginal distribution of $\mathcal{Y}$ at y, then $p(x|y) \equiv \frac{p(x,y)}{p(y)}$ for each x within the range of $\mathcal{X}$ given $\mathcal{Y} = y$, and provided that $p(y) \neq 0$, is called the conditional probability function.

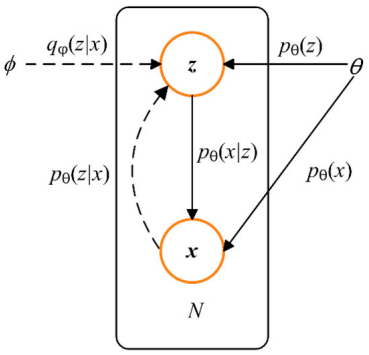


Figure 4.2: The directed probability graph model for the variational autoencoder. Solid lines denotes the generative model, whilest dashed lines denote the variational (approximate) interference. Image source: *A Variational Stacked Autoencoder with Harmony Search Optimizer for Valve Train Fault Diagnosis of Diesel Engine. Chen K, et al. (2019).*

In particular, the objective is to compute *the posterior* $p_\theta(\mathbf{z}|\mathbf{x})$, i.e. the conditional density of $\mathbf{z}$ given $\mathbf{x}$. Bayes' theorem[2] can be used to find the expression for such *probabilistic encoder*:

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})}, \tag{4.3}$$

and where its directed graphical model is seen partly in Fig. 4.2. The *marginal likelihood* (or *evidence*), $p_\theta(\mathbf{x})$, is the distribution of $\mathbf{x}$ that needs to be generated, and is attained by marginalizing out the latent variables from the joint probability distribution, so that

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})\mathrm{d}\mathbf{z} = \int p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})\mathrm{d}\mathbf{z}. \tag{4.4}$$

However, evaluation of this evidence integral requires exponential time to compute and it follows that the posterior cannot be evaluated in closed-form, hence it is intractable [8].

Consequently, it is necessary to somehow approximate the posterior. One option would be to estimate the objective via sampling-based techniques such as *Markov chain Monte Carlo* or *Monte Carlo expectation-maximization* algorithms[3]. However, such techniques are ruled out for being inapplicable in this setting due to them being computationally expensive when N is large [40]. An alternative approach to approximate Bayesian inference would be using deterministic approximation techniques, such as *variational interference* [38].

Variational inference is used to circumvent this intractability by getting rid of the integration. It formulates inference as an optimization problem, and seeks an *approximate posterior* $q_\phi(\mathbf{z}|\mathbf{x})$ closets to the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ in order to make the interference computationally feasible.

The KL divergence from section § 2.1.4 is used as a measure of dissimilarity between the two distributions[4]:

$$D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})\big) = \mathbb{E}_{q_\phi}\big[\log_e q_\phi(\mathbf{z}|\mathbf{x}) - \log_e p_\theta(\mathbf{z}|\mathbf{x})\big], \tag{4.5}$$

which quantifies how much information is lost when $q_\phi(\mathbf{z}|\mathbf{x})$ represents $p_\theta(\mathbf{z}|\mathbf{x})$. Consequently, the approximate posterior is parameterized by *variational parameters*, ϕ, and the problem is rendered to the following optimization problem [8]

$$\phi^* = \arg\min_\phi D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})\big). \tag{4.6}$$

2. Given events A and B, $A \neq B$, *Bayes' Theorem* is stated by

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Here $P(A)$ and $P(B) \neq 0$ are marginal probabilities, whiles $P(A|B)$ and $P(B|A)$ are conditional probabilities.

3. Such methods for sampling from a probability distribution will not be discussed, as they are not relevant for the work in this thesis.

4. For the sake of brevity $\mathbb{E}_{\mathbf{z}\sim q_{\phi(\mathbf{z}|\mathbf{x})}} \equiv \mathbb{E}_{q_\phi}$ is defined.

Hence, it seeks to find the best approximation that minimize the KL divergence to the exact posterior, such that $q_\phi(\mathbf{z}|\mathbf{x})$ is "closets" to $p_\theta(\mathbf{z}|\mathbf{x})$ in some sense.

Accordingly, applying Bayes' theorem on $p_\theta(\mathbf{z}|\mathbf{x})$ and using the definition of conditional probability, eq. (4.5) can be simplified as following:

$$\begin{aligned} D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})\big) &= \mathbb{E}_{q_\phi}\big[\log_e q_\phi(\mathbf{z}|\mathbf{x})\big] - \mathbb{E}_{q_\phi}\big[\log_e p_\theta(\mathbf{z}|\mathbf{x})\big] \\ &= \mathbb{E}_{q_\phi}\big[\log_e q_\phi(\mathbf{z}|\mathbf{x})\big] - \mathbb{E}_{q_\phi}\left[\log_e \frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{x})}\right] \\ &= \mathbb{E}_{q_\phi}\left[\log_e \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x},\mathbf{z})}\right] + \mathbb{E}_{q_\phi}\big[\log_e p_\theta(\mathbf{x})\big] \\ &= \mathbb{E}_{q_\phi}\big[\log_e q_\phi(\mathbf{z}|\mathbf{x}) - \log_e p_\theta(\mathbf{x},\mathbf{z})\big] \\ &\quad + \log_e p_\theta(\mathbf{x}), \end{aligned} \tag{4.7}$$

since $\log_e p_\theta(\mathbf{x})$ is a constant that can break out of the expectation. Unfortunately, from eq. (4.7) it can be seen that the KL divergence also depends on the intractable $p_\theta(\mathbf{x})$, which means the minimization problem described by eq. (4.6) is infeasible, because the intractable marginal likelihood cannot be written down exactly. Instead, an alternative tractable objective function is found and maximized, such that the KL divergence in turn is minimized indirectly.

This trick is done by finding a *variational lower bound* on the log marginal likelihood. Ergo, contemplate the *Jensen's inequality* that is given by $g\big(\mathbb{E}[X]\big) \geq \mathbb{E}\big[g(X)\big]$ for any concave function $g(\cdot)$. Accordingly, using concavity of the natural logarithm, the log marginal likelihood is

EVIDENCE LOWER BOUND (ELBO)

$$\begin{aligned} \log_e p_\theta(\mathbf{x}) &= \log_e \int p_\theta(\mathbf{x},\mathbf{z})\mathrm{d}\mathbf{z} = \log_e \int p_\theta(\mathbf{x},\mathbf{z})\frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\mathrm{d}\mathbf{z} \\ &= \log_e \mathbb{E}_{q_\phi}\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \geq \mathbb{E}_{q_\phi}\big[\log_e p_\theta(\mathbf{x},\mathbf{z}) - \log_e q_\phi(\mathbf{z}|\mathbf{x})\big] \\ &\triangleq \mathrm{ELBO}\big(\mathbf{x};\boldsymbol{\theta},\boldsymbol{\phi}\big), \end{aligned} \tag{4.8}$$

where the right-hand side is known as the *evidence lower bound (ELBO)*. Upon inspection with eq. (4.7), the ELBO is related to the KL divergence according to

$$D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})\big) = -\mathrm{ELBO}\big(\mathbf{x};\boldsymbol{\theta},\boldsymbol{\phi}\big) + \log_e p_\theta(\mathbf{x}). \tag{4.9}$$

Importantly, the ELBO is a lower bound to the log marginal likelihood. I.e. as $\log_e p_\theta(\mathbf{x})$ is independent of $\boldsymbol{\phi}$ it can be disregarded as a constant, such that minimizing the KL divergence w.r.t. to $\boldsymbol{\phi}$ amounts to maximizing the ELBO. Specifically, the gap between the ELBO and the marginal log likelihood is called the *tightness* of the lower bound and is defined by

$D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})\big) \geq 0$ [41]. The better $q_\phi(\mathbf{z}|\mathbf{x})$ approximates $p_\theta(\mathbf{x}|\mathbf{z})$, the smaller the Kullback-Liebler (KL) divergence gap is — see Fig. 4.3 for a simple illustration. Only if $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{x}|\mathbf{z})$, the ELBO and $\log_e p_\theta(\mathbf{x})$ coincide and the bound is tight.

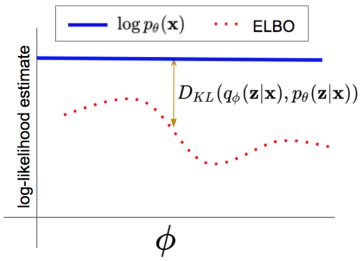


Figure 4.3: Illustration for the KL divergence gap between $\log_e p_\theta$ for a point $\mathbf{x}^{(i)}$ and the corresponding ELBO for any value ϕ (with fixed θ). The better $q_\phi(\mathbf{z}|\mathbf{x})$ approximates $p_\theta(\mathbf{x}|\mathbf{z})$ in terms of eq. (4.9), the thighter the gap is. Image adapted from: *Variational Rejection Sampling, Grover, A. (2018).*

Furthermore, recall that the primary goal of the generative process was to estimate θ^* through maximum likelihood estimation. It follows that maximizing ELBO w.r.t. to θ approximately maximizes $\log_e p_\theta(\mathbf{x})$ [41]. Hence, the original optimization problem described by eq. (4.6) can be revised into

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \mathrm{ELBO}\big(\mathbf{x};\theta,\phi\big). \tag{4.10}$$

That is, the variational parameters are simultaneously optimized with the generative parameters when maximizing the lower bound.

Moreover, the ELBO can be rearranged and decomposed into two terms according to

$$\begin{aligned}
\mathrm{ELBO}\big(\mathbf{x};\theta,\phi\big) &= \mathbb{E}_{q_\phi}\big[\log_e p_\theta(\mathbf{x},\mathbf{z}) - \log_e q_\phi(\mathbf{z}|\mathbf{x})\big] \\
&= \mathbb{E}_{q_\phi}\big[\log_e p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}) - \log_e q_\phi(\mathbf{z}|\mathbf{x})\big] \\
&= \mathbb{E}_{q_\phi}\big[\log_e p_\theta(\mathbf{x}|\mathbf{z})\big] - \mathbb{E}_{q_\phi}\big[\log_e q_\phi(\mathbf{z}|\mathbf{x}) - \log_e p_\theta(\mathbf{z})\big] \\
&= \mathbb{E}_{q_\phi}\big[\log_e p_\theta(\mathbf{x}|\mathbf{z})\big] - D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\big) \\
&\triangleq -\mathcal{L}_{\mathrm{rec}} - \mathcal{L}_{\mathrm{KL}}.
\end{aligned} \tag{4.11}$$

Recall eqs. (2.13)-(2.15). It can be deduced that the reconstruction term $\mathbb{E}_{q_\phi}\big[\log_e p_\theta(\mathbf{x}|\mathbf{z})\big] = -H\big(q_\phi(\mathbf{z}|\mathbf{x}), p_\theta(\mathbf{x}|\mathbf{z})\big)$ is actually the cross-entropy between the approximate encoder and the decoder. The reconstruction term encourages the model to be able to reconstruct the data accurately whiles the regularization term, $\mathcal{L}_{\mathrm{KL}}$, penalizes posterior approximations that are too far from the prior.

As a result, the ELBO for a single data point $\mathbf{x}^{(i)}$ can be parametrised by [40]

$$\begin{aligned}
\mathrm{ELBO}\big(\mathbf{x}^{(i)};\theta,\phi\big) &= \sum_{i=1}^{N} \mathrm{ELBO}\big(\mathbf{x}^{(i)};\theta,\phi\big) \\
&= \mathbb{E}_{q_\phi}\big[\log_e p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\big] - D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})\|p_\theta(\mathbf{z})\big).
\end{aligned} \tag{4.12}$$

4.2.2 *CVAE in a Strictly Gaussian Case*

Eq. (4.11) can be applied to just about any distribution, and the task is now to evaluate the closed-form of the KL term and the reconstruction term. Supposed that the prior is a standard multivariate Gaussian, and assume

that the likelihood as well as the posterior and its approximation follows a multivariate isotropic Gaussian with diagonal covariance, according to[5]

5. For the sake of brevity the subscripts for $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ for the mean and variances will be omitted later on.

$$p_{\boldsymbol{\theta}}(\mathbf{z}) \equiv \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4.13}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(i)}) \equiv \mathcal{N}\big(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}^{(i)}), \boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\mathbf{z}^{(i)}) \cdot \mathbf{I}\big), \tag{4.14}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)}) \equiv \mathcal{N}\big(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}) \cdot \mathbf{I}\big), \tag{4.15}$$

where $\boldsymbol{\mu}(\mathbf{x}^{(\mathbf{i})})$ and $\boldsymbol{\sigma}^2(\mathbf{x}^{(\mathbf{i})})$ are the mean and variance for data point $\mathbf{x}^{(i)}$, respectively. Note that this particular prior has no parameters $\boldsymbol{\theta}$.

The imposed prior distribution assumed over the latent space acts as a regularizer, enforcing the learned latent representations to follow a profound structure. The prior distribution thus gives significant control over how one want to model the latent distribution, and changing the assumed prior would thus change the characterized data distribution.

Suppose J is the dimensionality of the latent variable, such that $\mathbf{z} \in \mathbb{R}^J$, and let i denote the data point at which evaluation occurs. As shown in [40] and given all these assumptions, evaluating of the KL divergence then reduces to:

REGULARIZATION LOSS GIVEN THE GAUSSIAN CASE

$$\begin{aligned} -D_{\mathrm{KL}}\big(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)}) \| p_{\boldsymbol{\theta}}(\mathbf{z})\big) &= -D_{\mathrm{KL}}\big(\mathcal{N}\big(\boldsymbol{\mu}(\mathbf{x}^{(i)}), \boldsymbol{\sigma}^2(\mathbf{x}^{(i)}) \cdot \mathbf{I}\big) \| \mathcal{N}(\mathbf{0}, \mathbf{I})\big) \\ &= \frac{1}{2} \sum_{j=1}^{J} \Big(1 + \log_e \big(\sigma_j^{(i)}\big)^2 - \big(\mu_j^{(i)}\big)^2 - \big(\sigma_j^{(i)}\big)^2\Big). \end{aligned} \tag{4.16}$$

The cross entropy term can be estimated using Monte Carlo estimators of exceptions for large enough samples [40]. The likelihood is assumed to be an isotropic Gaussian, and not a Bernoulli distribution (which is otherwise the usual assumption when given binary data). I.e. using eq. (4.14) the expectation can be decomposed according to

$$\mathbb{E}_{q_{\boldsymbol{\phi}}}\big[\log_e \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{z}), \boldsymbol{\sigma}^2(\mathbf{z}) \cdot \mathbf{I}\big)\big] \simeq \frac{1}{N} \sum_{i=1}^{N} \log_e \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{z}^{(i)}), \boldsymbol{\sigma}^2(\mathbf{z}^{(\mathbf{i})}) \cdot \mathbf{I}\big).$$

In accordance with [19], the log probability of $\mathbf{x}$ is then proportional to the negative squared Euclidean distance between reconstruction $\tilde{\mathbf{x}}$ and ground truth $\mathbf{x}$. And as shown by e.g. [27] and [25], the reconstruction loss yields the negative mean squared Euclidean distance, or MSE:

RECONSTRUCTION LOSS GIVEN THE GAUSSIAN CASE

$$\mathbb{E}_{q_{\boldsymbol{\phi}}}\big[\log_e \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{z}^{(i)}), \boldsymbol{\sigma}^2(\mathbf{z}^{(i)}) \cdot \mathbf{I}\big)\big] = -\frac{1}{N} \sum_{i=1}^{N} \big(x^{(i)} - \tilde{x}^{(i)}\big)^2, \tag{4.17}$$

where the reconstructed variance has been fixed, $\boldsymbol{\sigma}(\mathbf{z}) = 1, \forall \mathbf{z}$, and $\tilde{\mathbf{x}} = \boldsymbol{\mu}(\mathbf{z})$ represents the mean value of the probabilistic decoder.

4.2.3 Reparameterization Trick

In order for the model to learn back-propagation is required, i.e. the gradients of the ELBO w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ is needed. In particular, $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(\mathbf{i})})}\left[\log_e p_{\boldsymbol{\theta}}(\mathbf{x}^{(\mathbf{i})}, \mathbf{z}) - \log_e q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(\mathbf{i})})\right]$ needs to be evaluated [53]. The gradient w.r.t. $\boldsymbol{\theta}$: $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(\mathbf{i})})}\left[\nabla_{\boldsymbol{\theta}} \log_e p_{\boldsymbol{\theta}}(\mathbf{x}^{(\mathbf{i})}, \mathbf{z})\right] \simeq \nabla_{\boldsymbol{\theta}} \log_e p_{\boldsymbol{\theta}}(\mathbf{x}^{(\mathbf{i})}, \mathbf{z})$, is immediate and can be estimated by Monte Carlo sampling [41]. On the other hand, the gradient w.r.t. $\boldsymbol{\phi}$: $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(\mathbf{i})})}\left[\left(\log_e p_{\boldsymbol{\theta}}(\mathbf{x}^{(\mathbf{i})}, \mathbf{z}) - \log_e q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(\mathbf{i})})\right)\nabla_{\boldsymbol{\phi}} \log_e q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(\mathbf{i})})\right]$, is the problematic one as its estimator exhibits high variance, thus being impractical in this setting [53].

Accordingly, the current issue is that the network involves a sampling step, as $\mathbf{z}$ is drawn from a distribution and is not a function of $\boldsymbol{\phi}$, and there is no way to differentiate through a stochastic node. Instead, the *reparameterization trick* [40] is used to facilitate back-propagation. The stochastic part is moved into a branch of the network, such that $\mathbf{z}$ is written as a deterministic transformation of a simpler random auxiliary noise variable, $\boldsymbol{\epsilon}$, governed by a parameterless distribution, so as $\boldsymbol{\epsilon} \sim P(\boldsymbol{\epsilon})$. Furthermore, the differentiable, deterministic vector-valued function $g_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon})$ to the random noise $\boldsymbol{\epsilon}$ and input $\mathbf{x}$, parametrised by $\boldsymbol{\phi}$, is introduced.

Hence, using the Gaussian distribution specified by eq. (4.15), and fixing $\boldsymbol{\epsilon}$ to be drawn from the normal base distribution, then sampling from the probabilistic encoder $z \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is simulated by evaluating

$$\mathbf{z} = g_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon}) = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\sigma}(\mathbf{x}) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4.18}$$

where $\boldsymbol{\mu}(\mathbf{x})$ and $\boldsymbol{\sigma}(\mathbf{x})$ are the output of the inference network with $\boldsymbol{\phi}$.

It is now possible to compute the the gradient w.r.t. $\boldsymbol{\phi}$, which in turn allows back-propagation. In particular, given objective function $f(\mathbf{z})$, expectations can be rewritten in terms of $\boldsymbol{\epsilon}$, such that $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] = \mathbb{E}_{P(\boldsymbol{\epsilon})}[f(\mathbf{z})]$ [41]. In turn, substituting all occurrences of $\mathbf{z}$ with $g_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon})$ forms a simple Monte Carlo estimator: $\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] = \mathbb{E}_{P(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} f(\mathbf{z})] \simeq \nabla_{\boldsymbol{\phi}} f(\mathbf{z})$. See Fig. 4.4 for an illustration between the naïve and reparametrized implementations.

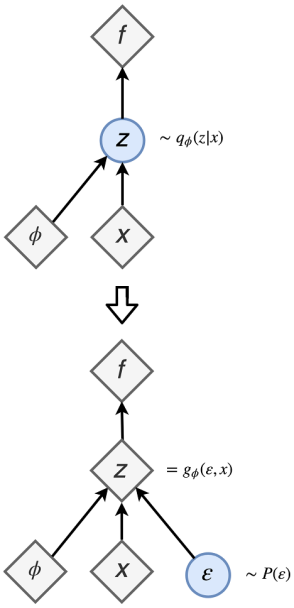


Figure 4.4: Upper: The original form. Lower: The reparameterized form. Gray triangles and blue circles indicates deterministic nodes and random nodes, respectively. In a typically Gaussian setting then $P(\boldsymbol{\epsilon}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Image adapted from: *Notes on the Reparametrization Trick. Errica, F. (2018).*

REPARAMETERIZATION TRICK

4.2.4 Deep Learning Interpretation

In this section an intuition behind the CVAE is given and how it is related to deep learning.

First and foremost, it is important to note that the gradient descent algorithm, by default, seeks to minimize the loss function (as elaborated in section § 2.1.5).

The ELBO serves as the loss function, however the objective was to maximize ELBO by eq. (4.10). Hence, ELBO is modified into a loss function that is to be minimized such that $\mathcal{L}_{\text{VAE}} \equiv -\text{ELBO}$ [41], i.e.:

LOSS FUNCTION FOR VARIATIONAL AUTOENCODER

$$\begin{aligned}\mathcal{L}_{\text{VAE}} &\triangleq \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{KL}} \\ &= -\mathbb{E}_{q_\phi}\big[\log_e p_\theta(\mathbf{x}|\mathbf{z})\big] + D_{\text{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\big).\end{aligned} \tag{4.19}$$

Specifically, given the Gaussian densities from section § 4.2.2, the $\mathcal{L}_{\text{KL}}$ term can be substituted by eq. (4.16) and the $\mathcal{L}_{\text{rec}}$ term by eq. (4.17).

Given a data set of input images $X = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$, the probabilistic encoder specified by eq. (4.15) produces parameters of Gaussian distributions in the the lower dimensional latent space $\mathbf{z}^{(i)}, \mathbf{z} \in \mathbb{R}^J$. It is expected that $q_\phi(\mathbf{z}|\mathbf{x})$ approximately maps the input images to $p_\theta(\mathbf{z})$, such that

$$\sum_{i=1}^{N} q_\phi(\mathbf{z}|\mathbf{x}^{(\mathbf{i})}) = \mathcal{N}\left(\sum_{i=1}^{N} \boldsymbol{\mu}(\mathbf{x}^{(i)}), \sum_{i=1}^{N} \boldsymbol{\sigma}^2(\mathbf{x}^{(i)}) \cdot \mathbf{I}\right) \approx p(\mathbf{z}), \tag{4.20}$$

entailing that the the encoder happens to approximate the prior given by eq. (4.13). The quality of how well the sum of distributions produced by $q_\phi(\mathbf{z}|\mathbf{x})$ approximates the standard normal distribution is exactly what the KL divergence, $D_{\text{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\big)$, in the loss function measures.

So, the data points in the latent space are distributed following an approximate standard normal distribution. Drawing $\mathbf{z}^{(i)} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$, the probabilistic decoder returns a reasonable reconstructed image of the original input image. Specially, assuming that the probabilistic decoder is described by eq. (4.14), $\mathbf{z}$ is transformed into a mean value of the distribution $p_\theta(\mathbf{x}|\mathbf{z})$. I.e. new artificial images are generated according to $\boldsymbol{\mu}(\mathbf{z}) = \tilde{\mathbf{x}}$.

The reparameterization trick is used in order to be able to implement the CVAE. In particular, Fig. 4.5 shows an illustration for the Gaussian cases described by the distributions in § 4.2.2. In the naïve implementation of the CVAE, the input $\mathbf{x}$ is mapped to the intermediate layers taking the values of $\boldsymbol{\mu}_\phi(\mathbf{x})$ and $\boldsymbol{\sigma}^2_\phi(\mathbf{x})$ as described by eq. (4.15). The latent variable $\mathbf{z}$ is then sampled from the probabilistic encoder $z \sim q_\phi(\mathbf{z}|\mathbf{x})$. Finally, through the decoder z is mapped back to the input dimension yielding the reconstruction $\tilde{\mathbf{x}}$. Contrary, using the reparameterization trick, sampling from $z \sim q_\phi(\mathbf{z}|\mathbf{x})$ is simulated by evaluating eq. (4.18). It makes back-propagation feasible in the network such that the optimization problem derived in eq. (4.10) can be solved.

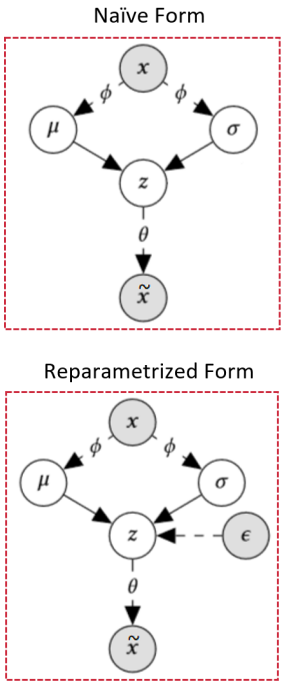


Figure 4.5: Taxonomies of CVAE implementations. The dashed arrows denotes a sampling operation. Image adapted from: *Variational Autoencoders for Collaborative Filtering. Liang, D. et al. (2018).*

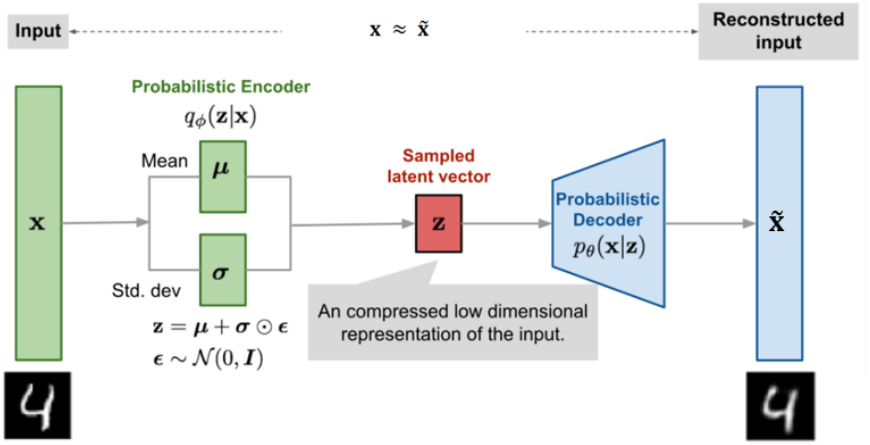


Figure 4.6: Architecture of the CVAE model specified by the the multivariate Gaussian assumptions. For a given input image, the encoder produces a distribution governed by parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Using the reparameterization trick, a point in the latent space is sampled from $z \sim q_\phi(\mathbf{z}|\mathbf{x})$. Hereafter the point is propagated through the decoder, which produces an artificial image. Image adapted from: *Autoencoder to Beta-VAE. Weng, L. (2018).*

Finally, consider Fig. 4.6 and take a look at the full architecture of the CVAE. The encoder takes an input image and compresses it into a smaller form through a CNN (as summarized in § 2.2), and produces parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ that describes a probability distribution in the latent space. Using the reparameterization trick, a latent point sampled from this distribution of $q_\phi(\mathbf{z}|\mathbf{x})$ is then simulated. The decoder decompresses this latent point through an inverse architecture of the CNN used, and transforms it into an approximate reconstruction of the input.

CONTENTS

THE DATA

5

> *Life is like a box of chocolates. You never know what you're gonna get.*
>
> —From the film *Forrest Gump*

This chapter serves as a presentation to the case studies and data used for the implementations in chapter § 6. However, first it is necessary to review and examine advanced computer vision techniques that have been utilized for preprocessing the data, and these technical explorations are found in section § 5.1. Subsequently, the data are formally presented in section § 5.2. This includes how the data was constructed as well as the pipelines for how it have been prepared for the neural networks.

5.1 COMPUTER VISION

This section is dedicated to provide the necessary understandings of various computer vision techniques that all will be proved to be very useful for the preprocessing processes of the datasets introduced in § 5.2. Firstly, in section § 5.1.1 two different techniques of intensity transformations in images are introduced. In sections § 5.1.2 and § 5.1.3 two methods for matching the intensities in images are reviewed. Thereupon an advanced filtering method is examined in § 5.1.4. Section § 5.1.5 is dedicated to asses various methods for image resizing. Section § 5.1.6 serves as a necessary introducing for a method able to correct distorted pixels in an image. Additionally, § 5.1.7 serves as a brief discussion for the importance and need of feature scaling in neural networks. Finally, before images are fed to the neural networks it is good practice to perform various image transformations, a concept introduced in section § 5.1.8.

5.1.1 Intensity Transformations

The *contrast* of a grayscale image indicates how easily objects can be distinguished from other objects and is essentially a matter of how different the gray-level values are [55]. Especially, the contrast of an image can be revealed and quantified by its *image histogram*, for which count of pixels versus pixel intensity values are plotted, cf. eq. (5.3). A low contrast image have pixel values concentrated in a narrow range, and thus the human eye has a hard time distinguishing them. On the other hand, a high contrast image have pixel values far from each others, which makes them easy to distinguish.

Point processing manipulations are applied directly on the pixels of an image, i.e. in the spatial domain. Two fundamental *intensity transformations* operations are descried in the section; the *contrast stretching* and *gamma correction* that have been used for contrast and brightness manipulations. For both datasets these two image enhancement methods have been used in favour of increasing the contrast of the foreign objects as well as obtaining white uniform backgrounds.

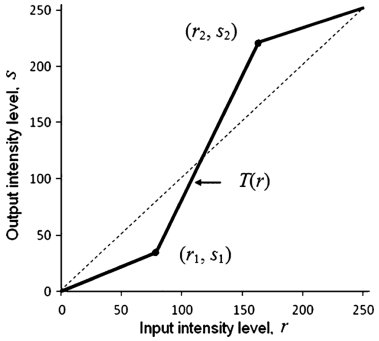


Figure 5.1: Illustration of contrast stretching on an image. Each piecewise linear constrast stretch is computed using the equation of a straight line; $s - s_k = \frac{s_{k+1}-s_k}{r_{k+1}-r_k}(r - r_k)$, for $r_k < r \leq r_{k+1}, k = 0, ..., K-1$. Image source: *Python | Intensity Transformation Operations on Images. GeeksforGeeks (2019).*

Piecewise Linear Contrast Stretching

Contrast stretching is used to enhance a low contrast 8-bit grayscale image. The aim is to improve the contrast by stretching the intensity values to fill the entire dynamic range of the image. Using a *piecewise linear function* it transforms the gray levels in the range $[0, 255]$. It involves the identification of a number of linear enhancement steps, and for each segmented line the equation of a straight line is used to compute the piecewise linear function [52]. Thus, letting T denote a function that maps $r = I(i, j)$ to $s = \tilde{I}(i, j)$, the operation can be expressed by [63]

PIECEWISE LINEAR CONTRAST STRETCHING

$$T(r) = \begin{cases} (s_1/r_1)r, & \text{for } 0 \leq r \leq r_1 \\ \dfrac{s_2 - s_1}{r_2 - r_1}(r - r_1) + s_1, & \text{for } r_1 < r \leq r_2 \\ \dfrac{255 - s_2}{255 - r_2}(r - r_2) + s_2, & \text{for } r_2 < r \leq 255, \end{cases} \tag{5.1}$$

where r and s denotes the gray level of the input image and the output image at any point (i, j), respectively. In general $r_1 \leq r_2$ and $s_1 \leq s_2$ is chosen such that the function used is always linear and monotonically increasing [26].

As illustrated in Fig. 5.1, by changing the location of points (r_1, s_1) and (r_2, s_2) the shape of the transformation function can be controlled. If $s_1 = r_1$ and $s_2 = s_2$ the operation is simply an identity transformation and no change will occur in the image. When $r_1 = r_2$, $s_1 = 0$ and $s_2 = 255$ a *threshold* function is applied, i.e. the images is converted into a binary image with black and

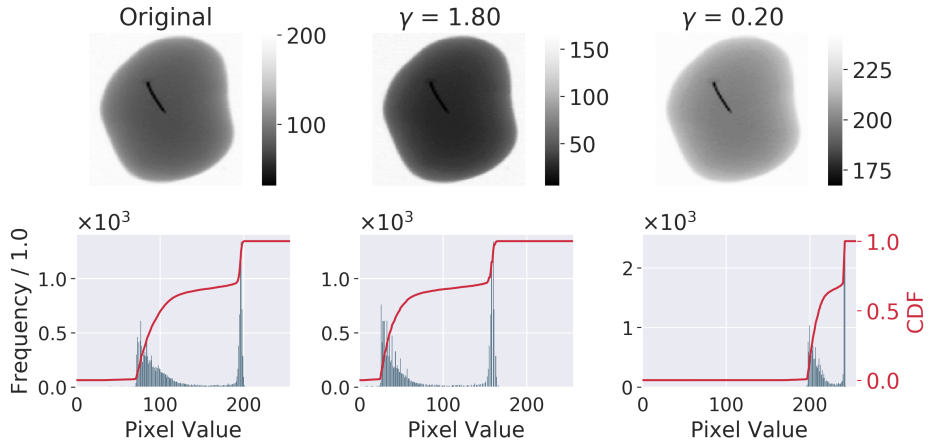


Figure 5.3: To the left the original image is shown. Gamma correction with $\gamma = 1.8$ results in a darker output image while $\gamma = 0.2$ returns a brighter output. $\gamma = 1$ would return an image with no change as it simple corresponds to a linear mapping.

white pixels. Various degree of spread in the intensity level of the output image is produced by choosing intermediate values of (r_1, s_1) and (r_2, s_2).

Gamma Correction

Gamma correction is used for brightness manipulations by use of a non-linear transformation between the input image $I(i, j)$ and the processed image $\tilde{I}(i, j)$. Mathematically, this power-law transformation is expressed as [26]

$$\tilde{I}(i, j) = c\big[I(i, j)\big]^{\gamma}, \qquad c > 0. \tag{5.2}$$

GAMMA CORRECTION

In the common case of $c = 1$ the gamma-mapping is defined such that input and output pixels are in the domain of $[0, 1]$ [55]. Nevertheless, it is easy to convert pixel values between the two ranges of $[0, 1]$ and $[0, 255]$, in particular cf. eq. (5.16) for this rescaling process of pixels values. This transformation can easily be incorporated as an intermediate step in eq. (5.2) if the user want values to be scaled in $[0, 255]$, specifically by $\tilde{I}(i, j) = 255\big[I(i, j)/255\big]^{\gamma}$.

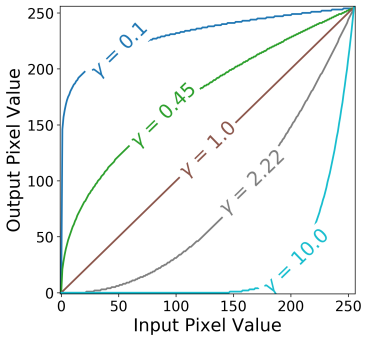


Figure 5.2: Gamma-mapping curves for different values of gamma. Pixel values in domain $[0, 255]$.

As the gamma transformation is non-linear, the effect depends on the original pixel values and will not be the same for all pixels. In Fig. 5.2 some different gamma-mapping curves are illustrated. Given $\gamma = 1$ the identity-mapping is rendered. For $0 < \gamma < 1$ the intensity increases, i.e. the dark regions of the input image becomes brighter. On the other hand, if $\gamma > 1$ the processed image becomes darker as the intensity of pixels decreases.

To better illustrate the effect of the gamma mapping, Fig. 5.3 shows some concrete image examples as well as their respective image histogram, for where γ has been varied towards both small and large values. Given $\gamma > 1$ the histogram shifts towards the left, resulting in the bright regions becomes darker, thus the resulting image will be darker compared to the input image.

Vice versa, for $\gamma < 1$ the output image will be brighter than the input image as the histogram shifts towards the right.

5.1.2 Histogram Matching

For simplicity, consider again a grayscale image $I(i, j) \in \mathbb{R}^{H \times W}$. The histogram of such an image is a *discrete frequency distribution* as the intensity value in each bin describes the observed counts of the corresponding value. Formally, given a grayscale value s, the sum of all histogram entries is [10]

IMAGE HISTOGRAM

$$\sum_s h(s) = H \cdot W, \tag{5.3}$$

since the total number of pixels in an image is $H \cdot W$. The probability of occurrence of each pixel value can easily be computed by

NORMALIZED HISTOGRAM

$$p(s) = \frac{h(s)}{H \cdot W}, \qquad \text{where } \sum_{s=0}^{k} p(s) = 1 \text{ as } p(s) \geq 0, \tag{5.4}$$

and is known as the *normalized histogram*, or *probability density function (PDF)*. The associated discrete *cumulative distribution function (CDF)* is then the sum of all frequencies of intensity values lying in its domain, defined by

CUMULATIVE DISTRIBUTION FUNCTION

$$C(k) = \frac{1}{H \cdot W} \sum_{s=0}^{k} h(s) = \sum_{s=0}^{k} p(s), \qquad k = 0, ..., L-1, \tag{5.5}$$

where L is the number of possible intensity levels in an image [10, 26]. It follows that the CDF is monotonic increasing upwards and $C(k) \in [0, 1]$.

Sometimes circumstances occurs where images have been taken under e.g. different conditions or from different sources. Given such cases it can be useful to be able to specify the shape of the histogram that the processed image should have. Such an image processing method able to manipulate the pixels of a source image in such a way that its histogram matches that of an target image does indeed exist, and is known as *histogram matching* (or *histogram specification*).

Given two grayscale images, a source image $I_s(i, j) \in \mathbb{R}^{H_s \times W_s}$ and target image $I_t(i, j) \in \mathbb{R}^{H_t \times W_t}$, each of their respective histograms are computed by eq. (5.3). Following, given the number of bins, L, their CDFs are obtained by eq. (5.5), that is

$$C_s(k) = \frac{1}{H_s \cdot W_s} \sum_{s=0}^{k} h_s(s) \qquad \text{and} \qquad C_t(k) = \frac{1}{H_t \cdot W_t} \sum_{s=0}^{k} h_t(s), \tag{5.6}$$

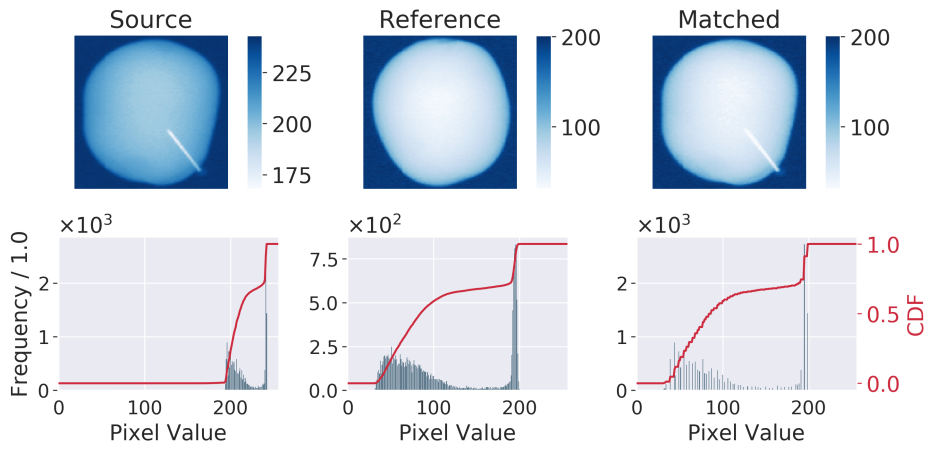


Figure 5.4: The histogram of the source image is modified based on the distribution of the target/reference image. Notice how the intensities of the source image have been mapped towards the target image, resulting in an optimal matched image.

where $k = 0, ..., L-1$, for which $L = 256$ is the upper limit for an 8-bit grayscale image. Subsequently, given the information above, the matched image can then be described by the mapping [10, 26]

HISTOGRAM MATCHING

$$\tilde{I}(i,j) = C_{\mathrm{t}}^{(-1)}\Big(C_{\mathrm{s}}\big(I_{\mathrm{s}}(i,j)\big)\Big). \tag{5.7}$$

Hence, histogram matching performs a mapping that optimally transform intensities of the source image towards the target image, such that the matched image $\tilde{I}(i,j)$ with distribution C'_{s} matches the reference distribution C_{t} as closely as possible, that is $C'_{\mathrm{s}}(k) \approx C_{\mathrm{t}}(k)$. If the images have multiple color channels, the histogram matching is performed independently for each channel.

In general the CDF is however not invertible since constant plateaus are encountered at ranges where intensities are not occurring in the image, hence it might not be a *strictly* monotonically increasing function. In order to overcome the problem of non-invertibility for histogram matching the pseudo-inverse is considered instead. That is, for any CDF defined on the integer set $s \in \{0, ..., 255\}$, its pseudo-inverse is defined as following [10]

$$C^{(-1)}(l) = \min\{s \mid C(s) \geq l\}, \qquad \text{where } l \in [0,1]. \tag{5.8}$$

To illustrate the effect of the histogram matching, the image histogram and CDF for the gray channel is plotted on Fig. 5.4. Clearly, the matched output image has the same CDF as the reference image.

5.1.3 General Midway Equalization

Instead of matching the histogram of a source image with the histogram of a target image as described in § 5.1.2, a pair of images can instead be matched toward a 'midway' histogram, yielding them the same distribution.

If given two grayscale images, $I_1(i,j)$ and $I_2(i,j)$, the change given by [17]

$$\varphi(x) = \frac{1}{2}\Big(C_1^{(-1)}(x) + C_2^{(-1)}(x)\Big), \tag{5.9}$$

where $\tilde{I}_1(i,j) = \varphi(C_1 \circ I_1) = \varphi\big(C_1(I_1)\big)$ and $\tilde{I}_2(i,j) = \varphi(C_2 \circ I_2) = \varphi\big(C_2(I_2)\big)$, are called the *midway equalizations* (or *midway specifications*) of I_1 and I_2. In order words, by writing the equations explicitly the corrected images are easily constructed by

$$\begin{aligned}
\tilde{I}_1(i,j) &= \frac{1}{2}\Big(C_1^{(-1)}\big(C_1(I_1)\big) + C_2^{(-1)}\big(C_1(I_1)\big)\Big) = \frac{1}{2}\Big(I_1 + C_2^{(-1)}\big(C_1(I_1)\big)\Big) \\
\tilde{I}_2(i,j) &= \frac{1}{2}\Big(C_1^{(-1)}\big(C_2(I_2)\big) + C_2^{(-1)}\big(C_2(I_2)\big)\Big) = \frac{1}{2}\Big(C_1^{(-1)}\big(C_2(I_2)\big) + I_2\Big),
\end{aligned}$$

as $id = C_k^{(-1)}(C_k \circ I_k), k \in \{1, .., , K\}$ are identified as the identity functions [17]. This way, $\tilde{I}_1$ and $\tilde{I}_2$ will end up having the same cumulative histogram φ found by the harmonic mean between the input images cumulative histograms C_1 and C_2. Do notice that the pseudo-inverse $C^{(-1)}$ is computed by eq. (5.8).

The midway equalization method as described by eq. (5.9) can however be generalized to N arbitrary numbers of images. Specifically, the *general midway equalization* [30]

GENERAL MIDWAY EQUALIZATION

$$\varphi(x) = \frac{1}{N}\sum_{p=1}^{N} C_p^{(-1)}(x), \tag{5.10}$$

for $p \in \{1, ..., N\}$ and where $\tilde{I}_n = \varphi\big(C_n(I_n)\big)$ with $n \in \{1, ..., N\}$. Given RGB images, the midway equalization is applied channel-wise.

To illustrate the effect of the general midway equalization three grayscale images of the same size are considered on Fig. 5.5. Notice how the contrast of all input images have been equalized with each others, hence the output images yields the same intermediary grey level distribution.

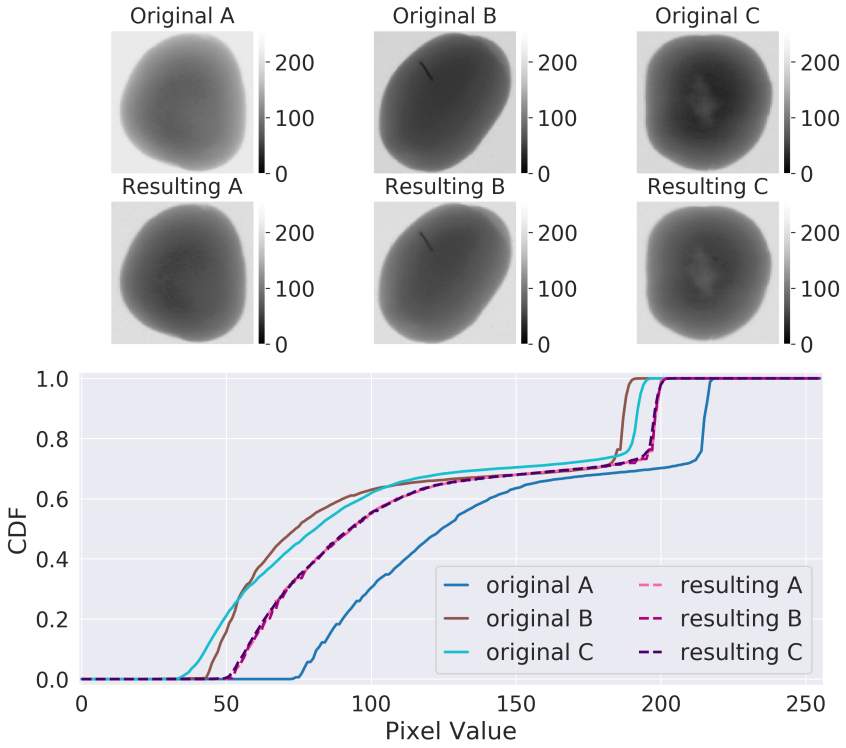


Figure 5.5: General midway equalization for three grayscale images with same size. The CDFs of the input images and output image are represented by continuous and dashed lines, respectively.

5.1.4 *Filtering*

The images are infected by noise, i.e. random variations of brightness. In order to modify such noisy variations, images can be smoothed by *filtering*. Filtering methods are so-called *neighborhood processings* as the outputs is determined by not only the pixel value at the same position as the input, but of the values of the pixel neighborhood at the same time [55].

Neighborhood processing of an 2D image, $I \in \mathbb{R}^{H\times W\times C}$, are performed though convolution (or correlation) with a kernel $K \in \mathbb{R}^{k_1\times k_2\times C}$. The process is described in details in section § 2.2.2, and mathematically expressed by

2D CONVOLUTION

$$\tilde{I}(i,j) = (K * I)(i,j) = \sum_{n=0}^{k_1-1}\sum_{m=0}^{k_2-1} K(m,n)I(i-m, j-n), \tag{5.11}$$

As will be seen in section § 5.2 below, the datasets contains important edge informations, and therefore convolution with filters such as *mean* or *median* kernels, that typically blurs out a significant amount of details, have not been utilized. Instead, in order to preserve edges when filtering noise, the *bilateral filter* has been applied successfully. Given pixel positions $\mathbf{p}$ and $\mathbf{q}$, the resulting intensity at center position $\mathbf{p}$ is defined by [44]

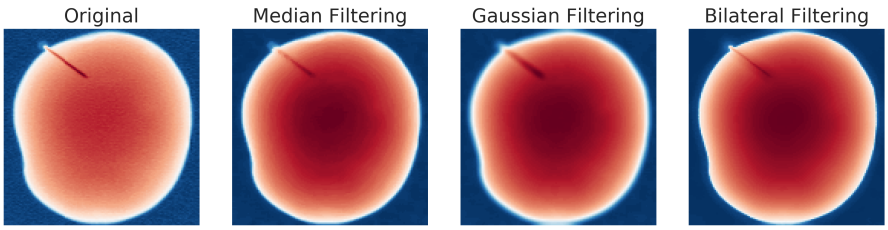


Figure 5.6: Median, Gaussian and bilateral denoising filters applied onto the same noisy input image. Notice their abilities in preserving edges and texture while removing noise.

BILATERAL FILTER

$$\mathrm{BF}[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|_2) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}, \tag{5.12}$$

where the normalization factor $W_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|_2) G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)$ ensures pixel weights sums to 1.0. Here, $\mathrm{s} \in \mathcal{S}$ denotes the set of all possible pixel locations in the *spatial domain* and the *range domain* $\mathrm{r} \in \mathcal{R}$ corresponds to all possible intensity values. Hence, this non-linear filter takes a 2D Gaussian kernel in both space and range following

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right). \tag{5.13}$$

The Gaussian function in domain performs a Gaussian blurring of nearby pixels, and as the spatial parameter σ_s increases, the larger the neighborhood. This Gaussian blurring is simple a weighted average of intensities, thus leading to potential edge blurring. As pixel values at edges corresponds to a significant change in gray-level values, the Gaussian function of intensity difference ensures that only pixels with similar intensities to the central pixel are considered for the blurring process. Hence, this component penalize pixels with a different intensity, and the smaller the range parameter σ_r, the shaper the edge. As $\sigma_r \to \infty$, eq. (5.12) tends to a Gaussian blurring since the range of eq. (5.13) widens and flattens.

To illustrate the convolution with different kernels consider Fig. 5.6, where image blurring utilizing a Gaussian, median and bilateral filter have been considered. In general, the bigger the kernel, the more smoothing are provided. The Gaussian filter results in both noise and edges being blurred. The median filter is effective at removing the noise, however here there is also a loss of texture. On the other hand, the bilateral filter is better at preserving edges while also removing noise.

5.1.5 Downsampling

Downsampling is the process of reducing the dimensions and spatial resolution of an image while keeping the 2D representation, and this process is

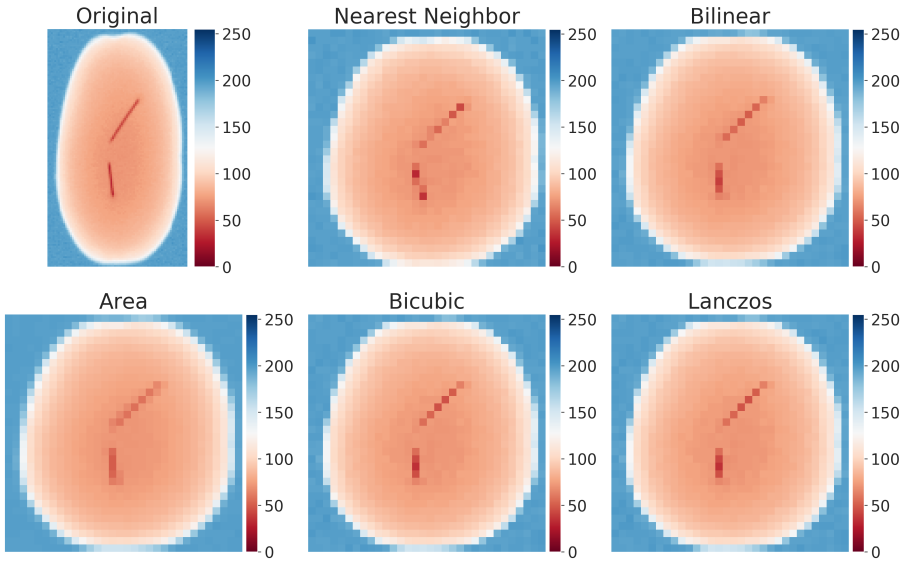


Figure 5.7: Original image downsampled from $(136, 80) \rightarrow (32, 32)$ pixels using various interpolation methods.

typically achieved though *image interpolation* techniques. Particularly, it is common practice to standardize the image size given to a neural network, although they do no require any specific pixel dimensions. Furthermore, downscaling of an image makes the data of a more manageable size, hence reducing the storage size and enables faster processing of the data.

Consider Fig. 5.7 where five different and commonly used downscaling methods have been examined: nearest neighbor, bicubic, bilinear, area, and lanczos interpolation. As noticed, each method has its own advantages and disadvantages.

The nearest neighbor interpolation preserve the edges in the image, but it does not work well for the finer details. That is, the downsampled image looks edgy and sharp as the method does not blend the information together smoothly. Hence, it result in the foreign object being less visible compared to the other methods, and so this technique is not ideal for the data.

Both the bilinear and bicubic interpolations tends to blur the image. Although it is not very noticeable on the figure, the bicubic method usually performs better than the bilinear method [24]. On the other hand, the Lanczos technique can lead to *ringing effects*. Ringing effects in image processing are the appearance of rippling artifacts in the vicinity of image edges, producing bands of "echos" that potentially can make it hard to approximate sharp edges — effects one would like to avoid.

The area method (or box filter) results in an over-smoothing, but the foreign objects are still visible. Nevertheless, this over-smoothing of the food product is actually desirable, since it is expected this will help the CAE and CVAE models better reconstruct the original samples as the pixel values does not differs as much.

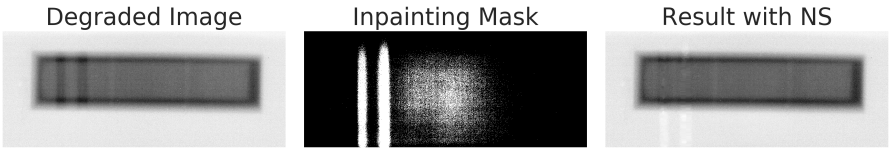


Figure 5.8: Left: The degraded input image. Middle: The binary mask used for the image inpainting. Right: The resulting image using the Navier-Stokes method.

5.1.6 Image Inpainting

Unfortunately, due to the scanning set-up of one particular dataset introduced in section § 5.2.1 — the chocolate bar dataset — it is necessary to consider *image inpainting*, which is another image interpolation technique.

Image inpainting is the process of restoring damaged or missing parts of an image. Using a *binary mask* that specifies the location of the damaged pixels in the input image, these pixels are automatically reconstructed by some chosen algorithm. In particular, presented information from non-damaged regions, the reconstruction is performed automatic with the *Navier-Stokes* inpainting technique [6].

As the name might be spoiling, the mathematical framework of this algorithm is based on *fluid dynamic* and utilize partial differential equations to travel along boundaries until a steady state is reached, subsequently filling the inpaintig region with the correct color gradient. The mathematical description behind the method is non-trivial plus rather tedious, and is beyond the scope of this thesis. However, the curious reader may find detailed walk-through in [2, 6]. Rather, it is primary the practical application behind image inpaining that is examined in this section.

Suppose the binary mask

$$M(i,j) = \begin{cases} 1, & \text{if } (i,j) \textit{ is} \text{ damaged in } I(i,j) \\ 0, & \text{if } (i,j) \textit{ is not} \text{ damaged in } I(i,j) \end{cases} \tag{5.14}$$

specifies the location of the damaged pixels (i, j) in the input image $I(i, j)$ that should be corrected. I.e. given the mask, non-zero pixels corresponds to the areas that are to be modified, and regions of zero pixels should be kept as it is.

Hence, a mask of the same size as the input image should be created. Fig. 5.8 shows first an degraded input image and secondly the constructed mask. Notice the two remarkable thick vertical stripes going though the chocolate bar; these are mainly the ones we wish to correct. The mask have automatic been constructed by subtracting two calibration images from each other — subtracting of an "exposure OFF calibration image" with an "exposure ON calibration image" — and subsequently applying *global thresholding* in order to produce a binary mask. The final image is the result of applying the Navier-Stokes algorithm. This resulting image is not perfectly restored, as the vertical lines are very thick and thus hard to correct, however the inpainting regions have reasonable been filled in by the surrounding regions.

5.1.7 Feature Scaling

The objective of *feature scaling* is to change numerical values to a common scale without distorting differences in the ranges of values, helping to ensure all feature contributed equally and no specific feature dominates the other. Additionally in deep learning, data scaling typically improves the performance of the model and accelerate training [81]. The specific choice of which scaling methods that are to be performed typically depends on the specific of the problem as well as on the statistics of the whole dataset. Especially, it is important to note that the scale of the output variable of the network matches that of the activation function on the output layer.

The *min-max normalization* is one of the most common ways to normalize data. The mathematical formulation for the min-max scaling is given by

MIN-MAX NORMALIZATION

$$\mathbf{x}' = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}. \tag{5.15}$$

That is, given an 8-bit grayscale image, the minimum pixel value gets transformed into a 0 and the maximum pixel value gets transformed into a 255. Every other pixel value gets transformed into a decimal between 0 and 255. Hence, for images this is a form of histogram stretching (or contrast stretching) as the process changes the range of pixel intensity values to span the full domain.

The raw grayscale images with pixel integers in the range between 0 and 255 can be presented directly to neural networks however, as discussed in § 2.1.2, inputs with large integer values slows down training and can disrupt the learning process. Instead, prior to modeling the pixel values are rescaled into the domain $[0, 1]$ by applying the following *normalization* procedure

NORMALIZATION

$$\mathbf{x}' = \frac{\mathbf{x}}{255.0}. \tag{5.16}$$

This is beneficial for the model as it likely will perform better and speed up training. As the sigmoid activation given by eq. (2.4) renders outputs between 0 and 1, it is consistently used as the output function in the implementations presented in section § 6. Especially for the CAE and CVAE, that compares input image with output image, pixel values needs to be enforced into the same hard range. Henceforth, the feature scaling given by eqs. (5.15)-(5.16) are the only one utilized.

5.1.8 Image Augmentation

Image *augmentation* can be used to enlarge a dataset, which is especially helpful when the number of samples is sparse or if the scan image were taken

under a limited set of conditions. By generating synthetically modified data to the training set, it will help the neural network model to generalize better as both the count of images and their associated variations are increased [27]. A partial goal with augmentation is to make the neural network robust, i.e. invariant to relevant variances such as translation, size, viewpoint or illumination.

Augmentation is pretty straightforward to perform on images as they includes a range of different factors which easily can be simulated, such as distinct locations, orientations or scales. Depending on what kind of images the dataset contains and on the process of how scan images are being taken, only relevant augmentations should be performed. The user usually have knowledge about what kind of variances the network should be invariant to — and this is also the case for the datasets introduced in section § 5.2 below.

When feeding data to the neural network, the transformations can be performed in two ways; either by *offline augmentation* or *online augmentation* (*on-the-fly augmentation*). Offline augmentation is performed before data is fed to the model, and this option is usually preferred for relatively smaller datasets. On the other hand, online augmentation is utilized while the network is being trained, specifically the transformations are performed on the mini-batched fed to the model. This method is typically preferable for larger datasets.

5.2 CREATING THE DATASETS

In this thesis two different device under tests (DUTs) case studies are examined. They cannot be compared easily and so the considerations made about the datasets differs at some points. The data are formally presented in section § 5.2.1 whereas the applied foreign objects and otherwise defects are outlined in section § 5.2.2.

It is worthwhile to mention that all X-ray images are read and processed as 8-bit grayscale images, which scale is shown in Fig. 2.12. Accordingly, in order for a good model to be trained, image preprocessing is needed before applying the data to the algorithms established in section § 4. Thereby, § 5.2.3 presents the individual preprocessing pipelines applied to each individual dataset.

Subsequently, in § 5.2.4 the training, validation and testing sets and their assumptions are formally presented. As the number of images taken by the detectors are limited, image augmentation has been used in order to increase the total image count, hence the applied image augmentations for each dataset are also outlined here.

Table 5.1: Foreign objects imposed on anomalous samples. All sizes are measured using an electronic caliper, and as the specific foreign objects have their own natural variances, the reported measures are approximate. If the length, l, is known for non-spherical objects the measurement is reported. The estimated chemical compositions are specified if known.

Type	Chemical Composition	Morphology	Diameter [mm]	
			Size 1	Size 2
Lead ball	Lead (94%)	Spherical	2	—
Steel ball	Iron (95%)	Spherical	1.5	4
Steel wire	Unknown	Rectangle	0.4	0.8
Thumbtack	Aluminium (99%)	Rectangle (l = 5 mm)	1	—
Staple	Iron (62%), zinc (37%)	Rectangle (l = 7 mm)	0.3	—
Needle	Iron	Rectangle (l = 20 mm)	1	—

5.2.1 Data Presentation: Scan Settings

Two type of food products have been investigated; a particular type of chocolate bar and potatoes. Information about the different classes and how they have been constructed are found in § 5.2.2 below.

The first dataset consists of X-ray images of motionless chocolate bars inside their food wrappers. The images have been recorded by a 2D scintillator detector [1] with tunings 60 kV (tube voltage) and 7.5 mA (tube current).

1. Once again, I owe my gratitude to *Newtec Engineering A/S* for forwarding me images of the chocolate bars recorded on their 2D scintillator detector.

The second dataset is comprised of X-ray images of mobile potatoes. These images have been taken by the ButeoX scanner introduced in section § 1.3.4. This data were recorded at 80 kV voltage and 100 W power, i.e. corresponding at an 1, 25 mA tube current. Moreover, the line rate was set to 1000 Hz.

5.2.2 Foreign Objects

In this section the foreign objects applied to the X-ray images and other types of defects are outlined. Normal samples are typically alike and relatively easy to collect and define. In practice, defective samples are scarce and underrepresented, thus typically not easily obtained nor collected. Instead, all data labelled as abnormal have been created artificially. Table 5.1 shows an overview of the foreign objects and their measured sizes and compositions used in creating the abnormal data.

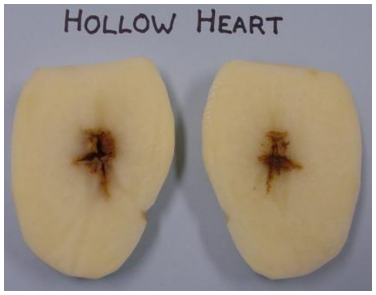


Figure 5.9: RGB image of potato with hollow heart disease. Image source: *Hollow heart in potatoes. Agriculture and Food, Australia (2016).*

On the one hand, the *hollow heart disease* that might emerge internal in potatoes are an international problem. One cannot distinguish perfect potatoes from potatoes with hollow heart until they are cut open, whereas it then will be obvious. As shown in Fig. 5.9 hollow heart in potatoes manifests irregularly-shaped and rather in a 'star'-shaped cavity form. The hollow hearts varies in shape and size, nevertheless X-ray imaging are able to detect this disease.

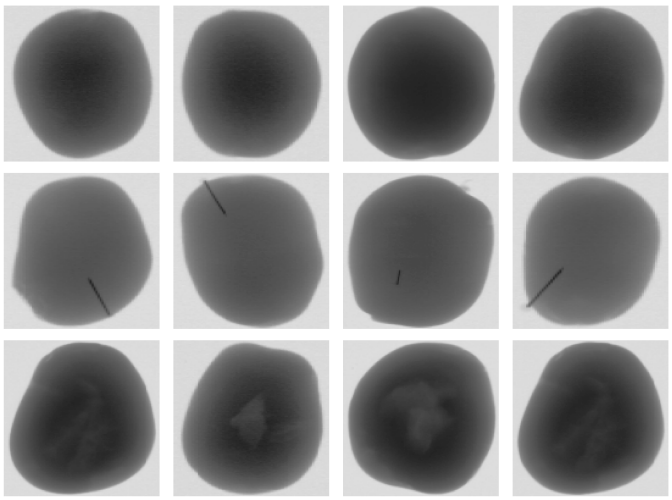


Figure 5.10: Small selections of raw X-ray images of potatoes and after downsampling to dimensions (128, 128). Upper row: Perfect samples. Middle row: Potatoes with inserted needles. Lower row: Potatoes with artificially hollow heart disease.

On the other hand, in Australia needles have been found in strawberries and later on in potatoes as well[2]. Thus, it motivates to consider a second anomaly class.

2. The curious reader may find more information in the article: *Strawberry needle scare: Growers to look to metal detectors to contain contamination crisis. In: Australian Broadcasting Corporation. By: Jo Prendergast and David Weber (2018).*

The potato dataset consists of 50 'perfect' potatoes. The foreign object categorized as 'needle' in Table 5.1 were randomly inserted in 49 of these same potatoes. Subsequently, the needles were removed and hollow hearts were artificially created in 39 of these potatoes. Each of these classes were recorded four times in the ButeoX scanner, yielding a dataset consisting of 540 images. On Fig. 5.10 a small X-ray selection of perfect potatoes, potatoes with added metal and potatoes with artificial made hollow hearts are shown.

Figure 5.11: Selected RGB images of chocolate bars used for the case study. Left: normal sample. Right: added foreign objects and a vertical crack.

The chocolate bar dataset consists of 22 recorded images in two classes: 10 normal samples and 12 abnormal samples. 5 out of 10 bars were used to construct the abnormal samples; first by randomly placing the remaining foreign objects in Table 5.1 onto the bars, and afterwards by tearing them apart vertically — with or without added foreign objects, cf. Fig. 5.11. A selection of X-ray images of the chocolate bars before preprocessing can be seen in Fig. 5.12. Contrary to the potato dataset, which already poses real-life problems, the chocolate bar dataset is made as an experimental set-up in order to investigate both another shape and possible defects of a food product.

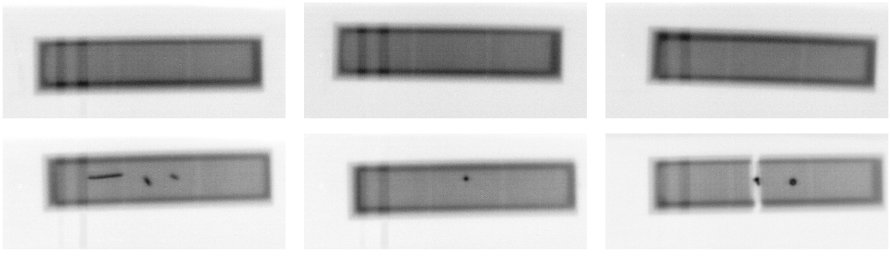


Figure 5.12: Selection of X-ray samples of chocolate bars before preprocessing. Upper row: normal samples. Lower row: anomalous samples with randomly added foreign objects and possible vertical cracks.

5.2.3 *Image Preprocessing: Pipelines*

The raw X-ray images of the two datasets cannot be easily compared, and so the particular preprocessing methods and their applied sequences are slightly different. However, the intentions and motivations behind the preprocessing are the same. Notice, a particular pipeline has been applied on both normal and anomalous data.

Commonly, as elaborated in § 4.1-§ 4.2, the generative models seeks to reconstruct images as close to the originals as possible, hence it serves as motivation enough for 'removing' the background around the object at interest. The grayscale background of the raw images is a mix of values within the range of 0 to 255. But by removing the background all these values can instead be replaced by a single intensity value, such as 255. Feeding the generative models with images of removed backgrounds should thus serve as being better at reconstructing the original images. As will be demonstrated below, the piecewise linear contrast stretching offers an elegant approach for replacing backgrounds containing various intensities with a singular white background.

Primarily, the focus lies on increasing the contrast in the images. In particular, the data are undergoing selected intensity transformations in order to increase the contrast between product and foreign objects or other defects.

Moreover, in preparation of feeding images to the neural networks, all image samples of both datasets have been rescaled to a lower dimension, specially $I \in \mathbb{R}^{\text{original} \times \text{original}} \rightarrow I \in \mathbb{R}^{128 \times 128}$. As the computational memory is limited, this lower dimensionality have first and foremost been chosen in order to avoid the memory limit and for faster computations. Secondly, this square dimensionality have convincingly struck a reasonable balance between resolution and dimensionality reduction, and is furthermore an appropriate input dimension for both the CAE and CVAE models.

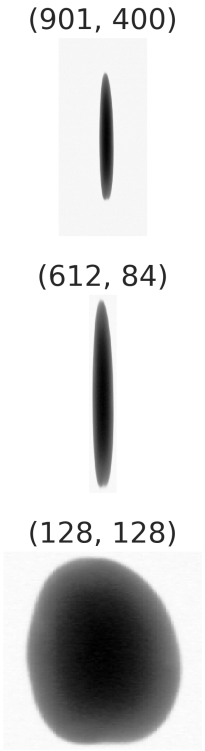


Figure 5.13: The downsampling process for potatoes. Upper: the raw scan of a perfect potato. Middle: The background is cropped away such that only the potato itself is left in the frame. Lower: Utilzing the area interpolation, the image is downsampled to its desired dimensions.

Dataset: Potatoes

For the potato dataset, the following preprocessing pipeline and parameters have been applied:

i Due to the line rate and the speed of the conveyor belt not being synchronized, the recorded potato images have been stretched along the direction of travel. Hence, firstly the backgrounds have been cropped away, whereas all images have been downsampled to the desired dimensions of $(128, 128)$ by use of the area interpolation, cf. § 5.1.5. On Fig. 5.13 an example is shown.

ii Consider the image histogram of a normal potato on Fig. 5.14. This histogram is representative for the whole potato dataset, and it can be

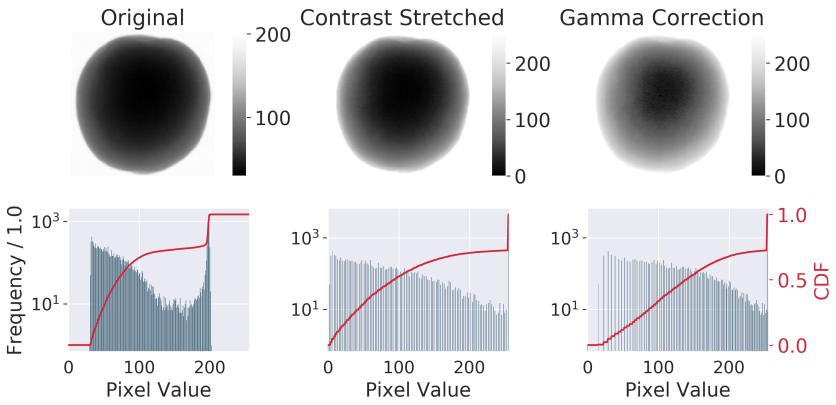


Figure 5.14: Left: Raw X-ray image example of perfect potato after being downscaled to $(128, 128)$. Middle: Result of utilizing piecewise linear contrast stretching given $(r_1, s_1) = (30, 0)$ and $(r_2, s_2) = (140, 255)$. Right: Result of applied gamma correction given $\gamma = 0.6$. Notice the log scale on the y-axis of the image histograms.

seen it has low contrast that does not spend the full range of $[0, 255]$. By scanning the histogram of its pixel groupings, the operation given by eq. (5.1) is applied onto the full dataset with settings $(r_1, s_1) = (30, 0)$ and $(r_2, s_2) = (140, 255)$. Hence, pixels values less than 30 becomes 0 and pixels above 140 becomes 255, i.e. this way the non-uniform gray background is automatic removed and replaced by an uniform white background. Furthermore, the contrast in the image is increased such that bright pixels becomes brighter and dark pixels becomes darker, and the image gets normalized in the process as well since it spans the full range of $[0, 255]$.

iii Subsequently, in order to increase the contrast between foreign objects and hollow hearts, gamma correction given $\gamma = 0.6$ has been applied as well, see Fig. 5.14.

iv To reduce noise the bilateral filter given parameters $\sigma_s = 15$ and $\sigma_r = 20$ and neighbour diameter have hereafter been applied, cf. § 5.1.4.

v Lastly, in order to prepare the data for the neural networks, the normalization described by eq. (5.16) have been utilized.

Dataset: Chocolate Bars

The preprocessing steps for the chocolate bar dataset is very much in accordance with the potato dataset above. However, there are a few changes and the applied parameters differs slightly. Nonetheless, the following preprocessing pipeline and parameters for the full chocolate bar dataset have been utilized:

i First and foremost, inpainint utilizing Navier-Stokes method have been applied, see § 5.1.6. The approach for finding the most appealing solution is heuristic, and an inpainting radius, i.e. the neighborhood around a

pixel to inpaint, of 20 has been implemented, due to the regions that are to be inpainitng are pretty thick.

ii The images are noisy, and in order to denoise the images while preserving edges the bilateral filter given neighborhood diameter $d = 10$, and parameters $\sigma_s = 20$ and $\sigma_r = 25$ have successfully been applied, cf. § 5.1.4.

iii As for the potato dataset, the piecewise linear contrast stretching have been utilized in order to obtain a white background consisting of intensity values 255, cf. § 5.1.1. The image histograms of normal chocolate bars have been scanned, for which parameters $(r_1, s_1) = (170, 0)$ and $(r_2, s_2) = (240, 255)$ heuristically have been chosen.

iv Next, gamma correction given $\gamma = 0.8$ have been applied, cf. § 5.1.1.

v Downsampling using the area interpolation have been applied, cf. § 5.1.5. All images have been resized to the desired dimensions of $(128, 128)$.

vi Furthermore, in preparation for the neural networks, the images are scaled using the min-max normalization and the normalization given by eq. (5.15) and eq. (5.16), respectively.

5.2.4 *Splitting the Data*

An X-ray image of a regular intact chocolate bar is defined as normal, $\mathcal{N}^{\text{choc}}$, while an X-ray image of a sample that might have been slightly broken and where foreign objects have been placed randomly is refereed to as an anomaly, $\mathcal{A}^{\text{choc}}$. The training data singularly consists of normal data, while the test data consists of both normal and anomalies, i.e. $\mathcal{D}^{\text{choc}}_{\text{train}} = \{\mathcal{N}^{\text{choc}}_{\text{train}}\}$ and $\mathcal{D}^{\text{choc}}_{\text{test}} = \{\mathcal{N}^{\text{choc}}_{\text{test}}, \mathcal{A}^{\text{choc}}\}$. Accordingly, the validation set consists exclusively of normal samples, i.e. $\mathcal{D}^{\text{choc}}_{\text{val}} = \{\mathcal{N}^{\text{choc}}_{\text{val}}\}$.

Consistent with the chocolate set, the potato images are also distributed in normal and anomaly classes such that $\mathcal{D}^{\text{potato}}_{\text{train}} = \{\mathcal{N}^{\text{potato}}_{\text{train}}\}$, $\mathcal{D}^{\text{potato}}_{\text{val}} = \{\mathcal{N}^{\text{potato}}_{\text{val}}\}$, and $\mathcal{D}^{\text{potato}}_{\text{test}} = \{\mathcal{N}^{\text{potato}}_{\text{test}}, \mathcal{A}^{\text{potato}}\}$. There are two type of anomaly instances for this potato set; (1) potatoes with needles, and (2) hollow potatoes, hence $\mathcal{A}^{\text{potato}} = \{\mathcal{A}^{\text{potato}}_{\text{metal}}, \mathcal{A}^{\text{potato}}_{\text{hollow}}\}$. As far as the X-ray imaging have been able to capture, the normal class consists entirely of 'perfect' potatoes ready to be distributed to customers.

Accordingly, the flowchart of splitting the datasets are shown in Fig. 5.15. All original data have been undergoing the exact same data preprocessing, whose individual pipelines are reported in § 5.2.3. Hereafter normal and abnormal samples have been subdivided into each of their individual classes. Subsequently, the normal data were randomly split into a large training set and smaller validation and test sets. Henceforth, all anomalous data have been combined with the normal test data in order to construct a full test set.

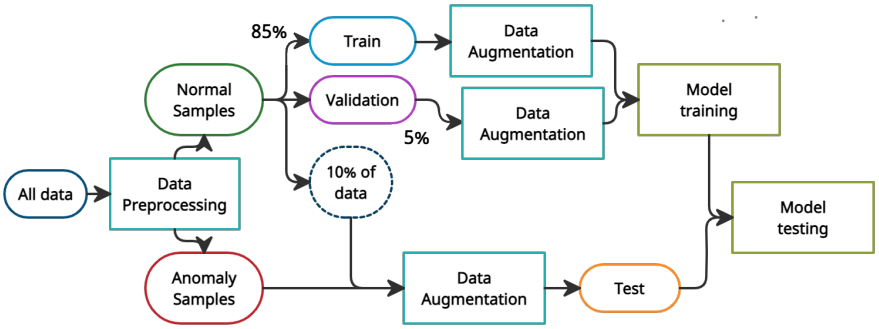


Figure 5.15: Flowchart of the main steps of splitting the data using data augmentation. Percentages are approximate and may be changed appropriately. Flowchart created with `Creately`.

Usually data augmentation is a step solely utilized onto the training data, and only after data is split into their subsets. Augmentation is performed on the training set in order to help the neural network generalize better. As the trained neural network is intended to make predictions for unseen data, the same operations on the validation and test data are normally performed. Due to the sparse number of accessible and original data, it have been necessary to also perform augmentation on the test and validation sets, and if more real data was available this would not be a requirement. A bias might be introduced by performing the augmentations, but leaving out variations in the validation set might as well present a bias if such left-out variations will occur in the inference phase.

The datasets introduced in sections § 5.2.1-§ 5.2.2 are sparse — the potato dataset only consists of a little more than five hundred images in total, and the chocolate only around twenty or so in total. Since so few original images obviously are not enough for training an outstanding neural network, the number of samples is increased by applying offline image augmentations in order increase the size of the datasets. To make sure the implementations in section § 6 can be compared appropriated, the same transformations for the individual datasets have been performed. The ultimate goal is however to have the augmented samples reflect variations of how "real" data looks like as close as possible.

The chocolate bars barely varies in shape and size — they are after-all produced to be homogeneous. If one hypothesize an assembly line used for sorting of chocolate bars, it may just likely be that the products are rotated randomly, shifted slightly in each direction or flipped upside down, as an X-ray image is taken at a point in time using a constant tuning setting. Henceforth, this motivates the following chosen transformations to be used in the augmentations of the chocolate set: in-plane $\theta \in (0, 360)^\circ$ rotations, translation in horizontal and vertical axis, as well as a small scale transformation for compensating natural variances in the samples.

As previously stated the potato dataset have been recorded by a line detector. But as the images are resized to a square dimension, it is obvious to notice that potatoes indeed do vary in shape and size. Thus, the potatoes are also

Table 5.2: Total counts of the data having been randomly split into training, validation and testing sets. The data is utilized in section § 6. Here the default anomaly rate has been reported, which is simply approximately 50/50 between normal and abnormal samples.

Dataset	$\lvert\mathcal{D}_{\text{train}}\rvert$	$\lvert\mathcal{D}_{\text{val}}\rvert$	$\lvert\mathcal{D}_{\text{test}}\rvert$	
			$\lvert\mathcal{N}_{\text{test}}\rvert^*$	$\lvert\mathcal{A}\rvert^*$
Chocolate Bar	1400	100	340	340
Potato	1639	100	350	350

allowed for in-plane $\theta \in (0, 360)^\circ$ rotations, small scale transformations, and very small translations in the horizontal and vertical axis.

As mentioned in section § 3.1, anomalous instances are very scarce and only occurs a small fraction of the time. For practical evaluation of the demonstrations, in these case studies the rate of anomalies are set to occur 50% of the time as the default setting. Nevertheless, this does not change the fact and problem that there is a massive imbalance between normal and abnormal classes in real life.

Hence, in order to match the anomalies with the normal samples in the inference phase, the test set is balanced with $\sim$ 50% normal and $\sim$ 50% anomalies. Specifically, Table 5.2 summarizes the total counts of images generated by their respective augmentation pipeline, all having dimensions $(1, 128, 128)$. It is important for the data to be reproducible through runs, and to ensure the evaluating is not based on a particular "lucky" split of the data, random splits have been performed.

In order to investigate the anomaly ratio's impact on the models, different anomaly percentages are varied. The sample counts are calculated upon fixing a total count, and Table 5.3 elaborates the different types of anomalies within $|\mathcal{N}_{\text{test}}|$ and $|\mathcal{A}|$.

Table 5.3: Different anomaly rates used in the inference phases in section § 6. Percentages are approximate and rounded w.r.t. to a fixed $|\mathcal{D}_{\text{test}}| = |\mathcal{N}_{\text{test}}| + |\mathcal{A}|$.

Anomaly Ratio	5%	10%	30%	50%	70%	90%	95%
Chocolate Bar							
Total $\lvert\mathcal{A}\rvert$	20	40	160	340	560	594	599
Total $\lvert\mathcal{N}_{\text{test}}\rvert$	380	360	350	340	240	66	32
Potato							
$\lvert\mathcal{A}_{\text{metal}}\rvert$	10	20	72	175	273	279	285
$\lvert\mathcal{A}_{\text{hollow}}\rvert$	10	21	72	175	273	279	285
Total $\lvert\mathcal{A}\rvert$	20	41	144	350	546	558	570
Total $\lvert\mathcal{N}_{\text{test}}\rvert$	380	369	336	350	234	62	30

CONTENTS

DEMONSTRATIONS

6

In this chapter, the anomaly detection techniques introduced in chapter § 4 is applied onto the dataset described in chapter § 5.

A large quantity of training samples is needed to train a model so that it can learn the best possible weights. Usage of a *graphics processing unit (GPU)* is therefore preferable to reduce the time of training, as GPUs can significantly accelerate the training process for many deep learning models.

In all algorithms the normal and anomaly data are correctly labelled, so basically it is known what are normal and what are abnormalities. But do keep in mind, that in real-life sorting scenarios this is not the case, which is exactly why applications of novelty detections training on only normal data are at interest. The correctly labelled data is simply used to evaluate the models and their performances.

6.1 IMPLEMENTATIONS OF CAE AND CVAE

The implemented architectures of the CAE and CVAE is described in sections § 6.1.2 and § 6.1.1, respectively. Both have been constructed with the open-source library `TensorFlow 2.3.0` along with `Keras 2.4.3`. Throughout the applications, a couple of `Keras Callbacks` have been implemented, which includes:

- **Check-pointing model with weights**. Model weights are saved after the first epoch as well as after each 200^{th} epochs.
- **Early stopping**. During training, the model will at some point start to make very marginal improvements or simply have converged. Early stopping can halt overfitting and significantly reduce the training time, as it terminates the training at this point [27]. Due to minimal loss improvements, the pre-specified patience to wait before stopping the training process is set to 50 epochs.

Batch sizes usually takes values of $\{32, 64, 128, 256\}$ [27]. In all model implementations and each training session the networks is trained for maximum 2000 epochs with a batch size of 256 on a GPU powered by NVIDIA Tesla P100s.

6.1.1 Architecture of CVAE

Special considerations have been taken into account upon building the main architecture of the CVAE. Especially, a modified version of the deep convolutional generative adversarial network (DCGAN) architecture in [60] has been adapted as it has proven to be very successful for generative models. The article presents a recommended guideline and adjusting it to the language of the CVAE some of its practices have carefully been selected.

Leaky ReLU activation have been employed for all layers in the encoder, expect for onto the bottleneck. No activation function for the $\boldsymbol{\mu} \in \mathbb{R}$ and $\boldsymbol{\sigma} \in \mathbb{R}$ layers is appropriate as they are unconstrained; instead they get send though a linear activation to allow for any value. Expect for the dense and output layer, ReLU activations have been used in all layers in the decoder. The ReLU and leaky ReLU activations are utilized in the intermediate layers for speed, and the logistic sigmoid is used as the output activation of the decoder to render pixel values back to the domain of $[0, 1]$. Meanwhile, the hyperbolic tangent is used as the immediately activation following the latent dimension, as the bottleneck assumingly follows an isotropic Gaussian distribution. Additionally, the hyperparameter for the leaky ReLU has been set to $\alpha = 0.1$ as it showed better reconstruction capabilities compared to the standard value $\alpha = 0.01$.

Furthermore, in accordance with [60, 69] it is recommended not to use (max) pooling layers, but instead be replaced in favour of using strided convolution to perform downsamplings in the encoder, hence strides of size 2 together with kernel sizes $[2, 2]$. Subsequently, fractional strided convolutions should be used for the upsampling process in the decoder. In addition, batch normalization is omitted from the network as it have shown to interrupt the probabilistic sampling process of the CVAE and hurts the performance of the model [42]. Adequate results are found by letting all convolutional layers have 32 filters.

Added dropout has shown to result in more blurriness in both generation and reconstruction of the CVAE [79]. As the model is to be exploited as a static novelty detector, the objective is to get as precise reconstructions as possible, and so the dropout scheme has not been included in any part of the architecture. Hence, the CVAE foregoes explicit regularizations and relies entirely on the image augmentations to produce a self-regularized neural network. Nevertheless, in [33] recent work has found that augmentation alone equals or betters the test performance compared to augmentation plus regularization techniques. I.e. performing heavier image augmentations increases the total number of data images and domain knowledge, which in turn allows the network to self-achieve its own regularization as it helps the CNN towards better mimicking the human visual processing.

Having taken all the above considerings into account, the main architecture of the CVAE is seen in Tables 6.1 and 6.2. A single dense bottleneck layer

Table 6.1: Encoder architecture of the CVAE. $J = \{2, 4, 8, 16, 32, 64, 128\}$ denotes the different latent dimensions used.

Encoder					
Layer	Params	Kernel	Padding	Stride	Output Shape
Input					(128, 128, 1)
Conv2D LeakyReLU	filters = 32 $\alpha = 0.1$	[2 x 2]	'same'	2	(64, 64, 32)
Conv2D LeakyReLU	filters = 32 $\alpha = 0.1$	[2 x 2]	'same'	2	(32, 32, 32)
Conv2D LeakyReLU	filters = 32 $\alpha = 0.1$	[2 x 2]	'same'	2	(16, 16, 32)
Conv2D LeakyReLU	filters = 32 $\alpha = 0.1$	[2 x 2]	'same'	2	(8, 8, 32)
Conv2D LeakyReLU	filters = 32 $\alpha = 0.1$	[2 x 2]	'same'	2	(4, 4, 32)
Flatten					(512)
Dense LeakyReLU	size = 200 $\alpha = 0.1$				(200)
2 x Dense Linear	size = J^*				2 x (J^*)

Table 6.2: Decoder architecture of the CVAE. $J = \{2, 4, 8, 16, 32, 64, 128\}$ denotes the different latent dimensions used.

Decoder					
Layer	Params	Kernel	Padding	Stride	Output Shape
Input					(J^*)
Dense Tanh	size = 512				(512)
Reshape					(4, 4, 32)
TransConv2D ReLU	filters = 32	[2 x 2]	'same'	2	(8, 8, 32)
TransConv2D ReLU	filters = 32	[2 x 2]	'same'	2	(16, 16, 32)
TransConv2D ReLU	filters = 32	[2 x 2]	'same'	2	(32, 32, 32)
TransConv2D ReLU	filters = 32	[2 x 2]	'same'	2	(64, 64, 32)
TransConv2D ReLU	filters = 32	[2 x 2]	'same'	2	(128, 128, 32)
TransConv2D Sigmoid	filters = 1	[2 x 2]	'same'	2	(128, 128, 1)

have been kept in the encoder as it showed to generate richer reconstructions compared to removing it. In particular, finding an optimal dimensionality of the latent vector is often a matter of guessing and varies for the application at hand and so this main architecture has been ran using different latent dimensions, specifically $J = \{2, 4, 8, 16, 32, 64, 128\}$ for testing a wide range range. In general it is expected that larger latent dimensions leads to smaller reconstruction loss, such that the decoder outputs higher-quality reconstructions [9]. Half padding is used to zero pad the input feature maps. Specifically, the `Keras` command `padding = 'same'` is the accordingly operation to be used [66].

6.1.2 *Architecture of CAE*

With the same reasoning as for the CVAE, in the CAE main architecture any pooling layer is replaced by strided and fractional strided convolutions for the encoder and decoder, respectively. Furthermore, dropout regularization has been foregone in favour of letting the network rely entirely on heavy image augmentations. ReLU activation has been used in all layers of the encoder, while Leaky ReLU given $\alpha = 0.1$ has been used in all layers of the decoder, expect for the output layer where the sigmoid activation has been used in order to render values into the range between 0 and 1. However, as the CAE is not a true generative model, in the light of the latent input variable not having a probability distribution associated to it and thus not being able to generate synthesis data, batch normalization has been added to its architecture in favour of speeding up training.

The main architecture of the CAE is found in Tables 6.3 and 6.4, and is trained using the different latent dimensions $J = \{2, 4, 8, 16, 32, 64, 128\}$.

6.2 INVESTIGATION OF LOSSES

Recall, during the back-propagation stage the model aims to minimize the loss function in each mini-batch, but the model is trained as the average of loss functions, i.e. the cost function. Formally, the loss functions given by eqs. (4.1) and (4.19), respectively, the convolutional autoencoder (CAE) is trained using the MSE:

LOSS FUNCTION FOR AUTOENCODER

$$\mathcal{L}_{\text{AE}} = \frac{1}{N} \sum_{i=1}^{N} \left(x^{(i)} - \tilde{x}^{(i)} \right)^2, \tag{6.1}$$

and utilizing the strictly Gaussian case for the CVAE:

Table 6.3: Encoder architecture of the CAE. $J = \{2, 4, 8, 16, 32, 64, 128\}$ denotes the different latent dimensions used.

Encoder					
Layer	Params	Kernel	Padding	Stride	Output Shape
Input					(128, 128, 1)
Conv2D BatchNorm LeakyReLU	filters = 32 $\alpha = 0.1$	[2 x 2]	'same'	2	(64, 64, 32)
Conv2D BatchNorm LeakyReLU	filters = 32 $\alpha = 0.1$	[2 x 2]	'same'	2	(32, 32, 32)
Conv2D BatchNorm LeakyReLU	filters = 32 $\alpha = 0.1$	[2 x 2]	'same'	2	(16, 16, 32)
Conv2D BatchNorm LeakyReLU	filters = 32 $\alpha = 0.1$	[2 x 2]	'same'	2	(8, 8, 32)
Conv2D BatchNorm LeakyReLU	filters = 32 $\alpha = 0.1$	[2 x 2]	'same'	2	(4, 4, 32)
Flatten					(512)
Dense LeakyReLU	size = J^* $\alpha = 0.1$				(J^*)

Table 6.4: Decoder architecture of the CAE. $J = \{2, 4, 8, 16, 32, 64, 128\}$ denotes the different latent dimensions used.

Decoder					
Layer	Params	Kernel	Padding	Stride	Output Shape
Input					(J^*)
Dense BatchNorm ReLU	size = 512				(512)
Reshape					(4, 4, 32)
TransConv2D BatchNorm ReLU	filters = 32	[2 x 2]	'same'	2	(8, 8, 32)
TransConv2D BatchNorm ReLU	filters = 32	[2 x 2]	'same'	2	(16, 16, 32)
TransConv2D BatchNorm ReLU	filters = 32	[2 x 2]	'same'	2	(32, 32, 32)
TransConv2D BatchNorm ReLU	filters = 32	[2 x 2]	'same'	2	(64, 64, 32)
TransConv2D BatchNorm ReLU	filters = 32	[2 x 2]	'same'	2	(128, 128, 32)
TransConv2D Sigmoid	filters = 1	[2 x 2]	'same'	2	(128, 128, 1)

LOSS FUNCTION FOR VARIATIONAL AUTOENCODER

$$\mathcal{L}_{\text{VAE}} = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \tilde{x}^{(i)}\right)^2 - \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log_e\left(\sigma_j^{(i)}\right)^2 - \left(\mu_j^{(i)}\right)^2 - \exp\left(\log_e(\sigma_j^{(i)})^2\right)\right), \tag{6.2}$$

with reparameterization trick $\mathbf{z} = \boldsymbol{\mu} + \exp(0.5 \log_e \boldsymbol{\sigma}^2) \odot \boldsymbol{\epsilon}$. Notice, in order to ensure numerical stability in the latent space, and without loss of generality, $\log_e \boldsymbol{\sigma}^2 \in \mathbb{R}$ is learning instead of the constrained $\boldsymbol{\sigma}^2 \in \mathbb{R}^+$.

Given $J = 64$, the average loss during training for both the CVAE and CAE models described by Tables 6.1-6.2 and 6.3-6.4, respectively, is seen on Fig. 6.1 for both the chocolate bar and the potato datasets. The initial learning rates have been chosen based the technique described in section § 6.2.1 below. Too many models have been trained to consider them all in one illustration, however in Appendix § A the reaming learning curves can be found.

In general the training cost is very large for the first couple of epochs whereafter it rapidly falls and slowly converges. It is evident that the average losses monotonically decreases as the epochs increases, which constitutes a sought converging behaviour. Moreover, it seems that the CVAEs have an easier time to generalize in the early training phase compared to the CAEs.

Furthermore, from the individual model's loss curves it can be seen they converges nicely towards the same tendencies as both the training and validation metrics are moving towards the same direction simultaneously. The small differences between the two learning curves are normal observations, and there are no indication of the models either overfitting or underfitting. The deep networks could be allowed to continue training, but the losses would simple oscillate minimally as a plateau in terms of what the capacity of the network is capable of learning has been reached. Furthermore, as training progresses, the model's ability to generalize to the unseen validation set tends to drop — and at this point the early stopping can help as an regularization technique as it stops the training when the performance degrades.

6.2.1 *Learning Rate Finder*

The Adam optimizer described by eq. (2.21) has been used in all implementations. The initial configurable exponential decay rates have been set to the `Keras` default values which are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate however has been chosen bases on the so-called *learning rate finder* method for each and every implemented model configuration. As discussed in § 2.1.5, the learning rate is the hyperparamter that has the largest impact during training of a model and therefore extra care have been taken into account.

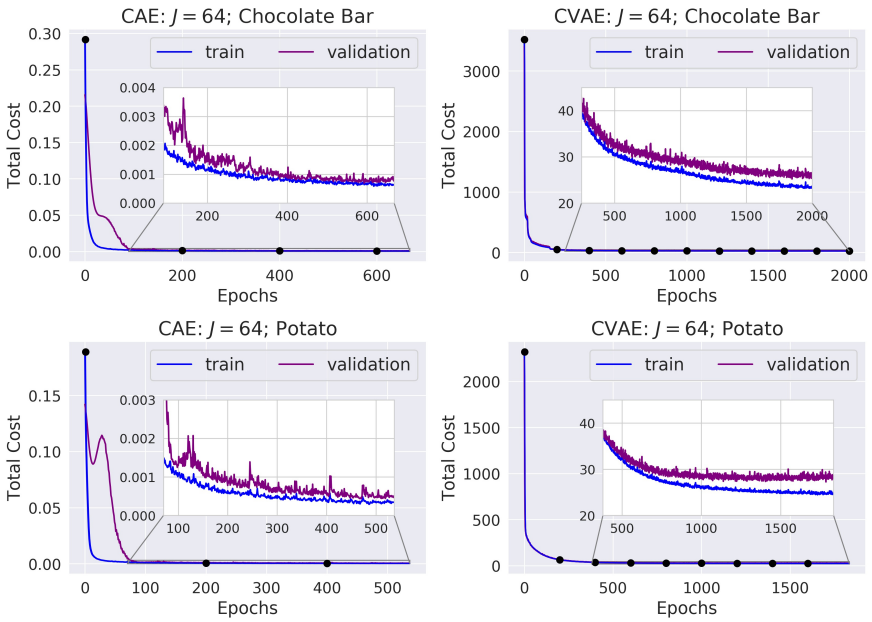


Figure 6.1: Average loss functions as a function of number of epochs, derived on training images. The points at certain epochs indicates check-point savings. Upper left: CAE, $J = 64$, chocolate bar. Upper right: CVAE, $J = 64$, chocolate bar. Lower left: CAE, $J = 64$, Potato. Lower right: CVAE, $J = 64$, Potato.

The approach to systematically find a learning rate was developed and presented in [68]. Specially, it was found there is enough information available to tune the learning rate simply by monitoring the loss early in the training. The idea is to initialize the model with a small learning rate and train it on a mini-batch of data. The learning rate is then increased exponentially at every iteration, so that each new mini-batch is trained using a larger learning rate than the previous one, until a high learning rate is reached or the loss value explodes. This entire training process typically only takes a couple of epochs to complete.

In Fig. 6.2 the associated smoothed losses to the learning rates for each iteration is plotted for the CVAE model given $J = 128$ for the potato dataset. The learning rate is picked by examining this plot. In the beginning for very low rates $\alpha_0 < 10^{-3}$ the loss is essentially flat as the rates are to small for the network to start learning. A little before 10^{-3} the network can actually learn as the loss begins to decrease. The network begins to learn very quickly in the range between 10^{-3} and 10^{-2} as the loss is rapidly decreasing. A little after $\alpha_0 > 10^{-2}$ is reached the rate is far to large as the loss begins to explode.

Contrary to belief, the optimal learning rate is not the value for which the loss reaches its minimum, but where the rate creates the greatest decrease in loss, i.e. where the loss is strictly decreasing at a rapid rate. The minima value is at the edge between improving the loss and fluctuating towards explosion, thus this valley is already a little to high of an learning rate. Instead, by choosing one order of magnitude smaller the model will still train quickly but be on the safe side from an explosion. Hence, for this model 10^{-3} is chosen as the initial learning rate.

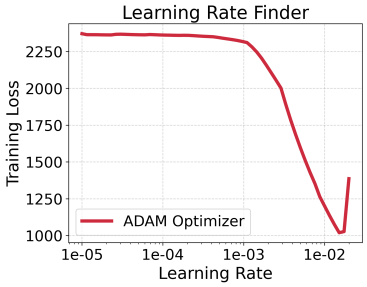


Figure 6.2: The results of the learning rate finder of the Adam optimizer on the CVAE model given $J = 128$ and using the potato dataset. Smoothed losses are plotted against the log-scaled learning rates. A good initial learning rate would be in the range where the loss is rapidly, but strictly descreasing, i.e. the region $[10^{-3}, 10^{-2}]$. Such an analysis is performed for every implemented model.

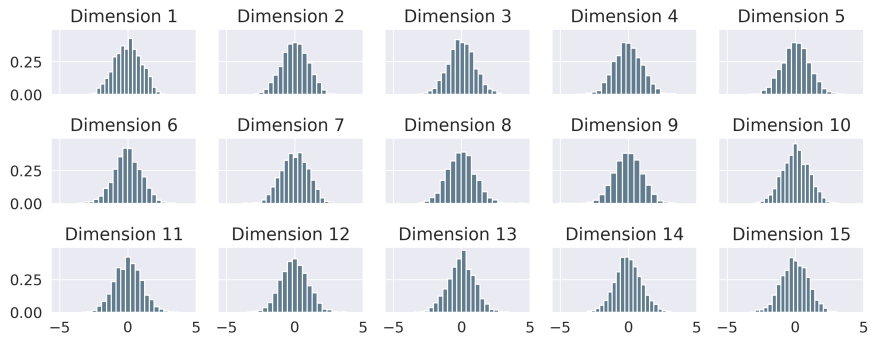


Figure 6.3: 15 random selected normalized histograms of the learned low-dimensional latent representations. Constructed from the CVAE model given $J = 32$ and the full chocolate bar dataset consisting of only normal samples.

6.3 RESULTS

6.3.1 *Entering the Encoded Space*

The CAE and CVAE models have been trained to reduce the set of images into a low-dimensional space made up of a limited set of J latent dimensions. Using the encoder half of either the CAE or CVAE models, each image is rendered to a compact data point set and can be represented in these J multi-dimensions, i.e. in the *encoded space*. This mapped space enable one to perform various analysis as it can be utilized as powerful tool to e.g. capture reduced visual structures or generalize the underlying features and variations into, hopefully, a spectrum of patterns or even emerging clusters.

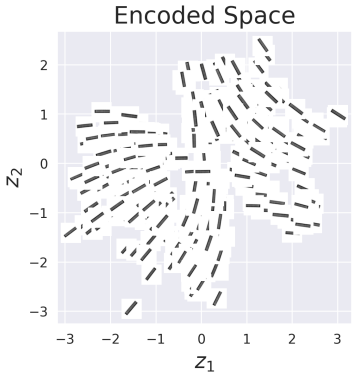


Figure 6.4: Two-dimensional embedding map of the encoded space. Result from the CVAE model given $J = 2$ and normal testing images of the chocolate bar dataset.

It is good practice to analyse the distribution of the extracted bottleneck layer of the CVAE model. As an example consider the CVAE model trained with $J = 32$ using the chocolate bar dataset. Due to space limitations, Fig. 6.3 shows the normalized histograms of 15 randomly chosen latent dimensions out of the possible 32. Note that each histogram is constructed using the corresponding hidden dimension for every normal sample in the full dataset. According to the CVAE model set-up, as described in § 4.2.2, all hidden latent dimensions are expected to be normally distributed — and not be separated in e.g. a mixture of Gaussians. Happily, this is indeed the case, as each dimension in general roughly follows a smooth and normal distribution.

This forced penalty of the CVAE model assists the network to learn the underlying low-dimensional data manifold and capture its local structures. Still considering the chocolate bar dataset, but in favour of simplicity, regard the resulting encoded space of the CVAE model given $J = 2$ as shown in Fig. 6.4. Each encoded dimensions describe a position of each image within this embedding space, and essentially this plot shows how similar or dissimilar two images are. Only the normal testing samples have been visualized, however it is clear to see a spectrum of variations, in particular varying rotations of the chocolate bars. This means that the CVAE model has successfully learned from higher order features of the training images, resultantly found some common specific characteristic between them — primary the rotation

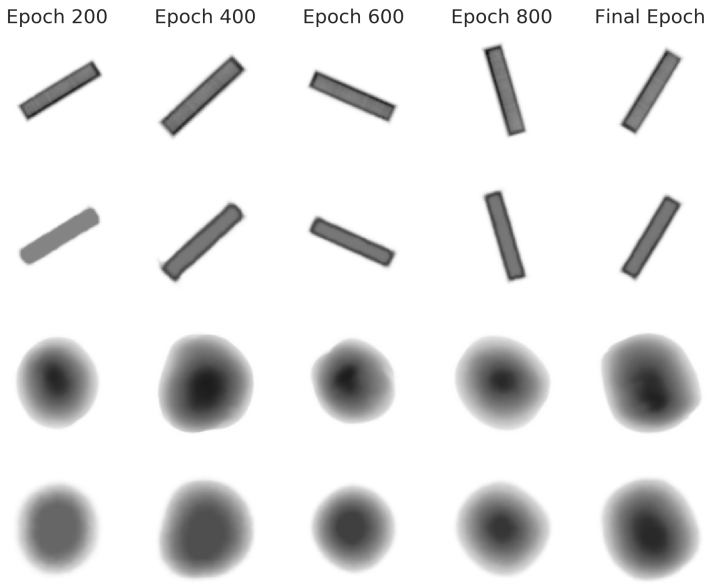


Figure 6.5: Randomly selected reconstructions of training images derived at different epochs. Upper rows for each datasets constitutes the inputs and lower rows the outputs of the CVAE networks at certain epochs. For the chocolate bar dataset $J = 8$ has been utilized, and for the potato dataset $J = 32$.

degrees — and are able to represent them in a set of normally distributed latent variables.

6.3.2 Evaluation of Reconstructions

Oppositely, the decoder half of the CAE and CVAE models takes the J latent dimensions representing a single image as their input, and have been trained to produce a version of the original image that is reconstructed from the encoding. If the networks have been well-trained, these reconstructed images will preserve the features the autoencoder has learned from, but they will not maintain every details from the original images.

In Fig. 6.5 the CVAE network's reconstructions of random training inputs can be seen at certain epochs for which the average loss is derived during training. In accordance with Fig. 6.1, it is evident that the models gradually increases their performance as the loss decreases with the number of epochs being run — the reconstructions becomes more and more alike with the input.

To empirically investigate the models' ability to generalize to unseen data some example reconstructions of test data derived at the models' final epochs are presented in Fig. 6.6 for both datasets. Promptly, there is a couple of noticeable observations worth discussing while examining these outcomes.

First and foremost, when inspecting the reconstructions by the CVAEs one immediately notice that the model has problems with generating sharp edges, and instead results in slightly blurred representations. This blurriness is a natural limitation to the CVAE as images are sampled using the Gaussian

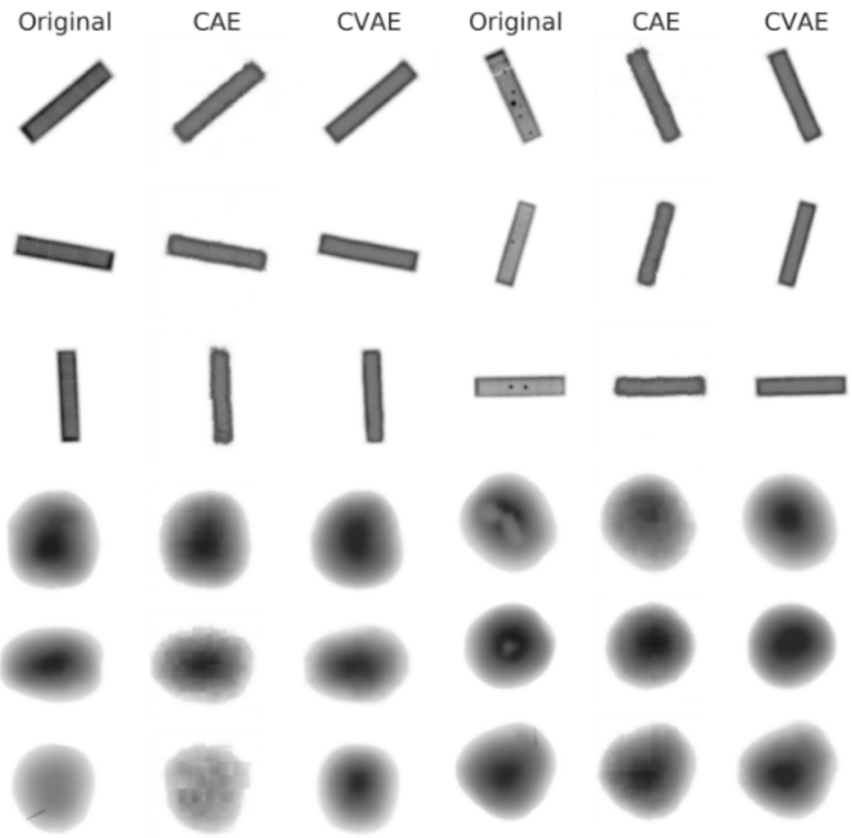


Figure 6.6: Randomly selected reconstructions of test images derived at the final epoch. Especially, notice the difference between the reconstructions of the CVAE models compared to the CAE model, as well as how they all fail to reconstruct anomalous samples. Here, $J = 8$ is considered for both the CAE and CVAE model given the chocolate bar test set. To the left normal samples are reconstructed by the models, and to the right anomalous samples are reconstructed. On the other hand, $J = 32$ has been considered for the potato test set for both the CAE and CVAE model. In the left first two rows perfect potatoes are reconstructed, and to the right hollow potatoes are considered. In the last row potatoes with metal in them are reconstructed.

assumption. Furthermore, using the MSE loss usually yields blurry results [20], and therefore blurriness is also observed in the resulting images of the CAEs. The chocolate bars have sharp edges compared to the potatoes with their ellipsoidal and rounder shapes, whereas it follow that the models have an 'easier' time generating the potatoes compared to the chocolate bars.

Secondly, the quality of the reconstructed images are, as anticipated, lower than that of the original inputs. In particular, there is a loss of textures and details; primary for the chocolate bar dataset for which the biscuit-like texture has disappeared. Nevertheless, the models shows their ability to reconstruct the images of the food products to a certain degree. Furthermore, comparing the reconstruction derived at the models' final epoch it is slightly apparent that the networks' performances are better on the training data than on the testing data, cf. Fig. 6.5. This is however an expected result.

Next, and the most important finding; when comparing the models' reconstructions of normal and anomaly samples there if a remarkable difference between the networks' ability to capture and reconstruct details. The quality of the models' reconstructions indicates they are good at reconstructing the normal samples, whereas they fail to reconstruct foreign objects ect. found on the anomaly instances. This malfunction is due to the fact that the models are solely trained on normal samples and thus have no idea what to do when

they encounter something they have never seen before. Still, empirically they strive to reconstruct the anomalous samples as being similar to the normal samples.

Hence, the networks have successfully learned from not only individual pixel intensity values, but also considers larger areas of the image, which empower them to extract insights from specific properties such as features, structures and illumination. Importantly, do notice the intensity differences between normal samples and anomaly samples — especially noticeable for the chocolate bars and the potatoes with inserted metal. This distinction is something that will be revisited in § 6.4.2, but for now simply bear it in mind. Nevertheless, it seems that, at least the CVAE models, have successfully managed to become robust to the augmentations applied, i.e. invariant to size, position and orientation, hence essentially boosting the overall performance.

Following, the CAEs attempts to represent inherent characteristics of the original images. However, contrary to the CVAE that enforces the encodings to fit a normal distribution during training, the CAE does not. Hence, the produced encoded representations of the CAE does not follow any kind of pre-defined structure or are limited in some confined space, which may make them hard to understand. Recall, the decoder attempts to reconstruct the original image using just the latent space representation. However, due to the usually lack of interpretable and exploitable structure of the CAE bottleneck, this network cannot produce any new content. However, the CVAE with its regularized and organised encoded distributions are able to cope with varieties in shape etc., hence able to generate new data.

I.e. as the CAE is solely trained to encode and decode with as minimal loss as possible it does so without taking into account the organisation of the bottleneck. To achieve this task the CAE will take advantage of any over-fitting possibilities during training. By introducing the regularization term to the CVAE, this model will ensure good properties in the encoded space that enables generative processes and avoids the same amount of over-fitting during training. All this is reflected in the reconstructions, where it is obvious that the CVAE model generates more precise reconstructions compared to the CAE model — especially for the potato set.

6.3.3 Static Novelty Detection: Automated Threshold Selection

As was explained in section § 3.1, recall the necessity of some boundary or threshold needed to distinguish between normal and abnormal instances in static novelty detection scenarios. The CAE and CVAE are both exploited as static novelty detections, and since their objective is similar, a common threshold metric as well as other evaluation metrics are ideal for comparisons.

The squared Euclidean distance, also known as the squared error (SE), is the

sum of all the squared differences between the ground truth and the predicted value. This particular metric has been chosen as the reconstruction error as it is a measure that puts a progressively larger weight on larger differences between pixel values, since the squares cause it to be very sensitive to large outliers:

RECONSTRUCTION ERROR

$$\text{SE} = \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2 = \sum_{k=1}^{HW} \left(x^{(k)} - \tilde{x}^{(k)}\right)^2. \tag{6.3}$$

I.e. accordingly to the gray levels of each pixel, the Euclidean distance converts each image $\mathbf{x} \in \mathbb{R}^{H \times W}$ and $\tilde{\mathbf{x}} \in \mathbb{R}^{H \times W}$ into vectors, whereas it compares the differences pixel by pixel. Together with various other anomaly scores, the Euclidean distance is further examined in § 6.4.2.

When the CAE and CVAE algorithms are trained solely on normal data instances they fail to reconstruct the anomalous data samples. It is therefore expected that the reconstruction error, as the anomaly score, have lower values if given normal test samples, while the error becomes larger given anomalous samples [62, 76]. Based on this tendency, simple thresholding can discriminate between those two classes. I.e. using eq. (6.3) a threshold value, ξ, must be chosen to determine if an image belongs to either the normal class or the anomaly class. This boundary must be carefully selected, and the aim is to maintain a large true positive (TP) rate whilst eliminating false positives (FPs). Samples with reconstruction error larger than ξ will be classified as anomaly, and as normal otherwise.

As an illustrative example, consider for now the CVAE model given $J = 32$ for the chocolate bar dataset provided a 50/50 split rate between true normal and anomalies samples in the test set. By computing the anomaly score given by (6.3), and knowing the true labels, the normalized sampling histograms in Fig. 6.7 have been obtained. As expected, compared to the anomaly score for the normal samples, the reconstruction error is larger for the anomalies. Furthermore, notice the histogram statistic of the two sampling distributions. Not surprisingly, their shapes both approximates (single-peaked) Gaussian distributions, and the spread of the distribution of the anomalies are larger than that of the distribution of the normal samples. As the overlap between the two distribution is very small, this is rather an exemplary case — and very close to an ideal case (where no overlap is present). Nevertheless, the next problem is now to figure out how to separate these two histograms in the best way possible, i.e. obtaining an ideal measure of separability.

The quick and dirty way to choose a threshold value would be by eye, but using such non-qualitative technique will never obtain the best possible cut-off. Instead, the *receiver operating characteristic (ROC)* curve can partly be used to determine an optimal optimal threshold value, ξ^*. The ROC assessment technique utilize the true positive rate ($\text{TPR} = \text{TP}/(\text{TP} + \text{FN})$) and false positive rate ($\text{FPR} = \text{FP}/(\text{FP} + \text{TN})$), where FN and TN denotes *false*

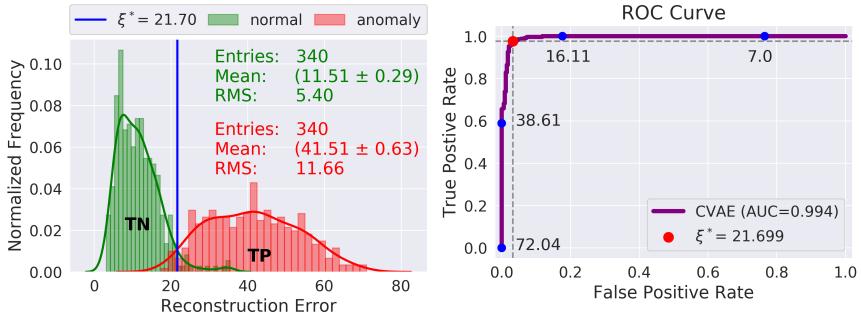


Figure 6.7: Results of the CVAE model given $J = 32$ for the chocolate bar dataset given an anomaly rate of 50%. Left: Normalized sampling distributions calculated upon eq. (6.3) and whereas the true labels are known. Right: Corresponding ROC curve for which the G-MEAN is utilized to determine on optimal threshold value.

negative and *true negative*, respectively [32]. By definition, the ROC space depends on the relative trade-offs between the benefits of TPs and costs of FPs, and so each point represents one predicted result.

Specifically, for this problem the ROC is calculated between the true labels and target anomaly score defined by eq. (6.3) for all test samples. Such a resulting ROC curve is seen on Fig. 6.7. As illustrated on this ROC curve, the threshold value can be adjusted, which in turn will change the point of separation between the sampling distributions. E.g. increasing the threshold value, corresponding to a leftward movement on the ROC curve, will result in fewer FPs at the cost of more false negatives (FNs). In general, the closer the ROC curve is to the upper left corner, the more efficient the model is. Subsequently, the objective is now to determine the threshold point in ROC space that produces the best possible prediction.

Since the two-class test set is severely imbalanced (in real life), the *geometric-mean (G-MEAN)* metric can be used to fit the optimal threshold on the ROC curve [49]. It seeks to balance the TPR and the true negative rate (TNR = TN/(TN + FP) = 1 − FPR), and is calculated for various threshold values. The G-MEAN aggregates both metric into a single value in $[0, 1]$ and assigns equal importance to both [21]:

$$\text{G-mean} = \sqrt{\text{TPR}(1 - \text{FPR})}. \tag{6.4}$$

Hence, by calculating the G-MEAN for all rates and optimizing it:

OPTIMAL THRESHOLD SELECTION

$$\xi^* = \arg\max_{\xi} \text{G-mean}, \tag{6.5}$$

the index of the largest G-MEAN value is located and consequently used to find the optimal threshold value. Hence, ξ^* is automated selected through this process.

Consider again Fig. 6.7. By use of eqs. (6.4)-(6.5) an optimal threshold value has successfully been determined as the ideal measure of separability, and is graphically placed in the upper left corner on the ROC curve. Following the

color codes on the plot over the sampling distributions, TPs are designated as anomaly (red) instances on the right of the threshold line. Consequently, true negatives (TNs) corresponds to normal (green) instances to the left of the line. FPs denotes normal samples wrongly predicted as anomalies, i.e. the green instances to the right of the line. On the other hand, red instances to the left of the line corresponds to anomaly samples being wrongly predicted as normal samples, i.e. FNs. In § 6.3.4 below, statistic over these different outcomes will be used for evaluating and comparing the models.

6.3.4 Evaluation Metrics

When comparing the performance of different learning algorithms using different data sets, evaluation metrics are used as measuring sticks. They must capture the details of what the frame of the problem seeks to solve, henceforth the selection of metrics is crucial. In this section the chosen metric are shortly introduced and later on in section § 6.3.5 used to evaluate the model implementations given their hyperparameters.

First and foremost, a *confusion matrix* is a cross table that records the number of occurrences and represents the proportion between true and predicted instances of classes, i.e. one point in ROC space. Computed directly from the confusion matrix, the *accuracy* and *error rate* (or *misclassification*) are used to describe a classifier's overall performance as they are calculated by [32]

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \qquad \text{error rate} = 1 - \text{accuracy}. \tag{6.6}$$

However, theses metrics are highly sensitive to the data distribution and can be deceiving for imbalanced datasets, as the majority classes will have a larger weight compared to the minority classes, henceforth the accuracy tends to hide strong classification errors for the smaller classes [29, 32]. Accordingly, in addition to the the accuracy, a couple of other frequently used metrics are evaluated to provide comprehensive performance representations of the models.

As both FPs and FNs are considered equally important and costly, the traditionally F_1-score $\in [0, 1]$ for a single class label is computed by [29]

$$F_1\text{-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \tag{6.7}$$

where

$$\text{precision} = \text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{recall} = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{6.8}$$

Precision expresses the total proportion of occurrences that are predicted positive and they actual positives, i.e it is a measure of how trustful the model is when predicting an unit as positive — namely as abnormal. *Recall* (or *sensitivity*), on the other hand, measures the proportion of actual positives that was identified correctly, i.e. it gives a measure of how accurately the model is able to identify relevant data. Thus, following the set-up of the problem, it specifies the ratio of how many anomaly samples are correctly identified as being abnormal. Precision and recall are dependent on each other, and changing the threshold value changes their rates accordingly, which is why the F_1-score is needed to seek a balance between them. Specifically, to figure out how the system performs overall across all class instances, it is the *macro-average F_1-score* that is evaluated. It is simply the harmonic mean of the macro-average precision and macro-average recall such that

$$F_1^{\text{macro}}\text{-score} = \frac{1}{K}\sum_{k=1}^{K} F_1\text{-score}_k, \tag{6.9}$$

given each class label k. Compared to the accuracy, the F_1-score usually gives a better performance estimate if the dataset has an uneven class distribution as it assigns same importance to each class instance.

Furthermore, another commonly reported evaluation metric is the *specificity*, which convey how many occurrences is predicted as negative out of all actual negatives, i.e. the true negative rate:

$$\text{specificity} = \text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{6.10}$$

Hence, this rate specifies how many normal samples are identified as being normal samples.

Finally, the *area under the curve (AUC)* in ROC space is another measurement commonly used when all classes are considered equally important. It is often used as a summary of the ROC curve as it aggregates values into $[0, 1]$. The closer the AUC is to 1.0, the better the average performance of the model at distinguishing between normal and anomaly classes.

6.3.5 Detecting Anomalies

Consider again the same case study introduced in the previous section § 6.3.3 and Fig. 6.7. Together with its confusion matrix, using the predictions of the classes based on the optimal threshold value determined by eq. (6.5), the anomalies in comparison to the normal data points can be visualized, cf. Fig. 6.8. As earlier observed; simply by looking at the sampling distributions this particular model is doing very well at separating normal instances from

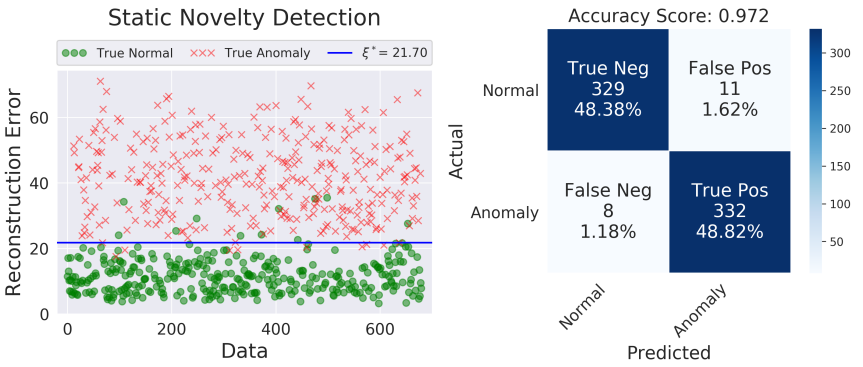


Figure 6.8: Continuing from the case described in Fig. 6.7; CVAE given $J = 32$ and 50% anomaly ratio for chocolate bar test set. Left: Anomalies relative to the optimal threshold value. Right: Visualization of its resulting confusion matrix and accuracy score.

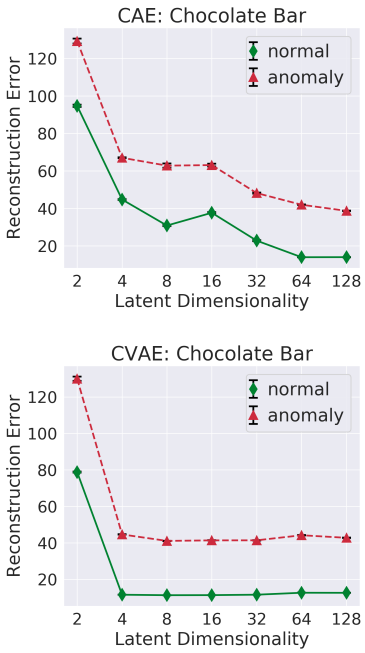


Figure 6.9: The average reconstruction errors and its errors in CAE and CVAE as a function of the latent dimensions, derived on the chocolate bar test set with a 50% anomaly ratio. Errorsbars represents the standard error on the mean. The larger the distance between normal and anomalous, the better the model typically is at correctly predicting the classes.

anomaly instances; an observation that is also backed-up by the metrics utilized for evaluating the model's performance. Green circles above the threshold line on Fig. 6.8 thus corresponds to normal samples being misclassified as abnormal (FPs); vice versa red crosses below the line are analogous to anomalous samples being predicted as normal (FNs). A few samples consistently being predicted wrong (upper row) and correctly (lower row) are shown on Fig. 6.10.

Chocolate bars being predicted correctly with high certainty are anomalous samples with an obvious intensity difference and whose foreign objects are cracks are obvious. Anomalous samples being wrongly predicted are typically samples with only one foreign object or whose overall luminance are closer to the normal samples. However, in general it especially depends on how well the network manages to reconstruct the structure — in particular the sharp edges of the chocolate bars.

As for the potato test set, the networks' ability to reconstruct their inherent shapes also plays a role. Here, they tends to predict perfect samples having non-typically potato shapes being wrong. Furthermore, they are inclined to wrongly predict potatoes with needles in the outer rim and potatoes whose hollow hearts are of smaller volumes as being normal. Even to the trained human eye such wrongly predicted potatoes can be difficult to assess. On the other hand, normal potatoes whose shapes are round or ellipsoidal without any obvious "weirdness", they tend to predict correct. Hollow hearts of larger volumes and potatoes with needles placed further toward the middles they also predict correctly with high certainty.

All the model performances across the evaluation metrics described in § 6.3.4 above are summarized in Table 6.5. In this table an anomaly ratio of 50% has consistently been used in the inference phase. In the following, the obtained results will be discussed in further details.

Consider the example cases on Fig. 6.9. Both the CAE and CVAE are trained

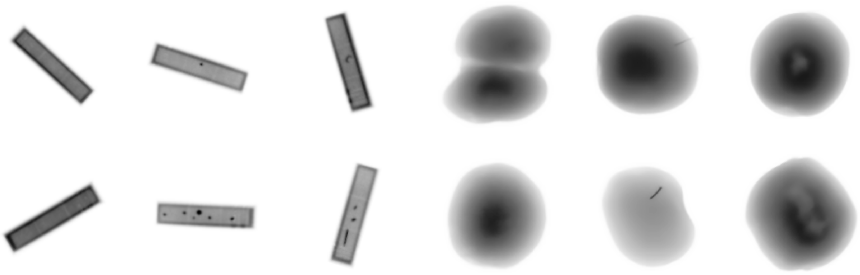


Figure 6.10: Few examples of samples being classified correctly and wrongly with high certainty — both datasets trained by the CVAE given $J = 32$. Upper row: Wrongly predicted samples. Lower row: Correctly predicted samples. First columns from the left for each individual datasets: normal samples. Last two columns for each datasets: anomalous samples.

using the chocolate bar dataset for various latent dimensions, and the average reconstruction errors of a 50% anomaly rate have been computed. Start of by considering the plot over the CVAE examples. The first observation to notice is that the average reconstruction error of anomaly samples consistently are larger than that of normal samples, no matter what the latent dimension is. Secondly, contrary to what was expected of the behaviour of the latent dimensionality, the average reconstruction error does not change significantly for $J > 2$ given this example case, implying that this dataset is pretty robust w.r.t. the latent dimension. The most noticeable finding is the results for $J = 2$, which reconstruction errors and instabilities are larger compared to the results for the remaining latent dimensions. On the other hand, the corresponding plot over the CAE examples corresponds to the expectations of the reconstruction error mainly decreases with increasing latent dimension, expect for $J = 16$ being an apparent outlier and for $J > 64$ the pattern starts to stagnate. These discovered tendencies are also reflected in the reported results found in Table 6.5.

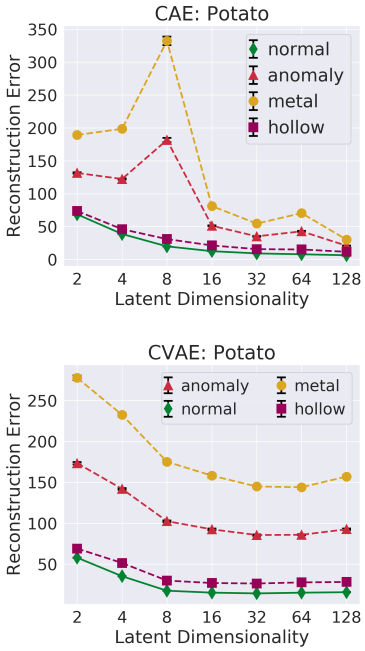


Figure 6.11: The average reconstruction errors and its errors in CAE and CVAE as a function of the latent dimensions, derived on the potato test set with a 50% anomaly ratio. Notice the difference in reconstruction errors for potatoes with foreign objects and potatoes with hollow heart disease. Errorsbars represents the standard error on the mean.

Besides, the same tendency of the CVAE model being more robust w.r.t. to the latent dimensionality contrary to the CAE model is also reflected by performance of the same analysis, but given the potato test set, cf. Fig. 6.11. What is however especially worth noticing in these plots are the differences in the reconstructions errors between potatoes with incorporated metal and potatoes with artificially hollow heart disease. Observe how close the reconstruction error of the hollow potatoes are to the perfect ones, as well as to how much larger the reconstruction error of potatoes with metal in them are. Since these two reconstruction errors are gathered into a single anomaly class, their scores meets in between.

As a consequence, the models are more prone to predict potatoes with metal in them correctly contrary to hollow potatoes. As an example, consider Fig. 6.12 where the CVAE model have been trained given $J = 32$ and the potato dataset with a 50/50 anomaly ratio. As can be seen, the sampling distribution over potatoes with metal in them are flat and wide, while potatoes with hollow hearts overlaps the distribution over perfect potatoes more thoroughly. Accordingly, as shown in the corresponding "pseudo-confusion

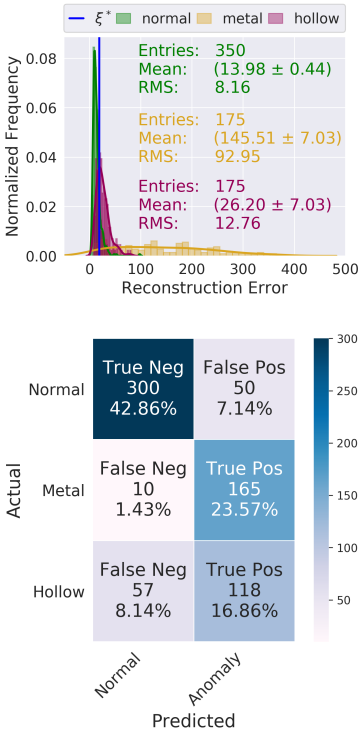


Figure 6.12: Upper: Sampling distributions over perfect potatoes, potatoes with needles and potatoes with hollow hearts as calulated by eq. (6.3) and results of the CVAE model given $J = 32$ for the potato dataset given an anomaly ratio of 50%. Lower: The corresponding "psedu-confusion matrix".

Table 6.5: Comparisons between the implemented algorithms using different datasets, but all having an anomaly ratio of 50%. J is the dimensionality of the bottleneck and different values has been used for both the CAE and CVAE models, whereas the mean results are reported. Here, it is the macro-average F_1-score, AUC-ROC and accuracy that is referred to. Boldface indicates the best individual results across the models. Mean values are reported, and uncertainties are calculated as the standard error on the mean.

CVAE

J	AUC-ROC	Specificity	Recall	Precision	F_1	Accuracy
Chocolate Bar						
2	0.740 ± 0.002	0.668 ± 0.012	0.703 ± 0.013	0.680 ± 0.004	0.685 ± 0.002	0.685 ± 0.002
4	**0.996 ± 0.000**	0.968 ± 0.003	0.973 ± 0.003	0.969 ± 0.002	0.970 ± 0.001	0.970 ± 0.001
8	0.996 ± 0.000	**0.972 ± 0.002**	0.976 ± 0.002	**0.973 ± 0.002**	**0.974 ± 0.001**	**0.974 ± 0.001**
16	0.995 ± 0.000	0.964 ± 0.003	**0.982 ± 0.002**	0.965 ± 0.003	0.973 ± 0.001	0.973 ± 0.001
32	0.996 ± 0.000	0.968 ± 0.002	0.975 ± 0.003	0.968 ± 0.002	0.972 ± 0.002	0.972 ± 0.002
64	0.993 ± 0.001	0.955 ± 0.003	0.965 ± 0.003	0.956 ± 0.003	0.960 ± 0.002	0.960 ± 0.002
128	0.991 ± 0.001	0.959 ± 0.002	0.956 ± 0.002	0.959 ± 0.002	0.957 ± 0.001	0.957 ± 0.001
Potato						
2	0.774 ± 0.002	0.674 ± 0.009	0.707 ± 0.011	0.685 ± 0.003	0.690 ± 0.002	0.690 ± 0.002
4	0.832 ± 0.002	0.715 ± 0.008	0.804 ± 0.009	0.739 ± 0.004	0.759 ± 0.002	0.760 ± 0.002
8	0.873 ± 0.002	0.830 ± 0.005	0.778 ± 0.004	0.821 ± 0.004	0.804 ± 0.002	0.804 ± 0.002
16	0.892 ± 0.002	0.840 ± 0.007	0.801 ± 0.007	0.834 ± 0.005	0.820 ± 0.002	0.820 ± 0.002
32	**0.903 ± 0.002**	**0.846 ± 0.009**	**0.814 ± 0.011**	**0.843 ± 0.006**	**0.830 ± 0.002**	**0.830 ± 0.002**
64	0.894 ± 0.001	0.832 ± 0.005	0.810 ± 0.005	0.829 ± 0.004	0.821 ± 0.001	0.821 ± 0.001
128	0.899 ± 0.001	0.845 ± 0.006	0.809 ± 0.007	0.840 ± 0.005	0.827 ± 0.002	0.827 ± 0.002

CAE

Chocolate Bar						
2	0.640 ± 0.003	0.623 ± 0.005	0.600 ± 0.006	0.614 ± 0.003	0.612 ± 0.003	0.612 ± 0.003
4	0.799 ± 0.003	0.683 ± 0.006	0.855 ± 0.006	0.729 ± 0.003	0.767 ± 0.003	0.769 ± 0.003
8	0.928 ± 0.001	0.853 ± 0.002	0.932 ± 0.002	0.864 ± 0.002	0.892 ± 0.001	0.893 ± 0.001
16	0.914 ± 0.002	0.832 ± 0.003	0.942 ± 0.003	0.849 ± 0.002	0.887 ± 0.002	0.887 ± 0.002
32	0.966 ± 0.001	0.931 ± 0.002	**0.959 ± 0.002**	0.933 ± 0.002	0.946 ± 0.001	0.946 ± 0.001
64	**0.990 ± 0.000**	**0.962 ± 0.001**	0.955 ± 0.002	**0.961 ± 0.001**	**0.958 ± 0.001**	**0.958 ± 0.001**
128	0.989 ± 0.000	0.953 ± 0.003	0.949 ± 0.003	0.953 ± 0.002	0.951 ± 0.001	0.951 ± 0.001
Potato						
2	0.742 ± 0.003	0.717 ± 0.007	0.623 ± 0.008	0.688 ± 0.003	0.669 ± 0.002	0.670 ± 0.002
4	0.803 ± 0.002	0.732 ± 0.013	0.697 ± 0.011	0.725 ± 0.007	0.714 ± 0.001	0.714 ± 0.001
8	0.847 ± 0.003	0.794 ± 0.006	0.757 ± 0.007	0.787 ± 0.004	0.775 ± 0.003	0.776 ± 0.003
16	0.882 ± 0.001	0.798 ± 0.008	0.811 ± 0.008	0.801 ± 0.005	0.804 ± 0.002	0.804 ± 0.002
32	0.870 ± 0.002	0.788 ± 0.004	0.831 ± 0.006	0.797 ± 0.002	0.809 ± 0.002	0.810 ± 0.002
64	0.873 ± 0.002	0.785 ± 0.007	0.816 ± 0.007	0.792 ± 0.005	0.800 ± 0.002	0.800 ± 0.002
128	**0.894 ± 0.001**	**0.826 ± 0.004**	**0.838 ± 0.004**	**0.828 ± 0.003**	**0.832 ± 0.002**	**0.832 ± 0.002**

matrix", hollow potatoes gets misclassified as being normal to a greater extent than potatoes with needles in them does.

Impact of the Anomaly Ratio

The effect of the anomaly ratio, given the numerical values found in Table 5.3, is now investigated. For completeness, and by screening Table 6.5, the models with the best overall performances in accuracy and F_1-score are empirically chosen for each dataset. Subsequently, the anomaly ratio is varied in order to examine its impact and the same evaluation metrics are reported in Table 6.6 given different ratios.

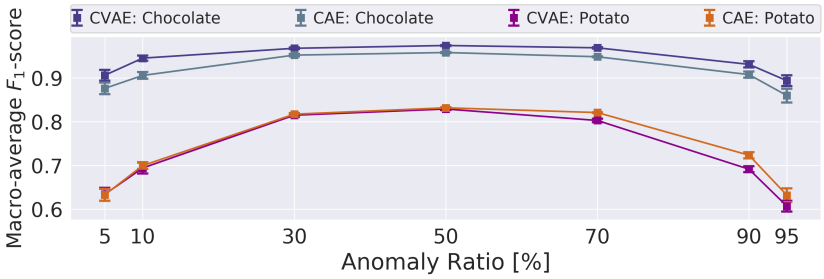


Figure 6.14: Mean macro-average F_1-scores across various anomaly ratios. The error bars represent the standard error on the mean.

Consider first the chocolate bars. From the results in Table 5.3 each individual models who did best for this dataset are the CVAE given $J = 8$ and CAE given $J = 64$. Notice, as in accordance with Fig. 6.9, the results for the CVAE model given $J > 2$ are all pretty close to each other since the reconstruction errors starts to flatten out for increasing latent dimensionality. As for the CAE model the results in general gets better with increasing latent dimensionality, which is also in line with the figure.

Moreover, no matter which individual latent dimensions are compared, the CVAE model seems to have performed better than the CAE model for all evaluation metrics. This is linked to the CVAE's ability to reconstruct and create variations of the images to a greater extent. The CAE lacks the capability of handling the variations such as illumination, orientation and shapes of the samples, thus resulting in worse reconstructions.

However, it actually seems like the CAE model given the potato dataset yields better results compared to the CVAE model in some of the larger latent dimensions(!) However, take then a look at some of the reconstructions of the CAE model given $J = 128$ in Fig. 6.13. The reconstructions are, to say, pretty useless as the model fails to capture the inherent features of potatoes. The results reported by this model are pretty much associated with 'luck' and the reconstructions suggest the need of additional variations in the training set. In summary, it is expected that in real life applications this model would be ineffectual.

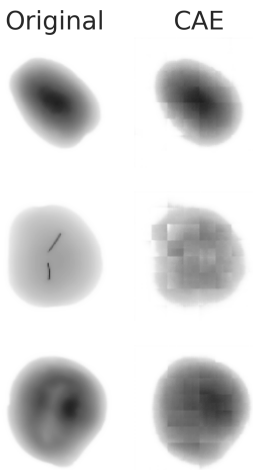


Figure 6.13: Selected reconstructions of potatoes of the CAE model given $J = 128$. From upper to lower row: reconstructions of perfect potato, potato with needle, and potato with hollow heart, respectively.

As can be seen from Table 6.6 and Fig. 6.14, in the potato dataset, when the anomaly ratio is between 30% and 70% the F_1-scores are all greater than 0.8 indicating the CAE and CVAE models have a good performance. When the anomaly ratio is grater than 70%, the F_1-scores of the networks decreases as the anomaly ratio increases, and on the other side; when the anomaly ratio is less than 30%, the F_1-scores decreases as the anomaly ratio decreases. The F_1-scores have a similar trend for the chocolate bar dataset, although their scores are better, indicating great performances.

These patterns are mainly due to when the anomaly ratio is small, the precisions are small, and as the anomaly ratio increases, the precision increases. However, this phenomenon is not so uncommon at all. It follows that as the anomaly score decreases the precision decrease, since there simply will be more FPs for every TP, because one is hunting for a "needle in a haystack",

Table 6.6: Based upon the results found in Table 6.5 the anomaly ratio is varied for (A) CVAE given $J = 8$ and CAE given $J = 64$ for the chocolate bar dataset; and (B) CVAE given $J = 32$ and CAE given $J = 128$ for the potato dataset. Mean values are reported, and uncertainties are calculated as the standard error on the mean.

Anomaly Ratio [%]	AUC-ROC	Specificity	Recall	Precision	F_1	Accuracy
CVAE						
Chocolate Bar						
5	**0.996 ± 0.001**	0.977 ± 0.004	**0.989 ± 0.005**	0.717 ± 0.033	0.906 ± 0.012	0.978 ± 0.004
10	0.995 ± 0.001	0.978 ± 0.003	0.979 ± 0.005	0.834 ± 0.018	0.945 ± 0.006	**0.979 ± 0.003**
30	0.995 ± 0.000	0.970 ± 0.002	0.978 ± 0.002	0.934 ± 0.004	0.968 ± 0.002	0.973 ± 0.002
50	0.996 ± 0.000	0.972 ± 0.002	0.976 ± 0.002	0.973 ± 0.002	**0.974 ± 0.001**	0.974 ± 0.001
70	0.995 ± 0.000	0.972 ± 0.005	0.975 ± 0.003	0.988 ± 0.002	0.969 ± 0.001	0.974 ± 0.001
90	0.996 ± 0.001	0.979 ± 0.006	0.971 ± 0.004	0.998 ± 0.001	0.931 ± 0.007	0.972 ± 0.003
95	0.995 ± 0.001	**0.985 ± 0.006**	0.973 ± 0.004	**0.999 ± 0.000**	0.894 ± 0.012	0.974 ± 0.003
Potato						
5	0.906 ± 0.007	0.832 ± 0.018	**0.880 ± 0.018**	0.234 ± 0.006	0.634 ± 0.015	0.835 ± 0.017
10	0.883 ± 0.006	0.828 ± 0.015	0.816 ± 0.017	0.361 ± 0.021	0.694 ± 0.012	0.827 ± 0.012
30	0.898 ± 0.003	0.859 ± 0.007	0.794 ± 0.010	0.709 ± 0.009	0.815 ± 0.004	**0.839 ± 0.004**
50	0.903 ± 0.002	0.846 ± 0.009	0.814 ± 0.011	0.843 ± 0.006	**0.830 ± 0.002**	0.830 ± 0.002
70	0.905 ± 0.002	0.852 ± 0.010	0.809 ± 0.010	0.928 ± 0.004	0.803 ± 0.004	0.822 ± 0.005
90	**0.908 ± 0.003**	**0.861 ± 0.016**	0.822 ± 0.013	0.982 ± 0.002	0.698 ± 0.009	0.826 ± 0.010
95	0.887 ± 0.007	0.840 ± 0.015	0.813 ± 0.013	**0.990 ± 0.000**	0.607 ± 0.013	0.815 ± 0.013
CAE						
Chocolate Bar						
5	0.990 ± 0.002	0.968 ± 0.005	**0.977 ± 0.009**	0.645 ± 0.032	0.876 ± 0.013	**0.969 ± 0.005**
10	0.989 ± 0.001	0.960 ± 0.005	0.963 ± 0.007	0.740 ± 0.021	0.906 ± 0.008	0.961 ± 0.004
30	0.990 ± 0.000	0.958 ± 0.004	0.962 ± 0.004	0.908 ± 0.007	0.952 ± 0.003	0.959 ± 0.003
50	0.990 ± 0.000	0.962 ± 0.001	0.955 ± 0.002	0.961 ± 0.001	**0.958 ± 0.001**	0.958 ± 0.001
70	0.991 ± 0.001	0.967 ± 0.002	0.951 ± 0.001	0.985 ± 0.001	0.949 ± 0.001	0.956 ± 0.001
90	**0.991 ± 0.001**	0.967 ± 0.006	0.961 ± 0.004	0.996 ± 0.001	0.908 ± 0.007	0.962 ± 0.003
95	0.990 ± 0.002	**0.971 ± 0.009**	0.962 ± 0.006	**0.998 ± 0.000**	0.860 ± 0.016	0.962 ± 0.005
Potato						
5	**0.908 ± 0.006**	0.829 ± 0.016	**0.900 ± 0.015**	0.230 ± 0.015	0.633 ± 0.014	0.833 ± 0.015
10	0.894 ± 0.006	0.822 ± 0.008	0.872 ± 0.015	0.356 ± 0.008	0.700 ± 0.006	0.827 ± 0.007
30	0.897 ± 0.003	0.827 ± 0.005	0.857 ± 0.007	0.681 ± 0.006	0.817 ± 0.003	0.836 ± 0.003
50	0.894 ± 0.001	0.826 ± 0.004	0.838 ± 0.004	0.828 ± 0.003	**0.832 ± 0.002**	0.832 ± 0.002
70	0.898 ± 0.002	0.826 ± 0.004	0.850 ± 0.005	0.919 ± 0.001	0.821 ± 0.002	0.842 ± 0.002
90	0.898 ± 0.007	0.834 ± 0.012	0.856 ± 0.005	0.979 ± 0.001	0.724 ± 0.005	**0.854 ± 0.004**
95	0.889 ± 0.007	**0.856 ± 0.017**	0.838 ± 0.014	**0.991 ± 0.001**	0.632 ± 0.012	0.839 ± 0.013

thus being more likely to mistake other things as the "needle" — namely the abnormal class.

The precision is the probability that instances being predicted as abnormal truly are an anomalous sample. For example, given a precision of 72%, then among those samples that are being predicted positive, i.e. as abnormal, the model are falsely predicting a normal sample as being anomalous 28% of the time.

Hence, given a low anomaly ratio the networks thus have some difficulties with correctly detecting normal samples, i.e. a large FP rate, but not many anomalous samples are being misclassified as being normal, i.e. few FNs. On the other hand, if the anomaly ratio is large, the models are prone to return a large FN rate, i.e. many anomalous samples are misclassified as being normal samples — which is typically the worst-case scenario. Accordingly, not a lot of normal samples are being misclassified as abnormal since they are so rare in this reverse case, i.e. a low rate of FPs.

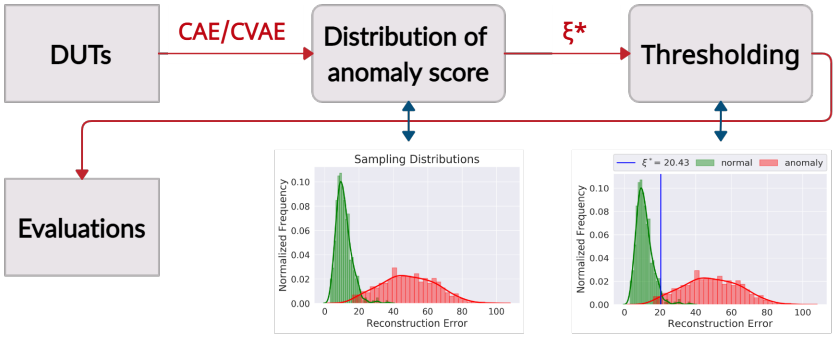


Figure 6.15: Generalized pipeline of the static novelty detection scheme as it has been compiled in this thesis. Flowchart created with `Creately`.

6.4 REFLECTIONS UPON RESULTS

6.4.1 *Static Novelty Detection: The Pipeline*

Let us take a step back and consider how the systematic framework of this static novelty detection scheme has been proceeding. The whole pipeline can be generalized, summarized and visualized as in Fig. 6.15. Essentially it is a rather simple set-up process, and the essential task lies in the choices of the generative model, the anomaly score and the technique of (automated) threshold selection. These selections can be replaced accordingly to the data considered and for the problem at hand and what it seeks to solve. Moreover, the largest challenge is training a generative model able to generate images similar to the training inputs to such a precision that small foreign objects can be detected in abnormal samples.

In this thesis two different generative models have been considered; the CAE and CVAE. Accordingly to the evaluations throughout § 6.3 the CVAE has consistently showed to perform the best out of the two models. However, why limit ourself to these two neural networks? Simply in the family of autoencoders other variations exists such as the *β-CVAE* [34], whereas an additional hyperparameter, β, is added to the loss function and works as an constraint on the bottleneck, potentially outperforming the CVAE. Alternative generative models that have not really been considered in this thesis is the family of generative adversarial networks (GANs) [28]. For instance, [16] and [64] utilizes similar detection pipelines, but instead of using the CAE or CVAE, they make use of the so-called *ADGAN* and *AnoGAN*, respectively. They both obtains promising performances on image benchmark datasets, hence playing a hand in proposing anomaly detections based on the family of GANs. GANs are however harder to train since the *generator* network that tries to fool the *discriminator* network plays a min-max game in such a way that the cost function of the model may not converge using gradient descent.

Additionally, the choice of the anomaly score can be attuned. In the above performances and empirical results in § 6.3, the reconstruction error — cal-

culated as the Euclidean distance and given by eq. (6.3) — was chosen as the anomaly score. The particular choice of metric function for the anomaly score is however pretty extensive, as the user themself has the freedom to pick and utilize any kind of useful similarity measure. In particular, a couple of different anomaly scores are revisited in § 6.4.2 below.

Finally, it is appropriate to also consider ways to choose and tuning the classification threshold. The choice of the G-MEAN has empirically shown to work well in the above result sections for for various anomaly ratios. Here, the true negative rate and true positive rate were considered equally important and that is a fact worth noticing. E.g. alternatively of computing the G-MEAN, the F_1-score (or accuracy score if non-skewed classes) could instead be computed at each threshold value, subsequently choosing an optimal cut-off by maximizing this metric.

For instance, if the positive class, i.e. the abnormal samples, were given a higher priority of discovering — upon the cost of predicting an increasing number of false positives — another threshold metric should be considered. Due to the way the problem is defined, on the ROC curve this scenario would corresponds to a rightward movement of the threshold value. In this case the recall/sensitivity (i.e. the TPR) should be maximized as one would want to find all abnormal samples, but it will be at the expense of the specificity (aka the TNR) falling accordingly. Vice versa if the negative class, i.e. the normal class, was deemed most important.

Nevertheless, when the anomaly score is computed the samples are rendered to histograms, i.e. probability distributions. And probabilities we understand though statistic. Hence, practices from advanced statistic could also be considered for detecting outliers, ultimately finding some tolerance limit in order separate the two class distributions. For instance, if the two class distributions are reasonable separated, the so-called *part average testing* [14] could be exploited.

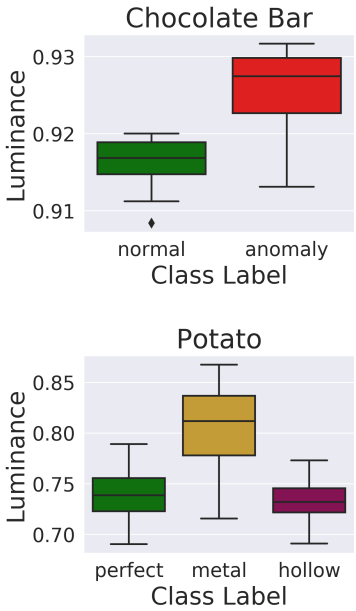


Figure 6.16: Image luminances as computed by eq. (6.15) over original preprocessed images. Due to the white background area in the chocolate bar dataset, their average pixel values are automatic increased.

6.4.2 *Anomaly Scores Revisited*

Recall the (noticeable) intensity differences between the normal and abnormal images showcased in Fig. 6.6. To quantify, the luminance calculated by eq. (6.15) can loosely be though of as the image brightness, as it simply sums all pixels values and takes the average, i.e. the overall mean intensity value of an image. On Fig. 6.16 the luminance of each preprocessed original image (i.e. excluded augmented images) have been computed for both the chocolate bar and potato datasets, respectively.

As discussed in section § 1.4, due to the large atomic numbers of foreign objects, when they are present in the X-ray imaging they absorbs almost all the photons, hence increasing the overall luminance of the resulting image.

Hence, the noticeable luminance difference of anomalous samples in the chocolate bar dataset and in general when needles are present in the potato dataset. On the other hand, the mean image luminance *decreases* slightly for potatoes with hollow hearts, but overall is similarly to the image luminance for perfect potatoes.

As will be discussed below, the (squared) Euclidean distance, used as the anomaly score and given by eq. (6.3), is sensitive to illuminations. Hence, it motivates us to explore other potential metrics that can be utilized as the anomaly score, and in the end determining if such non-invariance to changes in intensity are either beneficial or disadvantageous given a case study.

There exists a certain degree of flexibility in selecting a metric by which to evaluate the candidate models. The essential part lies in choosing some kind of appropriate distance or similarity measure between the original and reconstructed image. In this section a few image matching techniques are explored and later on applied to case studies.

IMage Euclidean Distance (IMED)

Due to the simplicity of the Euclidean distance it is one of the most commonly used distance measurements in the field of computer vision. Contemplate therefore again the squared Euclidean distance given by eq. (6.3). Formally, accordingly to the gray levels of each pixel an image with fixed width W and height H can be transformed from its 2D matrix form to the 1D vector $\mathbf{x} = \{x^{(1)}, x^{(2)}, ..., x^{(HW)}\}$. Given the vectorized original and reconstructed images, $\mathbf{x} \in \mathbb{R}^{H \times W}$ and $\tilde{\mathbf{x}} \in \mathbb{R}^{H \times W}$ respectively, the SE can also be written as [51, 73]

$$d^2_{\text{ED}} = \sum_{k=1}^{HW} \left(x^{(k)} - \tilde{x}^{(k)}\right)^2 = (\mathbf{x} - \tilde{\mathbf{x}})^T (\mathbf{x} - \tilde{\mathbf{x}}). \tag{6.11}$$

I.e., the Euclidean distance $d_{\text{ED}} \in [0, \sqrt{HW} \cdot 255]$ is only a summation of the pixel-wise intensity differences between two 8-bit gray scale images. I.e. this metric is sensitive and non-invariant to constant changes in brightness, such as noise. Furthermore, the squared Euclidean distance cannot reflect the real spatial distances as it only takes gray levels on pixels into account. Thus, it discards the image structures and cannot properly represent real distances between two images.

On the other hand, the *IMage Euclidean Distance (IMED)* overcomes this problem by exploring the spatial relationship between pixels. The IMED between two images is defined by [73]

$$d^2_{\text{IMED}}(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{i=1}^{HW} \sum_{j=1}^{HW} g_{ij} \left(x^{(i)} - \tilde{x}^{(i)}\right)\left(x^{(j)} - \tilde{x}^{(j)}\right) = (\mathbf{x} - \tilde{\mathbf{x}})^T \mathbf{G} (\mathbf{x} - \tilde{\mathbf{x}}), \tag{6.12}$$

IMAGE EUCLIDEAN DISTANCE

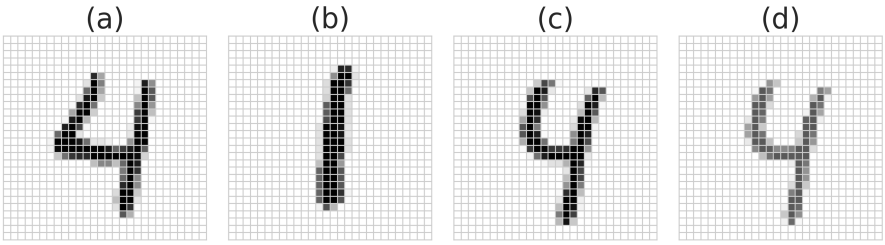


Figure 6.17: Similar and dissimilar 28×28 8-bit gray scale images. Here image **d** has the same structure as that of image **c**, but its brightness has been manipulated by adding a constant $c = 70$; making the image brighter. Data from the *MNIST handwritten digits*.

introducing the symmetric and positive-definite matrix[1] $\mathbf{G} = (g_{ij})_{HW \times HW}$, called the *metric matrix*. Here, g_{ij} is the metric coefficients which indicates the spatial relationship between pixels p_j and p_j whose entries are given by the formula

$$g_{ij} = f\left(d_{ij}^s\right) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(d_{ij}^s)^2}{2\sigma^2}\right), \tag{6.13}$$

where $\sigma = 1$ is the width parameter, and $d_{ij}^s = |p_i - p_j|$ the spatial distance between pixels p_j and p_j for $i, j = 1, 2, ..., HW$. E.g. if p_i and p_j are located at locations (k, l) and (k', l') on the image lattice respectively, then the distance between them is

$$d_{ij}^s = \sqrt{(k - k')^2 + (l - l')^2}. \tag{6.14}$$

Notice, eq. (6.13) is identified as the Gaussian function; thus f is a continuous function, and as $|p_i - p_j|$ increases, g_{ij} decreases monotonically [73]. Hence, eq. (6.12) determines the relationship between pixels on the images by using the prior knowledge that pixels located near one another have little variance in gray levels [51]. Thus, small spatial deformations yields small image distances, while larger deformations results in larger distances.

Hence, compared to eq. (6.11), that is highly sensitive to small spatial deformations, the IMED is relatively insensitive to such small perturbations. As an illustration, consider the example images shown in Fig. 6.17. Image $\mathbf{b} \in \mathbb{R}^{28\times28}$ is different from image $\mathbf{a} \in \mathbb{R}^{28\times28}$, while image $\mathbf{c} \in \mathbb{R}^{28\times28}$ is slightly deformed compared to **a**. Image $\mathbf{d} \in \mathbb{R}^{28\times28}$ is identical to **c** expects that its brightness has been manipulated. Computing the Euclidean distances yields $d_{\text{ED}}(\mathbf{a}, \mathbf{b}) = 117.85$ and $d_{\text{ED}}(\mathbf{a}, \mathbf{c}) = 123.74$. But, the pair with less similarity has a smaller Euclidean distance(!) On the other hand, computing the IMED gives $d_{\text{IMED}}(\mathbf{a}, \mathbf{b}) = 1707.42$ and $d_{\text{IMED}}(\mathbf{a}, \mathbf{c}) = 1308.18$. Hence, IMED provides intuitively better results, while the Euclidean distance yields counter intuitively results. Moreover, $d_{\text{ED}}(\mathbf{c}, \mathbf{d}) = 126.62$ and $d_{\text{IMED}}(\mathbf{c}, \mathbf{d}) = 1366.77$, implying their sensitivity to changes in illumination. Do notice, given the image dimensions 28×28 results in the metric matrix $\mathbf{G} \in \mathbb{R}^{784\times784}$, i.e. an already rather large matrix. Computing the IMED is however computational feasible for these small images, but it might pose some problems for larger images, such as large memory consumptions and execution times.

1. A symmetric matrix $\mathbf{M} \in \mathbb{R}^{N\times N}$ is said to be *positive-definite* if $\mathbf{x}^T\mathbf{M}\mathbf{x} > 0$ for all non-zero $\mathbf{x}$ in $\mathbb{R}^N$.

Structural Similarity Index Measure (SSIM)

Contrary to distance measurements, different similarity measures could also be investigated. The *structural similarity index measure (SSIM)* is such a metric and it extract three key features from an image; the *luminance*, *contrast* and *structure*, hence the comparison between two images are performed on the basis of these features [74].

The luminance of an image $\mathbf{x} \in \mathbb{R}^{H \times W}$ is measured by the average over all pixel values, i.e.

$$\mu_{\mathbf{x}} = \frac{1}{HW} \sum_{k=1}^{HW} x^{(k)}. \tag{6.15}$$

The luminance comparison function is then a function between the means $\mu_{\mathbf{x}}$ and $\mu_{\tilde{\mathbf{x}}}$, that is

$$l(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{2\mu_{\mathbf{x}}\mu_{\tilde{\mathbf{x}}} + C_1}{\mu_{\mathbf{x}}^2 + \mu_{\tilde{\mathbf{x}}}^2 + C_1}, \qquad C_1 = (K_1 L)^2, \tag{6.16}$$

given another image $\tilde{\mathbf{x}} \in \mathbb{R}^{H \times W}$. Here, C_1 is a stabilization constant and $K_1 \ll 1$ another small constant. L is the dynamic range for pixel values, which for 8-bit gray scale images is 256.

The contrast of image $\mathbf{x}$ is measured by taking the standard deviation of all the pixel values, i.e. $\sigma_{\mathbf{x}} = \left(\frac{1}{HW-1} \sum_{k=1}^{HW} \left(x^{(k)} - \mu_{\mathbf{x}} \right)^2 \right)^{1/2}$. Accordingly, the contrast comparison function is then

$$c(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{2\sigma_{\mathbf{x}}\sigma_{\tilde{\mathbf{x}}} + C_2}{\sigma_{\mathbf{x}}^2 + \sigma_{\tilde{\mathbf{x}}}^2 + C_2}, \qquad C_2 = (K_2 L)^2, \quad \text{where } K_2 \ll 1. \tag{6.17}$$

By removing the mean intensity from image $\mathbf{x}$ and dividing it with its own standard deviation results in the output being normalized and having unit standard deviation, that is $(\mathbf{x} - \mu_{\mathbf{x}})/\sigma_{\mathbf{x}}$. This allows for a more robust comparison, and the structure comparison function is thus

$$s(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{2\sigma_{\mathbf{x}\tilde{\mathbf{x}}} + C_3}{\sigma_{\mathbf{x}}\sigma_{\tilde{\mathbf{x}}} + C_3}, \tag{6.18}$$

where $\sigma_{\mathbf{x}\tilde{\mathbf{x}}}$ is the covariance which is estimated by $\sigma_{\mathbf{x}\tilde{\mathbf{x}}} = \left(\frac{1}{HW-1} \sum_{k=1}^{HW} \left(x^{(k)} - \mu_{\mathbf{x}} \right) \left(\tilde{x}^{(k)} - \mu_{\tilde{\mathbf{x}}} \right) \right)$.

The SSIM score is then a weighted combination of the luminance, contrast and structure between the two images and given by the combination function $\mathrm{SSIM}(\mathbf{x}, \tilde{\mathbf{x}}) = [l(\mathbf{x}, \tilde{\mathbf{x}})]^{\alpha} \cdot [c(\mathbf{x}, \tilde{\mathbf{x}})]^{\beta} \cdot [s(\mathbf{x}, \tilde{\mathbf{x}})]^{\gamma}$, where $\alpha, \beta, \gamma > 0$ denotes the

relative importance of each of the metric components. Assuming $\alpha, \beta, \gamma = 1$ and $C_3 = C_2/2$, the SSIM can then finally be described by

STRUCTURAL SIMILARITY INDEX MEASURE

$$\mathrm{SSIM}(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{(2\mu_{\mathbf{x}}\mu_{\tilde{\mathbf{x}}} + C_1)(2\sigma_{\mathbf{x}\tilde{\mathbf{x}}} + C_2)}{(\mu_{\mathbf{x}}^2 + \mu_{\tilde{\mathbf{x}}}^2 + C_1)(\sigma_{\mathbf{x}}^2 + \sigma_{\tilde{\mathbf{x}}}^2 + C_2)}. \tag{6.19}$$

Output values are $\mathrm{SSIM} \in [-1, 1]$, where 1 means a perfect similarity, 0 no similarity and -1 a maximum negative similarity.

The three components are relatively independent from each other. Consider the examples on Fig. 6.17. Computing the SSIM yields $\mathrm{SSIM}(\mathbf{a}, \mathbf{b}) = 0.12$, $\mathrm{SSIM}(\mathbf{a}, \mathbf{c}) = 0.68$ and $\mathrm{SSIM}(\mathbf{a}, \mathbf{d}) = 0.61$, which all are satisfactory results. Moreover $\mathrm{SSIM}(\mathbf{c}, \mathbf{d}) = 0.92$, i.e. a constant change in illumination but keeping the same structure still yields a large, but not perfect, similarity as expected.

Notice, it is expected that anomalous samples have an error lager than that of normal samples, but the SSIM is a similarity measure. Hence, for it to work as an anomaly score in the scheme of the novelty detection, then without loss of generality $1 - \max\big(0, \mathrm{SSIM}(\mathbf{x}, \tilde{\mathbf{x}})\big)$ should be utilized instead.

Zero-Mean Normalized Cross-Correlation (ZNCC)

Another similarity measure worth contemplating is the *zero-mean normalized cross-correlation (ZNCC)*, also known as *Pearson correlation coefficient*. In computer vision, this measure is widely used in template or image matching, whereas its purpose typically is to locate specific objects in an image by applying a template, or to compare two images.

As usual, considering an original and reconstructed image in their vectorized forms, the Pearson correlation coefficient is described by [78]

ZERO-MEAN NORMALIZED CROSS-CORRELATION

$$\mathrm{ZNCC}(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{\sum_{k=1}^{HW} \big(x^{(k)} - \bar{x}\big)\big(\tilde{x}^{(k)} - \bar{\tilde{x}}\big)}{\sqrt{\sum_{k=1}^{HW} \big(x^{(k)} - \bar{x}\big)^2}\sqrt{\sum_{k=1}^{HW} \big(\tilde{x}^{(k)} - \bar{\tilde{x}}\big)^2}}, \tag{6.20}$$

where $x^{(k)}$ is the k^{th} pixel in $\mathbf{x}$, and $\bar{x} = \frac{1}{HW}\sum_{k=1}^{HW} x^{(k)}$ the sample mean of $\mathbf{x}$; and analogously for $\tilde{\mathbf{x}}$. Its output domain is $\mathrm{ZNCC} \in [-1, 1]$, where 1 indicates a maximum similarity match, 0 for no similarity, and -1 for a complete negative correlation. Contrary to the Euclidean distance, the ZNCC is expectedly invariant to constant changes in brightness as the output only depends on how similar the reconstructed image are to the original image.

As an example, consider again the images on Fig. 6.17. Computing eq. (6.20) yields $\mathrm{ZNCC}(\mathbf{a}, \mathbf{b}) = 0.04$, and $\mathrm{ZNCC}(\mathbf{a}, \mathbf{c}) = 0.72$. These are quiet satisfactory results as far as the structural differences goes. Furthermore, computing

the similarity between the same image system but with different illumination gives $\mathrm{ZNCC}(\mathbf{c}, \mathbf{d}) = 0.99$, i.e. the ZNCC is indeed (almost) invariant to linear brightness and contrast variations, and the very small difference is as anticipated [78].

Notice, since the ZNCC is a similarity measure, it is necessary to subtract the metric by 1 such that it can be utilized as the anomaly score in an appropriate way, i.e. $1 - \max\left(0, \mathrm{ZNCC}(\mathbf{x}, \tilde{\mathbf{x}})\right)$, as it furthermore does not make sense to consider negative correlations in its application.

Gradient Magnitude Similarity Deviation (GMSD)

On the flip side, how about considering edges and gradients? Due to foreign objects and other defects present in abnormal samples, such as hollow hearts and cracks, the anomalous samples are more likely to posses an additional amount of edges compared to that of normal samples. Since the CAE and CVAE models are (successfully) trained to reconstruct only normal samples they fail to reconstruct the defects of anomalous samples. Hence, one could begin to wonder if such loss of edges could be utilized to make use of an anomaly score whose metric are based on gradients. Luckily, such a metric that compares the gradient magnitude similarity between the reference and distorted images to compute a quality score already exists, and is known as the *gradient magnitude similarity deviation (GMSD)* [77].

In order to detect edges in images, an image is convolved with a linear filter, whereas the convolution operation is defined by eq. (5.11) in section § 5. In [77] the authors have chosen to divide the well-known 3×3 *Prewitt filters* along the horizontal and vertical *directions* by 3, such that

$$\mathbf{h}_x = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}; \qquad \mathbf{h}_y = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \tag{6.21}$$

where $\mathbf{h}_x$ denotes the horizontal derivative, and $\mathbf{h}_y$ the vertical derivative. Convolving an image $\mathbf{x}$ with such linear filter yields the directional gradient of the image, i.e. the image derivatives $\partial/\partial x = \mathbf{x} * \mathbf{h}_x$ and $\partial/\partial y = \mathbf{x} * \mathbf{h}_y$.

Besides a direction the gradient also has a *magnitude*, which is simply the length of the gradient vector calculated as $|\mathbf{H}(\mathbf{h}_x, \mathbf{h}_y)| = \sqrt{\mathbf{h}_x^2 + \mathbf{h}_y^2}$. At pixel location i, the gradient magnitudes of the original image $\mathbf{x} \in \mathbb{R}^{H \times W}$ and the reconstructed image $\tilde{\mathbf{x}} \in \mathbb{R}^{H \times W}$ are then

$$\begin{aligned} \mathbf{m}_{\mathbf{x}}(i) &= \sqrt{\left(\mathbf{x} * \mathbf{h}_x\right)^2(i) + \left(\mathbf{x} * \mathbf{h}_y\right)^2(i)}, \\ \mathbf{m}_{\tilde{\mathbf{x}}}(i) &= \sqrt{\left(\tilde{\mathbf{x}} * \mathbf{h}_x\right)^2(i) + \left(\tilde{\mathbf{x}} * \mathbf{h}_y\right)^2(i)}. \end{aligned} \tag{6.22}$$

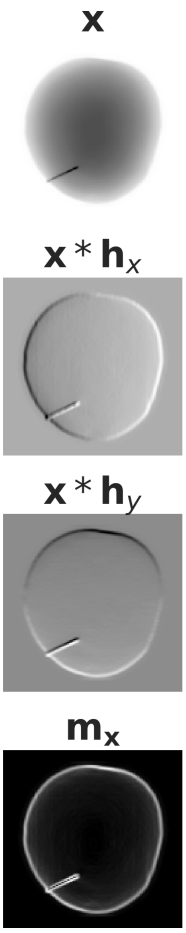


Figure 6.18: Prewitt filters decribed by eq. (6.21) applied to a preprocessed X-ray image of a potato with an inserted needle. The horizontal Prewitt enhances horizontal edges while the vertical Prewitt enhances vertical edges. The combined Prewitt, i.e. the gradient magnitude, creates very nice resulting edges.

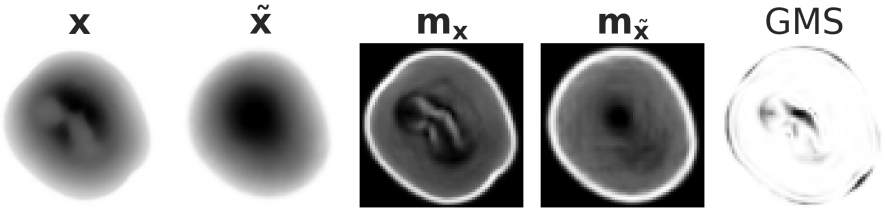


Figure 6.19: First: An input image of a preprocessed X-ray image of a potato with hollow heart. Second: The reconstructed image from the CVAE model given $J = 32$. Third and Fourth: The corresponding gradient magnitude images as computed by eq. (6.22). Fifth: The associated gradient magnitude similarity map, i.e. quality map, computed by eq. (6.23). The brighter the gray levels, the higher the local similarity between the two images.

To illustrate consider Fig. 6.18 where the Prewitt filters described by eq. (6.21) have been convolved with an input image of a preprocessed potato with an inserted needle. Each derivatives finds edges that the other does not, and by computing the gradient magnitude as described by eq. (6.22) it creates a final edge enhanced image.

Following, the *gradient magnitude similarity (GMS)* is then computed as

$$\text{GMS}(i) = \frac{2\mathbf{m}_{\mathbf{x}}(i)\mathbf{m}_{\tilde{\mathbf{x}}}(i) + c}{\mathbf{m}_{\mathbf{x}}^2(i) + \mathbf{m}_{\tilde{\mathbf{x}}}^2(i) + c}, \tag{6.23}$$

where the parameter $c > 0$ is a numerical constant ensuring numerical stability. If 8-bit images are in the domain of $[0, ..., 255]$, then $c = 170$ can be assumed as a default value, however this parameter could be exploited further [77]. The GMS serves as the *local quality map (LQM* of the reconstructed image, and reflects the local quality of each small patch in the reconstructed image. Hence, if $\mathbf{m}_{\mathbf{x}}(i)$ and $\mathbf{m}_{\tilde{\mathbf{x}}}(i)$ are the same, then $\text{GMS}(i) = 1$, where 1 is the maximal value.

As an example, consider Fig. 6.19. The first two images shows an preprocessed image of a potato with hollow heart disease and the CVAE's reconstruction of it, respectively. Their respective gradient magnitude images are computed and shown in the next two images. Finally, the corresponding GMS map as computed by eq. (6.23) is shown. The brighter the gray level, the larger the pixel values in $\text{GMS}(i) \in [0, 1]$, and thus the higher the predicted similarity.

The overall quality score can then be estimated from the LQM via a pooling stage, such as *average pooling*. I.e. by averaging the LQM values and taking the mean yields the *gradient magnitude similarity mean (GMSM)*

$$\text{GMSM} = \frac{1}{HW}\sum_{i=1}^{HW} \text{GMS}(i), \tag{6.24}$$

where $H \cdot W$ is the total number of pixels in the image. The higher the GMSM score, the higher the image similarity. Finally, the authors in [77] proposes the GMSD as the final score

$$\text{GMSD} = \sqrt{\frac{1}{HW}\sum_{i=1}^{HW}\big(\text{GMS}(i) - \text{GMSM}\big)^2}. \tag{6.25}$$

GRADIENT MAGNITUDE SIMILARITY DEVIATION

The higher the GMSD score, the lower the image similarity. I.e. translated to the set-up of the novelty detection, a larger GMSD score means a larger reconstruction error. Moreover, it is important to note that gradients are highly sensitive to noise. Thus it is essential to remove noise in the image preprocessing stages.

As a final example consider our favourite handwritten digits in Fig. 6.17. Computing the GMSD score yields $\text{GMSD}(\mathbf{a}, \mathbf{b}) = 0.40$ and $\text{GMSD}(\mathbf{a}, \mathbf{c}) = 0.26$, i.e. image $\mathbf{c}$ is indeed more similar to image $\mathbf{a}$ compared to image $\mathbf{b}$. Furthermore, it shows that the constant brightness difference does not have such a large impact as $\text{GMSD}(\mathbf{a}, \mathbf{d}) = 0.27$ and $\text{GMSD}(\mathbf{c}, \mathbf{d}) = 0.08$. Moreover, as a sanity check $\text{GMSD}(\mathbf{a}, \mathbf{a}) = 0.00$ as anticipated since the images are identical.

6.4.3 Case Studies Reexamined

In this section each of the various anomaly scores just introduced in section § 6.4.2 above are applied to the two prior case studies, followed by a comparison. Furthermore, two new case studies are investigated in order to further test the limits of the CVAE network and the anomaly scores.

The Chocolate Bar Dataset: Matching Intensities

Imaging if one had to deal with a dataset whose normal and abnormal samples had more or less the same luminance. As an experimental set-up one could thus try and even out the playing field between the normal and anomalous samples in the chocolate bar dataset by matching their histograms. To do so, the general midway equalization and histogram matching, whose techniques are described in sections § 5.1.3 and § 5.1.2 respectively, could be utilized.

Prior to the data preprocessing outlined in section § 5.2.3 for the chocolate bars, midway equalization using the CDFs of all normal and anomalous samples are applied. This application results in the luminance distributions shown on the upper plot in Fig. 6.20. As can be seen, the luminance stay roughly consistent, but there are still a difference in luminance between the two classes.

Attempting to match the intensities between the two classes even more, histogram matching is performed for all normal and abnormal samples, where the normal image corresponding to the median luminance is used

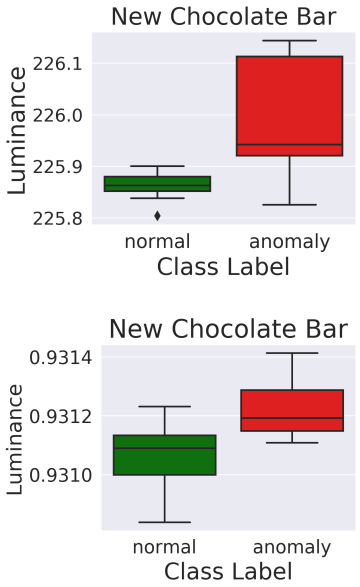


Figure 6.20: Image luminances as computed by eq. (6.15) over: (Upper) original chocolate bar images *before* preprocessing and *after* midway equalization and; (Lower) original chocolate bar images *after* preprecessing and histogram macthing.

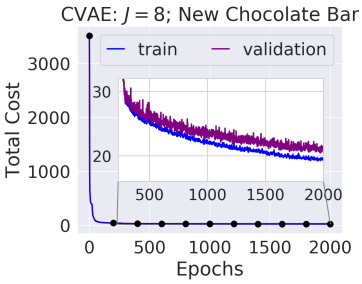


Figure 6.21: Average loss function as a function of number of epochs, derived on training images of the new chocolate bar dataset.

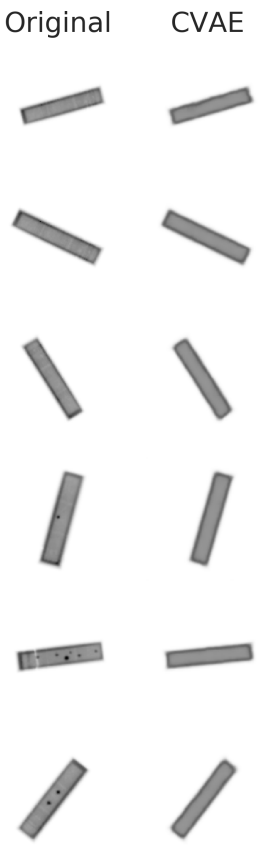


Figure 6.22: Selected reconstructions of test samples of the new chocolate bars by the CVAE model given $J = 8$.

as the reference. This results in the lower plot on Fig. 6.20. The histogram matching effectively regularizes the data as images gets shifted towards a common sample distribution, but still there is the smallest difference between the normal and abnormal classes. When these operations are performed a bias is however introduced at the same time.

Using this new histogram matched dataset it is trained by the CVAE model given $J = 8$. The same number of augmentations and split ratios are reused in this model. The average loss function as a function of number of epochs is seen on Fig. 6.21. Notice, compared to the cost curves of the prior chocolate bar, the cost at the final epoch is lower, indicating the model's ability to estimate reconstructions of training samples are 'better'. Furthermore, the loss have yet to converge and keeps decreasing slowly, suggesting the model could have been allowed to train for more epochs, but with minimal improvements.

Some selected reconstructions of test samples at the final epoch can be seen on Fig. 6.22. Compared to Fig. 6.6 notice how the luminances looks similar for both classes.

A New Potato Dataset

In the meantime, *Newtec Engineering A/S* has acquired new scan data of perfect potatoes and potatoes with natural hollow hearts (and not simple artificial created ones). They have been recorded by their in-house line scanner given tube current 5.5 mA and voltage 60 kV. Of distinct image samples, there are a total of 45 perfect potatoes and 66 hollow ones. The following preprocessing pipeline and image augmentations have been applied to this potato dataset:

i Piecewise linear contrast stretching, given by eq. (5.1) have been utilized given parameters $(r_1, s_1) = (12, 0)$ and $(r_2, s_2) = (140, 255)$.

ii Subsequently, application of gamma correction given $\gamma = 0.7$, cf. § 5.1.1.

iii The bilateral filter given neighborhood diameter $d = 5$, and parameters $\sigma_s = 15$ and $\sigma_r = 10$ have been applied, cf. § 5.1.4.

iv Contrary to the prior potato dataset, whose samples were elongated, these images of potatoes are also stretched, but using an interpolation method will not render the objects to be potato shaped. Nevertheless, the area interpolation showed in § 5.1.5 are utilized to resize images into the desired dimensions of $(128, 128)$.

v Applied min-max normalization and the normalization given by eq. (5.15) and eq. (5.16), respectively.

vi The augmentations applied on the other datasets, such as allowing rotations in full range, should not be applied to this dataset as it would not

reflect how "real" data looks like. Instead, it is needed to consider more complex types of augmentation. The following augmentations are utilized; distortions by shearing, elastic distortion, changing the perspective by zoom effect, and allowing potatoes to be rotated by 180 degrees.

Again computing the luminance of the preprocessed original images, it can be seen that there indeed are intensity differences between perfect and hollow potatoes, as shown in Fig. 6.23.

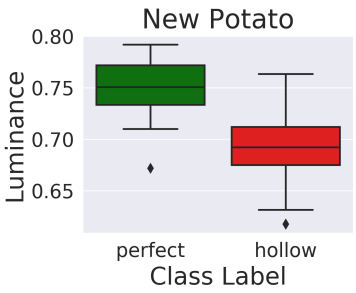


Figure 6.23: Image luminances as computed by eq. (6.15) over original preprocessed images of the new potato dataset.

Assuming that the performances of CVAE models are more or less robust to latent dimensions equal to or larger than 8, this dataset is trained using the CVAE architecture described by Tables 6.1-6.2 given $J = 32$. The model is trained using $\{\mathcal{N}_{\text{train}}^{\text{NewPo}}\} = 1683$ and validated by $\{\mathcal{N}_{\text{val}}^{\text{NewPo}}\} = 104$, and given an 50% anomaly ratio tested by $\{\mathcal{N}_{\text{test}}^{\text{NewPo}}\} = 350$ and $\{\mathcal{A}^{\text{NewPo}}\} = 350$. The cost function as a function of number of epoch are seen on Fig. 6.24. As can be seen, the training loss keeps decreasing while the validation loss starts to converge. This is an indication of the model beginning to overfit, as the model capture the training data well, but performs poorly on new data, thus not being able to generalize well. Several reasons could be the cause of this outcome, but expectedly it is due to the network architecture not being customized very well for this dataset, or that the augmented data are not sufficiently reflecting real data.

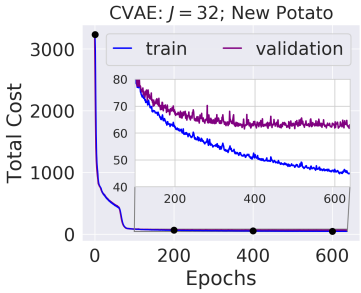


Figure 6.24: Average loss function as a function of number of epochs, derived on training and validation images of the new potato dataset.

On Fig. 6.25 a few selected test samples and their reconstructions are shown as derived at the final epoch. Notice how it is already pretty hard to distinguish perfect potatoes from hollow ones by the untrained human eye. Moreover, as reflected by Fig. 6.24 the CVAE model have a harder time training on this dataset compared to the other ones. The network is however pretty descent at reconstructing the outer shape of the potatoes, but still struggles with reconstructing the variations and structures of the inner potato masses.

Evaluations Based on Various Anomaly Scores

In the following all results are based on a 50% anomaly ratio in the test set and optimal thresholding automated by the G-MEAN, cf. eq. (6.4). Reusing

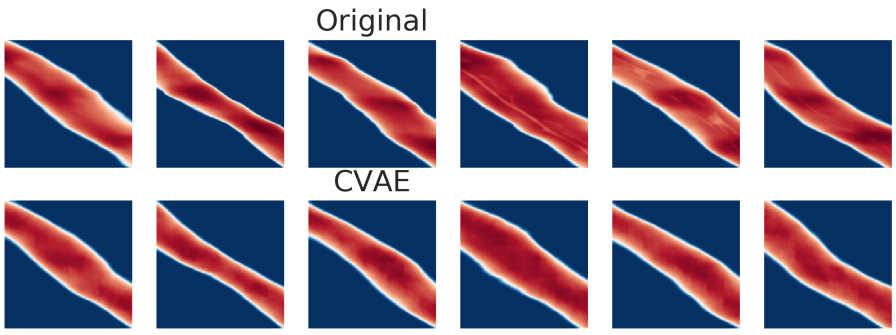


Figure 6.25: Selected test samples and their reconstructions by the CVAE network given $J = 32$ for the new potato dataset. First three columns are perfect potatoes and the last three columns are potatoes with hollow heart disease.

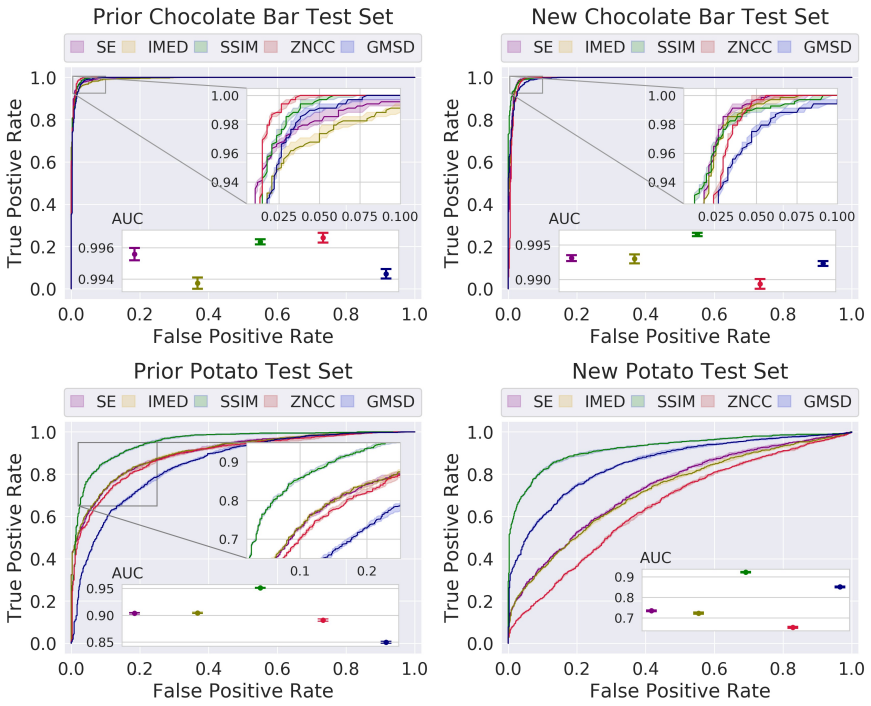


Figure 6.26: Resulting interquartile ROC curves and their mean AUC-ROC results. Error bars represents the standard error on the mean. Upper left: Prior chocolate bar dataset. Upper right: New chocolate bar dataset. Lower left: Prior potato dataset. Lower right: New potato dataset.

the former trained CVAE models given $J = 8$ for the prior chocolate bar dataset and $J = 32$ for the prior potato dataset, the models are evaluated by utilizing the various anomaly score described in section § 6.4.2 and by eq. (6.3). So is the newly trained chocolate bar dataset and second potato dataset described just above. Selected evaluating metric are summarized in Table 6.7. Furthermore, the interquartile ROC curves over all four models and each based on all five different anomaly scores can be found in Fig. 6.26.

Consider Fig. 6.27 where the in-class error rates and overall misclassifications are shown for the original chocolate and potato datasets as computed by the different anomaly score metrics. These two plots gives a clear overview of how the metrics reflects their abilities at distinguishing normal and anomalous samples.

By utilizing the SE, that relies on differences in pixel-wise intensities, potatoes with inserted needles whose overall image luminance are higher than images of perfect potatoes are easily detected, whereas images of hollow potatoes whose luminance are only slightly smaller than images for perfect potatoes are harder to detect. Though the IMED provides intuitively better results, its overall performance is in accordance with the SE. The difference lies in the IMED detecting a few more perfect potatoes as being normal, but at the cost of failing to detect a few more hollow potatoes as being abnormal samples. As for the chocolate bar dataset, the SE, that is sensitive to small spatial deformations, performs slightly better compared to the IMED, that is insensitive to small spatial deformations. This implies that since foreign objects are present at only small regions, the detection set-up works better

Figure 6.27: Mean class error rates as computed by the different anomaly score metrics. The former potato dataset trained by the CVAE model given $J = 32$ and former chocolate bar dataset given $J = 8$ and trained by the CVAE model, have been used with a 50% anomaly ratio. The total misclassification rate is simple the average of the in-class error rates. The error bars represent the standard error on the mean.

by putting a larger weights on small pixel differences compared to being insensitive to such small deformations.

The overall performance of the SSIM, whose score is a weighted combination of image luminance, contrast and structure, is the best out of all the anomaly scores for the potato dataset. In particular, it predicts perfect potatoes and hollow potatoes very well, however at the cost of wrongly predicting potatoes with inserted needles to a larger degree. The SSIM also performs well for the chocolate bar dataset where it correctly detects a larger amount of anomalous samples contrary to normal samples.

Against expectations, the ZNCC, that is (almost) invariant to constant changes in brightness and only depends on structural differences, is the worst out of all the metric at detecting hollow hearts. On the other hand, compared to the other anomaly scores, its ability to detect needles and perfect potatoes is adequate, however slightly poorer. This suggest that the network's ability to reconstruct perfect potatoes is not satisfactory enough to, structure-wise, provide a state-of-the-art performance, as the ZNCC has a hard time distinguishing between perfect and hollow heart potatoes. Again, surprisingly the ZNCC is the metric that performs best out of them all on the chocolate bar dataset. However, this is likely due to the fact that usually more than one foreign object is present in the chocolate bars, given rise to increased defective regions, thus larger structural differences.

The GMSD, that computes an overall quality score between gradient magnitude similarities, is the metric whose overall performance is the worst out of them all on the potato dataset. The complication lies in its ability to correctly predict perfect potatoes and potatoes with inserted needles in them. However, it is the second best metric at correctly predicting hollow potatoes(!) Recall the example potato in Fig. 6.19. Not only the hollow heart is being detected as being dissimilar, but the outer rim of the potatoes does to, i.e. based on this figure, it is obvious that the network does not reconstruct the inherent shapes of potatoes precisely. I.e. given reconstructions of perfect potatoes there are also a dissimilarity in primary their outer rims (and to smaller degree in their inner structures). Likewise, the small gradients of needles are not enough to

Table 6.7: Selected mean results evaluated on:(Prior) CVAE given $J = 8$ for former trained chocolate bar dataset; and CVAE given $J = 32$ for former trained potato dataset. (New) CVAE given $J = 8$ for the new chocolate bar dataset; and CVAE given $J = 32$ for the new potato dataset. Anomaly ratio sat to 50% in all studies and optimal threshold determined by the G-MEAN. Uncertainties are calculated as the standard error on the mean.

Anomaly Score	Prior			New		
	AUC-ROC	F_1	Accuracy	AUC-ROC	F_1	Accuracy
Chocolate Bar						
SE	0.996 ± 0.000	0.974 ± 0.001	0.974 ± 0.001	0.993 ± 0.000	**0.980 ± 0.001**	**0.980 ± 0.001**
IMED	0.994 ± 0.000	0.967 ± 0.001	0.967 ± 0.001	0.993 ± 0.001	0.977 ± 0.001	0.977 ± 0.001
SSIM	0.996 ± 0.000	0.980 ± 0.001	0.980 ± 0.001	**0.997 ± 0.000**	0.976 ± 0.001	0.976 ± 0.001
ZNCC	**0.997 ± 0.000**	**0.985 ± 0.001**	**0.985 ± 0.001**	0.989 ± 0.001	0.976 ± 0.001	0.976 ± 0.001
GMSD	0.994 ± 0.000	0.975 ± 0.001	0.975 ± 0.001	0.992 ± 0.000	0.967 ± 0.001	0.967 ± 0.001
Potato						
SE	0.903 ± 0.002	0.830 ± 0.002	0.830 ± 0.002	0.734 ± 0.002	0.680 ± 0.002	0.680 ± 0.002
IMED	0.904 ± 0.001	0.830 ± 0.002	0.830 ± 0.002	0.722 ± 0.003	0.672 ± 0.003	0.673 ± 0.003
SSIM	**0.951 ± 0.001**	**0.885 ± 0.002**	**0.885 ± 0.002**	**0.923 ± 0.001**	**0.865 ± 0.002**	**0.865 ± 0.002**
ZNCC	0.891 ± 0.002	0.820 ± 0.002	0.821 ± 0.002	0.653 ± 0.003	0.625 ± 0.003	0.626 ± 0.002
GMSD	0.849 ± 0.002	0.776 ± 0.002	0.776 ± 0.002	0.851 ± 0.002	0.779 ± 0.002	0.779 ± 0.002

significantly add to the quality score, hence failing to distinguish them from perfect potatoes.

Hence, it is expected that if a network could be trained to reconstruct the inherent structures of potatoes to an even larger precision, a larger number of perfect potatoes would be correctly predicted as their GMSD score would decrease, while the quality score of potatoes with needles and hollow hearts would increase. After all, the GMSD metric does perform reasonable well on the prior chocolate bar dataset whose reconstructions and inherent edges are more straightforward compared to the potatoes.

Against expectations the SE score yields better results for the new chocolate bar dataset compared to the prior one. A potential explanation for this observation lies in the altered intensity histograms that stays roughly consistent, which causes the model to be very sensitive to even small changes in luminance. The luminance of the two classes does not overlap completely as observed on Fig. 6.20, which might just be causing this issue. Therefore, this experiment have mainly been used to test the sensitivity of the CVAE model that utilizes the intensity based MSE metric as its reconstruction term during training. It has not been possible to completely overlap the distributions of luminance between the two classes, but only very closely. However, it is expected that if the distribution between the classes were to overlap each other, the performances based on the SE score would decrease as this per-pixel reconstruction error would likely fail to reveal defective regions to a larger extend.

Regarding the new potato dataset, the choice of anomaly score is especial crucial as each distinct score yields very dissimilar performances. The SSIM obviously performs the best, followed by the GMSD, while the ZNCC have the hardest time out of all metrics at correctly separating anomalous samples

from normal ones. Again, the SE performs slightly better than the IMED. Seemingly the weighted combination of luminance, contrast and structure are best at correctly predicting hollow hearts whenever it is the prior or new potato dataset.

6.4.4 Model Complexity and Hyperparameter Optimization

Hyperparameter optimization is non-trivial for neural networks or deep latent variable models. Below are some considerations about model complexity and hyperparameter tunings discussed.

The CVAE network architecture described in Tables 6.1-6.2 have carefully been chosen after repeated experimentations and comparisons of results by manual tuning of hyperparameters such as filter sizes, kernel sizes, choice of activation functions and sizes of the dense layers.

The complexity of the network depends partly on the input data dimension and on the number of hidden layers. A sparse network with too little capacity cannot learn the data representations well, whereas a deep network with too much capacity learns the training data well, but results in overfitting. In general it follows that the more number of parameters and layers, the larger the chance the deep network have of overfitting. In either case, it results in a model that does not generalize well. The final proposed architecture in Tables 6.1-6.2 have proved to work well on the first two case studies as it have showed to generalize well, but it is however prone to some overfitting typically given lower latent dimensions. Since the new potato dataset seemingly are more complex compared to the two other datasets, a deeper architecture of the CVAE could be constructed in order to potentially better extract the inherent features and capture more meaningful patterns that represents normal data.

Another issue is that of *cross-validation* during training of an unsupervised model. Cross-validation can be defined in supervised learning as data are labelled and for which performance of prediction are measured against ground truth (i.e. the labels), thus there are a clear definition of error. However, the objective in unsupervised learning are dissimilar as features are captured and passed to some feature space, i.e. there are no clear definition of error and cross-validation cannot be defined in the same way as in supervised learning. What can be done however is pre-splitting the data into training, validation and testing sets, subsequently checking the model performance by looking at the average loss of the validation set during training, whereat changes can be made, and the process repeated.

Alternatively, other optimizers than Adam could be tested, however since the results have shown to be justifiable, it has been deemed not to be necessity contrary to tuning other hyperparameters.

CONTENTS

WRAP-UP

7

7.1 OUTLOOK

The CAE and CVAE networks have successfully been trained using a given dataset. The decoders of the networks thus have the abilities to generate mock data instances corresponding to a given latent expression as based on the training data. When new inputs are fed into the network, the corresponding outputs are twofold; (a) if feeding the network with data input as that of the training dataset, the models will generate a similar output, because the networks are trained so. However, (b) if the input is dissimilar to that of the training data, the model output will likely be different from the input, because the output is generated from a generative model that is not the one that the decoder of the network approximates.

Hence, by having fed the networks with only the normal class, the generative models have captured a progressively rich representation of the data. Thereby comparing the input and output of the CAE and CVAE models, it have been possible to detect the inputs whose outputs is different from the training data; thus separating normal from anomalous instances.

7.1.1 Summary of Findings

Having worked with datasets from real food production lines, it can be testified that there are a lot of preprocessing that needs to be done — the reality is indeed pretty messy. Luckily, the labelling can be trusted as the annotating processes have been straightforward. However, due to the sparse data the trained models relies heavily on data augmentations as a compensation. Hence, there could likely be an over-representation of similar variations, thus the models might be overfitting to some degree.

The obtained results should thereby be taken with a grain of salt; for the model will only ever perform as well as the quantity and quality of the data given. I.e. the neural networks are only as good as the data they are fed. In particular, if such a novelty detection as examined in this thesis ever should be utilized in the industry, more data and variations of it are definitely required.

Mainly an anomaly ratio of 50% has been assumed throughout this thesis, however a few variations have been examined, namely $\{5, 10, 30, 70, 90, 95\}\%$. In reality, it is expected that anomalies found in device under tests (DUTs) are very rare, thus the ratio being on the lower end — and probably even

lower than the rates tested in the case studies. When the anomaly ratio is low, one should then be prepared for a decreased precision, as the models will be hunting for a "needle in a haystack", thus naturally misclassifying an increasing rate of normal samples as being anomalous.

Ultimately, a summarization of the key takeaways from the findings in this thesis are listed below:

- The ability to train a generative model able to learn and reconstruct commonalities and inherent characteristic of the specific DUTs, such as structures or densities, is crucial. However, both the CAE and CVAE are sensitive in the sense that inputs of deformed scan objects, the models have never seen before, have a high probability of getting detected as anomalous. Hence, it is essential that the generative model can reconstruct normal samples as precise as possible, or else there will be a higher certainty for normal samples being identified as anomalous and vice versa. Therefore, the CVAE in general produces better results compared to the CAE.
- It is worthwhile to experiment with different number of latent dimensions, as the results potentially can improve greatly with increasing dimensions.
- Since the generative models not only learn the underlying structures, but also individual pixel intensities, potential luminance differences between normal and anomalous images in the datasets should be considered and examined.
- The choice of anomaly score is essential. One should definitely examine different types and figure out which one could possible provide the best results on a given case study.
- The choice of optimal threshold technique based on which class is deemed most important, i.e. are false positive or false negatives equally costly or are false negatives more costly?

7.2 FUTURE WORK

7.2.1 Fusion of Anomaly Scores

The examined anomaly scores described in section § 6.4.2 each have their own advantages and disadvantages given a dataset, especially as seen on Fig. 6.27. It thus leads one to wonder if the scores could somehow be combined and given different weights or confidences, in order to increase the overall performances.

I.e. the problem lies in seeking a fusion approach that can aggregate the observed anomalous behaviours from each anomaly score into a final score in a somewhat meaningful way. In the end, improving the accuracy and robustness of the novelty detection system.

7.2.2 Preprocessing Processes

Another former Master student have been classifying perfect potatoes from hollow ones by using the supervised CNN method [72]. In the preprocessing stage he effectively utilized the so-called *content aware resizing* (or *seam carving*) algorithm [65], that relies on gradients in order to adaptively removing uninteresting part of images while keeping the interesting parts, such as foreign objects. This way, he reported that 50% in each image dimensions were removed, hence successfully downsampling images to lower dimensions, while keeping the interesting parts.

It could be interesting to explore this downsampling technique onto the datasets used in this thesis and examine how it fare. Subsequently, the complexity of the generative networks should be adapted to being less complex, since the image dimension would likely have been decreased further. Expectedly, if the technique successfully could remove uninteresting parts of the samples, the generative models potentially would have less commonalities to learn within the training data, potentially increasing the overall performances in the inference stage.

7.2.3 Altered Loss Function

Based on the fact the the SSIM metric in general performed well as the anomaly score, and since it is a dissimilarity measure like the MSE, it could be interesting to replace the MSE with the SSIM as the reconstruction term during training of the CAE or CVAE. In particular, [5] explores such a substitution for a CAE network, and reports promising results.

7.2.4 Alternative DAD Techniques

In this thesis, the only DAD technique that have been examined is that of unsupervised generative models. However, this is far away from being the only methods that can be utilized as a deep novelty detection. For a complete and recently updated list over deep novelty/anomaly detection techniques, the curious reader is recommended to take a look at the survey in [12]. Primary, remaining in the field of generative models, mentioned below

are a alternative practices that could be interesting to experiment with on DUTs.

Contrary to *closed-set classifiers*, such as the CNN that are error-prone to samples of unknown classes, *open-set classifiers* can detect samples that does not belong to any of the classes covered in the trained data. *Open-set recognition* typically relies on training deep neural networks in a supervised manner using known classes, and subsequently in the inference stage the algorithm will work as a dynamic novelty detection seeking to identify outliers and store them into a new unknown class while maintaining performance on the known classes. Open-set recognition are a relatively new field that has emerged within the last decade, and its possibilities and potential are vast.

One subfield is for which open-set recognition is combined with a generative network, and the study in [80] is an instance of such a fusion. Together with training a deep supervised classifier, the authors utilize an unsupervised generative model to efficiently learn latent representations within known classes. Hence, in addition to provide supervised class predictions the unsupervised learned representations works as a regularizer, which enables robust novelty detection without harming the accuracy of the known-class accuracy.

The possibility of utilizing a CVAE network in an open-set recognition set-up thus leads to a natural next step. Specially, since foreign objects and other malfunctions of DUTs typically are non-trivial to detect due to their commonalities, a state-of-the-art generative model must be trained for good performances. Hence, it would be compelling to employ a network alike the one in this thesis into a dynamic novelty detection regime and examine how it fares.

APPENDICES

A ADDITIONAL PLOTS

A.1 INVESTIGATION OF LOSSES

Below the investigation of average loss functions during training is plotted.

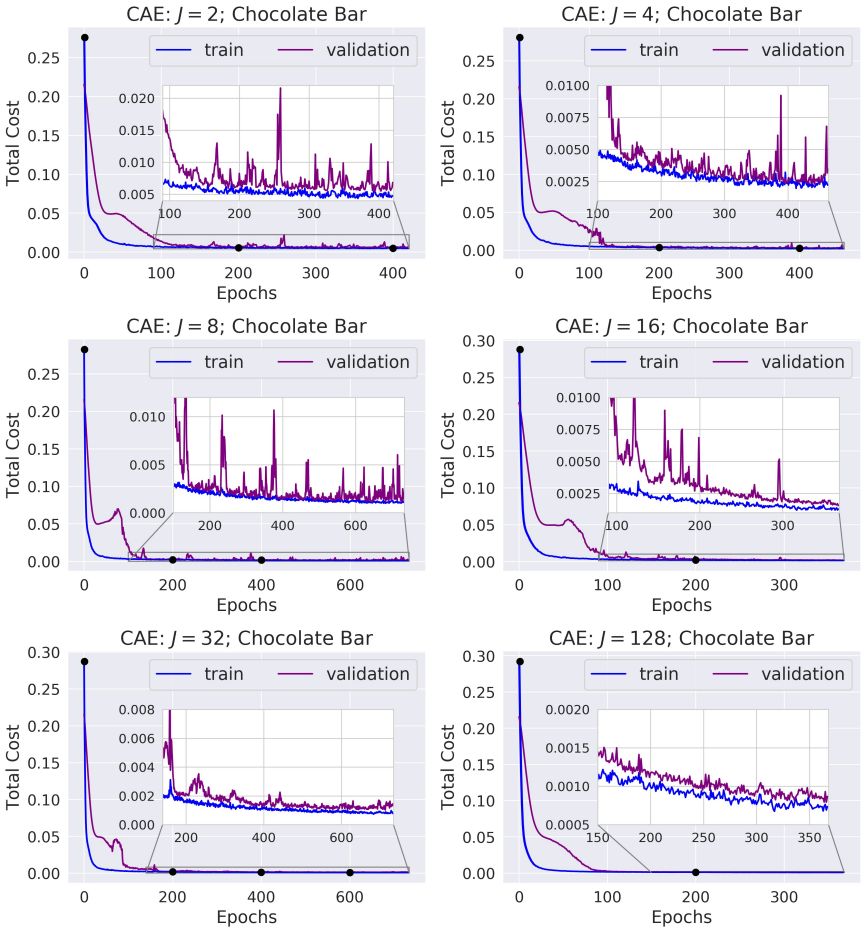


Figure A.1: Average loss functions as a function of number of epochs, derived on training images of the chocolate bar dataset of the CAE model.

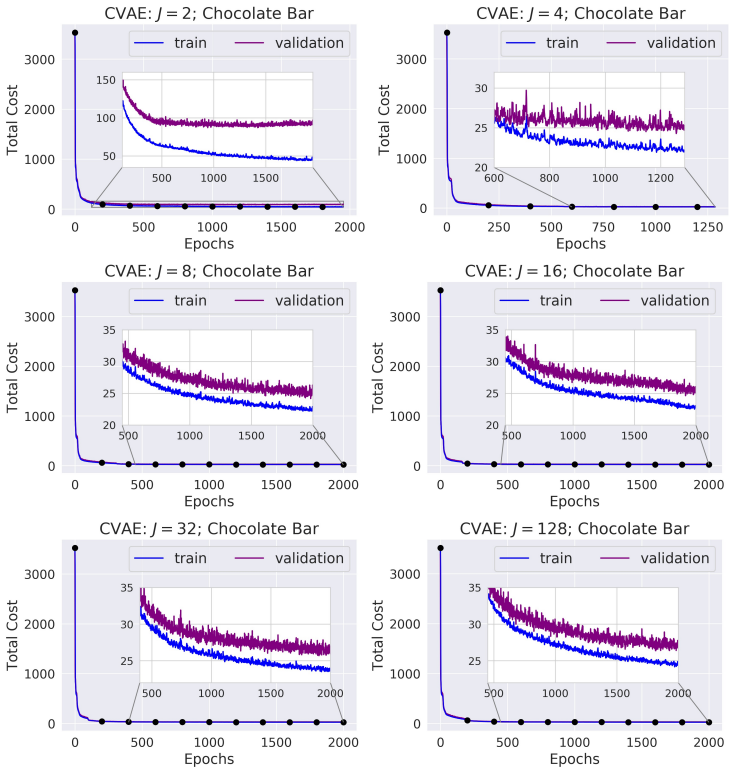


Figure A.2: Average loss functions as a function of number of epochs, derived on training images of the chocolate bar dataset of the CVAE model.

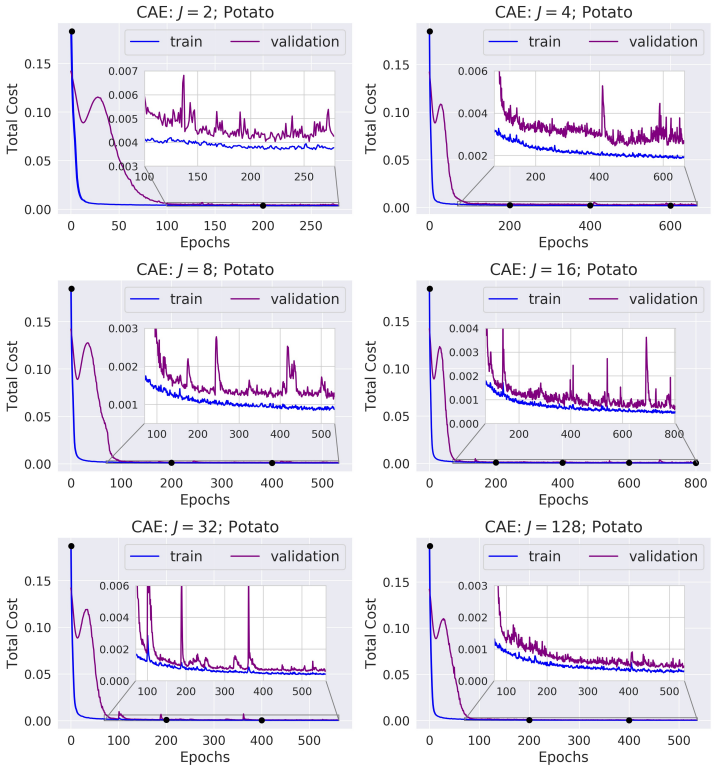


Figure A.3: Average loss functions as a function of number of epochs, derived on training images of the potato dataset of the CAE model.

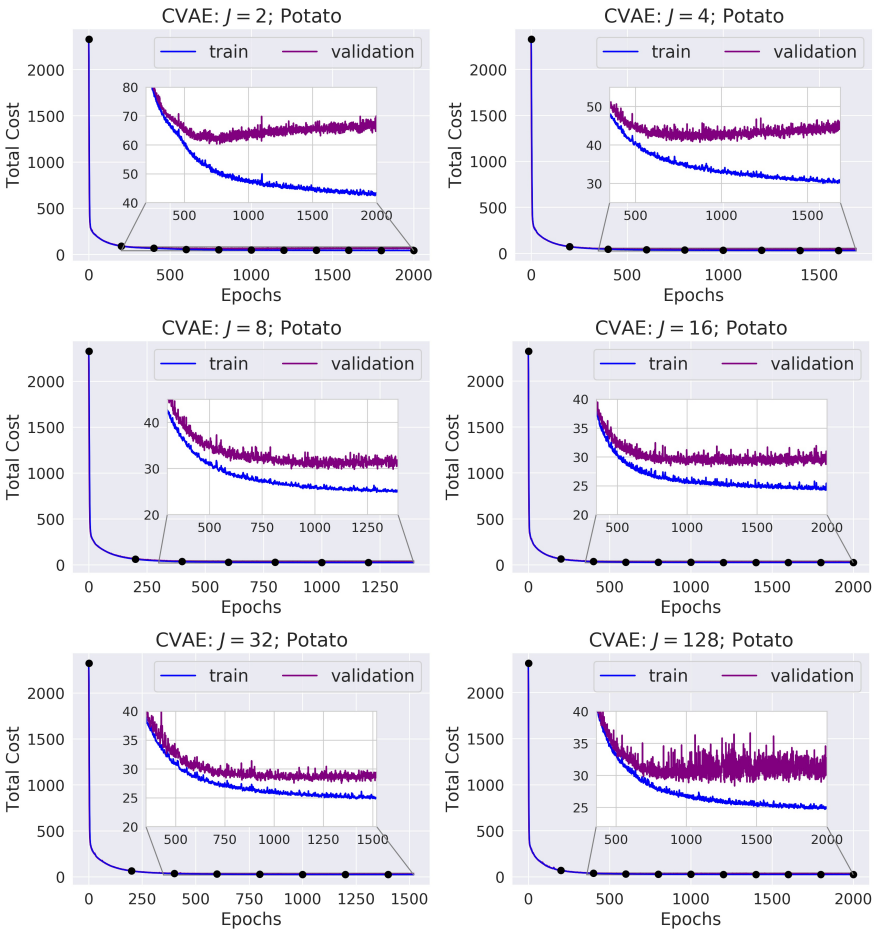


Figure A.4: Average loss functions as a function of number of epochs, derived on training images of the potato dataset of the CVAE model.

BIBLIOGRAPHY

[1] Alla, S. and Adari, S. K. (2019). *Beginning anomaly detection using Python-based deep learning: with Keras and PyTorch*. 1st ed. Apress. ISBN: 1-4842-5177-6 (Cited on pp. 16, 17, 35).

[2] Au, W. and Takei, R. (2002). *Image Inpainting with the Navier-Stokes Equations*. [Available at Final Report APMA 930 SFU.] (Cited on p. 60).

[3] Baldi, P. (Feb. 2012). "Autoencoders, Unsupervised Learning, and Deep Architectures." In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. Ed. by Guyon, I. et al. Vol. 27. Proceedings of Machine Learning Research. Bellevue, Washington, USA: JMLR Workshop and Conference Proceedings, pp. 37–49. URL: http://proceedings.mlr.press/v27/baldi12a.html (Cited on p. 40).

[4] Bank, D., Koenigstein, N., and Giryes, R. (2020). *Autoencoders*. arXiv: 2003.05991 (Cited on p. 41).

[5] Bergmann, P. et al. (2019). "Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders." In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. DOI: 10.5220/0007364503720380 (Cited on p. 109).

[6] Bertalmio, M., Bertozzi, A., and Sapiro, G. (Feb. 2001). "Navier-Stokes, fluid dynamics, and image and video inpainting." In: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit* 1, pp. I–355. DOI: 10.1109/CVPR.2001.990497 (Cited on p. 60).

[7] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. ISBN: 9780387310732 (Cited on pp. 21, 22).

[8] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (Apr. 2017). "Variational Inference: A Review for Statisticians." In: *Journal of the American Statistical Association* 112.518, pp. 859–877. ISSN: 1537-274X. arXiv: 1601.00670 (Cited on p. 43).

[9] Boom, C. D. et al. (2020). *Dynamic Narrowing of VAE Bottlenecks Using GECO and L_0 Regularization*. arXiv: 2003.10901 (Cited on p. 74).

[10] Burger, W. and Burge, M. (2016). *Digital Image Processing: An Algorithmic Introduction Using Java, Second Edition*. Second Edition. London, England: Springer. ISBN: 9781447166832 (Cited on pp. 54, 55).

[11] Carreño, A., Inza, I., and Lozano, J. (June 2020). "Analyzing rare event, anomaly, novelty and outlier detection terms under the supervised

classification framework." In: *Artificial Intelligence Review* 53. DOI: 10.1007/s10462-019-09771-y (Cited on pp. 35, 36).

[12] Chalapathy, R. and Chawla, S. (2019). *Deep Learning for Anomaly Detection: A Survey*. arXiv: 1901.03407 (Cited on pp. 2, 39, 40, 109).

[13] Choi, D. et al. (2020). *On Empirical Comparisons of Optimizers for Deep Learning*. arXiv: 1910.05446 (Cited on p. 25).

[14] Council, A. E. (2011). *Guidelines for Part Average Testing (provides guidelines for using statistical techniques and extended operating conditions to establish part test limits; this approach could be used to provide "Known Good Die)"*. URL: http://yieldwerx.com/part-average-testing-pat-tutorial/?utm_source=towardsdatascience.com&referrer-analytics=1 (Cited on p. 92).

[15] Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. Hoboken: John Wiley & Sons, Incorporated. ISBN: 9780471241959 (Cited on p. 22).

[16] Deecke, L. et al. (Sept. 2018). *Image Anomaly Detection with Generative Adversarial Networks*. URL: https://www.researchgate.net/publication/329944054_Image_Anomaly_Detection_with_Generative_Adversarial_Networks (Cited on p. 91).

[17] Delon, J. (2004). "Midway Image Equalization." In: *Journal of Mathematical Imaging and Vision* 21, pp. 119–134. DOI: 10.1023/B:JMIV.0000035178.72139.2d (Cited on p. 56).

[18] Dilokthanakul, N. et al. (2017). *Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders*. arXiv: 1611.02648.

[19] Doersch, C. (2021). *Tutorial on Variational Autoencoders*. arXiv: 1606.05908 (Cited on p. 46).

[20] Dosovitskiy, A. and Brox, T. (2016). *Generating Images with Perceptual Similarity Metrics based on Deep Networks*. arXiv: 1602.02644 (Cited on p. 80).

[21] Douzas, G., Bacao, F., and Last, F. (Oct. 2018). "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE." In: *Information Sciences* 465, pp. 1–20. ISSN: 0020-0255. arXiv: 1711.00837v2 (Cited on p. 83).

[22] Dumoulin, V. and Visin, F. (2018). *A guide to convolution arithmetic for deep learning*. arXiv: 1603.07285 (Cited on pp. 30–33).

[23] Elam, W., Ravel, B., and Sieber, J. (2002). "A new atomic database for X-ray spectroscopic calculations." In: *Radiation Physics and Chemistry* 63.2, pp. 121–128. DOI: https://doi.org/10.1016/S0969-806X(01)00227-4 (Cited on p. 8).

[24] Getreuer, P. (2011). "Linear Methods for Image Interpolation." In: *Image Processing On Line* 1, pp. 238–259. DOI: 10.5201/ipol.2011.g_lmii (Cited on p. 59).

[25] Ghosh, P. et al. (2020). *From Variational to Deterministic Autoencoders.* arXiv: 1903.12436 (Cited on p. 46).

[26] Gonzalez, R. . (2008). *Digital image processing.* 3.ed. New Jersey: Pearson. ISBN: 013505267x (Cited on pp. 52–55).

[27] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning.* http://www.deeplearningbook.org. MIT Press (Cited on pp. 15, 19, 22, 24–26, 29, 41, 46, 62, 71).

[28] Goodfellow, I. J. et al. (2014). *Generative Adversarial Networks.* arXiv: 1406.2661 (Cited on p. 91).

[29] Grandini, M., Bagli, E., and Visani, G. (2020). *Metrics for Multi-Class Classification: an Overview.* arXiv: 2008.05756 (Cited on p. 84).

[30] Guillemot, T. and Delon, J. (2016). "Implementation of the Midway Image Equalization." In: *Image Processing On Line* 6, pp. 114–129. DOI: 10.5201/ipol.2016.140 (Cited on p. 56).

[31] Halliday, D., Resnick, R., and Krane, K. S. (2002). *Physics: Volume 2.* 5th ed. New York: John Wiley & Sons. ISBN: 9780471401940 (Cited on p. 5).

[32] He, H. and Garcia, E. A. (2009). "Learning from Imbalanced Data." In: *IEEE Transactions on Knowledge and Data Engineering* 21.9, pp. 1263–1284. DOI: 10.1109/TKDE.2008.239 (Cited on pp. 83, 84).

[33] Hernández-García, A. and König, P. (2020). *Data augmentation instead of explicit regularization.* arXiv: 1806.03852 (Cited on p. 72).

[34] Higgins, I. et al. (2017). *beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework.* URL: https://openreview.net/forum?id=Sy2fzU9gl (Cited on p. 91).

[35] Hinton, G. E. et al. (2012). *Improving neural networks by preventing co-adaptation of feature detectors.* arXiv: 1207.0580 (Cited on p. 31).

[36] Imaging, S. (n.d.). *Quality right down the line.* Ed. by stemmer-imaging.com. [Online; acessed 01-May-2021]. URL: https://www.stemmer-imaging.com/en-se/technical-tips/line-scan-cameras/ (Cited on p. 11).

[37] Ioffe, S. and Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* arXiv: 1502.03167 (Cited on pp. 26, 27).

[38] Jordan, M. et al. (Nov. 1999). "An Introduction to Variational Methods for Graphical Models." In: *Machine Learning* 37, pp. 183–233. DOI: 10.1023/A:1007665907178 (Cited on p. 43).

[39] Kingma, D. P. and Ba, J. (2017). *Adam: A Method for Stochastic Optimization.* arXiv: 1412.6980 (Cited on pp. 25, 26).

[40] Kingma, D. P. and Welling, M. (2014). *Auto-Encoding Variational Bayes.* arXiv: 1312.6114 (Cited on pp. 41–43, 45–47).

[41] Kingma, D. P. and Welling, M. (2019). "An Introduction to Variational Autoencoders." In: *Foundations and Trends® in Machine Learning* 12.4, pp. 307–392. ISSN: 1935-8245. arXiv: 1906.02691 (Cited on pp. 45, 47, 48).

[42] Kingma, D. P. et al. (2017). *Improving Variational Inference with Inverse Autoregressive Flow.* arXiv: 1606.04934 (Cited on p. 72).

[43] Knoll, G. F. (2000). *Radiation Detection and Measurement.* 3rd ed. New York: John Wiley. ISBN: 0471073385 (Cited on pp. 6–8).

[44] Kornprobst, P., Tumblin, J., and Durand, F. (Jan. 2009). "Bilateral Filtering: Theory and Applications." In: *Foundations and Trends in Computer Graphics and Vision* 4, pp. 1–74. DOI: 10.1561/0600000020 (Cited on p. 57).

[45] Kriesel, D. (2007). *A Brief Introduction to Neural Networks.* URL: http://www.dkriesel.com/en/science/neural_networks (Cited on pp. 16, 17).

[46] Krizhevsky, A., Sutskever, I., and Hinton, G. (Jan. 2012). "ImageNet Classification with Deep Convolutional Neural Networks." In: *Neural Information Processing Systems* 25. DOI: 10.1145/3065386 (Cited on p. 19).

[47] Kullback, S. and Leibler, R. A. (Mar. 1951). "On Information and Sufficiency." In: *Annals of Mathematical Statistics* 22.1, pp. 79–86. DOI: 10.1214/aoms/1177729694 (Cited on p. 22).

[48] Kumar, T. and Verma, K. (Sept. 2010). "A Theory Based on Conversion of RGB image to Gray image." In: *International Journal of Computer Applications* 7. DOI: 10.5120/1140-1493 (Cited on p. 27).

[49] Kuncheva, L. I. et al. (2018). *Instance Selection Improves Geometric Mean Accuracy: A Study on Imbalanced Data Classification.* arXiv: 1804.07155 (Cited on p. 83).

[50] Lahiri, R. (2018). *Developing an intuition for better understanding of convolutional neural networks.* Ed. by Medium.com. [Online; posted 07-November-2018]. URL: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 (Cited on p. 28).

[51] Li, J. and Lu, B.-L. (Mar. 2009). "An adaptive image Euclidean distance." In: *Pattern Recognition* 42, pp. 349–357. DOI: 10.1016/j.patcog.2008.07.017 (Cited on pp. 93, 94).

[52] Lisani, J.-L., Petro, A.-B., and Sbert, C. (2012). "Color and Contrast Enhancement by Controlled Piecewise Affine Histogram Equalization."

In: *Image Processing On Line* 2, pp. 243–265. DOI: 10.5201/ipol.2012.lps-pae (Cited on p. 52).

[53] Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). *The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables*. arXiv: 1611.00712 (Cited on p. 47).

[54] Maier, A. and Steidl, S. (2013). *Medical Imaging Systems: An Introductory Guide*. Springer Nature. ISBN: 9783319965192 (Cited on pp. 5–7, 9).

[55] Moeslund, T. B. (2012). *Introduction to video and image processing: building real systems and applications*. 1st ed. 2012. London, England: Springer. ISBN: 1-4471-2503-7 (Cited on pp. 52, 53, 57).

[56] Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. http://neuralnetworksanddeeplearning.com. Determination Press (Cited on pp. 15–17, 21–23, 29–31).

[57] Parr, T. and Howard, J. (2018). *The Matrix Calculus You Need For Deep Learning*. arXiv: 1802.01528 (Cited on p. 20).

[58] Plaut, E. (2018). *From Principal Subspaces to Principal Components with Linear Autoencoders*. arXiv: 1804.10253 (Cited on p. 41).

[59] Podgorsak, E. B. (2016). *Radiation Physics for Medical Physicists*. 3rd ed. Graduate Texts in Physics. Cham: Springer International Publishing. ISBN: 97833192538oo (Cited on pp. 6, 7).

[60] Radford, A., Metz, L., and Chintala, S. (2016). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. arXiv: 1511.06434 (Cited on pp. 31, 72).

[61] Russo, P. (2018). *Handbook of X-ray Imaging: Physics and Technology*. Series in Medical Physics and Biomedical Engineering. Boca Raton, Florida: CRC Press. ISBN: 1498741541 (Cited on pp. 7, 8).

[62] Sakurada, M. and Yairi, T. (2014). "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction." In: *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*. MLSDA'14, pp. 4–11. DOI: 10.1145/2689746.2689747 (Cited on p. 82).

[63] Salem, M. et al. (Jan. 2010). "Linear and Non-linear Contrast Enhancement Image." In: *IJCSNS International Journal of Computer Science and Network Security* 10 (Cited on p. 52).

[64] Schlegl, T. et al. (2017). *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*. arXiv: 1703.05921 (Cited on p. 91).

[65] Setlur, V. et al. (2005). "Automatic Image Retargeting." In: MUM '05. Christchurch, New Zealand: Association for Computing Machinery,

pp. 59–68. ISBN: 0473106582. DOI: 10.1145/1149488.1149499 (Cited on p. 109).

[66] Sewak, M., Karim, M. R., and Pujari, P. (2018). *Practical convolutional neural networks: implement advanced deep learning models using Python.* 1st ed. Birmingham: PACKT Publishing. ISBN: 9781788392303 (Cited on pp. 17–20, 27, 28, 32, 74).

[67] Shannon, C. E. (1948). "A Mathematical Theory of Communication." In: *The Bell System Technical Journal* 27.4, pp. 623–656. DOI: 10.1002/j.1538-7305.1948.tb00917.x (Cited on p. 23).

[68] Smith, L. N. (2017). *Cyclical Learning Rates for Training Neural Networks.* arXiv: 1506.01186 (Cited on p. 77).

[69] Springenberg, J. T. et al. (2015). *Striving for Simplicity: The All Convolutional Net.* arXiv: 1412.6806 (Cited on pp. 31, 72).

[70] Srivastava, N. et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of Machine Learning Research* 15.56, pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (Cited on p. 32).

[71] StanfordCSclass (2020). *CS231n: Convolutional Neural Networks for Visual Recognition.* Ed. by stanford.edu. URL: https://cs231n.github.io/neural-networks-1/ (Cited on pp. 16, 18, 19).

[72] Topic, A. (2019). *Adaptive X-ray Inspection System (AXIS): Automating Classification in Food Inspection.* Ed. by Institutet, N. B. (Cited on p. 109).

[73] Wang, L., Zhang, Y., and Feng, J. (2005). "On the Euclidean distance of images." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8, pp. 1334–1339. DOI: 10.1109/TPAMI.2005.165 (Cited on pp. 93, 94).

[74] Wang, Z. et al. (2004). "Image quality assessment: from error visibility to structural similarity." In: *IEEE Transactions on Image Processing* 13.4, pp. 600–612. DOI: 10.1109/TIP.2003.819861 (Cited on p. 95).

[75] Wu, H. and Gu, X. (Nov. 2015). "Towards dropout training for convolutional neural networks." In: *Neural Networks* 71, pp. 1–10. ISSN: 0893-6080. arXiv: 1512.00242 (Cited on p. 28).

[76] Xia, Y. et al. (2015). "Learning Discriminative Reconstructions for Unsupervised Outlier Removal." In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1511–1519. DOI: 10.1109/ICCV.2015.177 (Cited on p. 82).

[77] Xue, W. et al. (2013). *Gradient Magnitude Similarity Deviation: A Highly Efficient Perceptual Image Quality Index.* arXiv: 1308.3052 (Cited on pp. 97, 98).

[78] Yen, E. K. and Johnston, R. G. (1996). "The Ineffectiveness of the Correlation Coefficient for Image Comparisons." In: *Technical Report* (Cited on pp. 96, 97).

[79] Yeung, S. et al. (2017). *Tackling Over-pruning in Variational Autoencoders*. arXiv: 1706.03643 (Cited on p. 72).

[80] Yoshihashi, R. et al. (2019). *Classification-Reconstruction Learning for Open-Set Recognition*. arXiv: 1812.04246 (Cited on p. 110).

[81] Zafar, I. e. a. (2018). *Hands-On Convolutional Neural Networks with TensorFlow: Solve Computer Vision Problems with Modeling in TensorFlow and Python*. Birmingham: Packt Publishing Ltd. ISBN: 9781789132823 (Cited on pp. 17, 18, 20, 25, 28, 41, 61).

[82] Zipf, M. (Apr. 2010). "Radiation Transmission-based Thickness Measurement Systems - Theory and Applications to Flat Rolled Strip Products." In: DOI: 10.5772/8729 (Cited on p. 10).

COLOPHON

This document was typeset using the custom LaTeX 2_ε document class `dionsthesis`, based on `uiothesis` developed by Eivind Uggedal. It uses Linux Libertine, and Fira Sans, developed by the Mozilla Foundation, as body fonts. `dionsthesis` is available at:

https://github.com/dionhaefner/dionsthesis/

The style of `uiothesis` was inspired by Robert Bringhurst's seminal book on typography *The Elements of Typographic Style*. Typographic, structural and graphical decisions in this document follow the ideas presented in Jean-Luc Doumont's book *Trees, Maps, and Theorems*.

DECLARATION

I declare that this thesis is my own personal effort. I have not already obtained a degree on the basis of this work. Furthermore, I took reasonable care to ensure that the work is original, and, to the best of my knowledge, does not breach copyright law, and has not been taken from other sources except where such work has been cited and acknowledged within the text.

Copenhagen, May 20, 2021

Alina Sode