Master of Science Thesis
HT2020

Noise Reduction of Scintillation Camera Images Using UNET: A Monte Carlo Simulation Approach

Johanna Korpijärvi

Supervisors

Linda Knutsson, Michael Ljungberg, Sajad Mohammed Ali and Johan Gustafsson

Medical Radiation Physics, Lund
Faculty of Science
Lund University
www.msf.lu.se

Contents

1 Acknowledgments

I devote my gratitude to my supervisors **Linda Knutsson**, **Michael Ljungberg**, **Sajad Mohammed Ali** and **Johan Gustafsson** for making my master thesis possible. You have been encouraging and respected my wishes by including programming and nuclear medicine into my master thesis.

Michael Ljungberg, thank you for all the help and advice regarding SIMIND and XCAT. Especially for all the lectures you have dedicated to the nuclear master thesis students. I am immensely grateful for the time you have spent to answer the countless emails we students have sent you. Your enthusiastic and positive attitude has encouraged us students.

Linda Knutsson and **Sajad Mohammed Ali**, I am beyond satisfied with incorporation of AI into my master thesis. It is a rapid developing field, and becoming more and more incorporated into our society. Thank you Linda for always having a positive attitude. Thank you Sajad for your supportive attitude and for all the help you have been giving me. The presentations you held and the codes you have reviewed. I am beyond grateful. And as we say in Swedish "god fortsättning" to your PhD.

Selma Curkic thank you for all your support and the kinetic curves you made for your master thesis, which I borrowed and used in mine.

Michael Ljungberg and **Johan Gustafsson**, I am thankful for all the nuclear medicine meetings you have dedicated to the nuclear master thesis students. The support and feedback the nuclear medicine group has given us regarding our presentations has been valuable.

2 Populärvetenskaplig sammanfattning

KAN EN DATOR LÄRA SIG BRUSREDUCERA NUKLEAR-MEDICINSKA BILDER?

Inom nuklearmedicinsk bildtagning injiceras en patient med ett läkemedel bundet till en radionuklid. De fotoner som emitteras kan detekteras med en gammakamera, vilket ger en planar bild av hur läkemedlet tagits upp i kroppen. Brusnivån relaterar till antalet fotoner som detekteras. En längre bildtagningstid och högre injicerad aktivitet minskar brusnivån, men ökar sannolikheten för rörelseartefakter respektive DNA skador. Alternativa metoder, exempelvis, filter kan användas för brusreducering, men på bekostnad av upplösningen i bilden. Syftet med maskininlärning är att lära en dator brusreducera nuklearmedicinska bilder och samtidigt försöka bibehålla upplösningen.

MASKININLÄRNING

Maskininlärning är en subkategori inom artificiell intelligens, där en dator tränas till att lösa ett problem med hjälp av indata och utdata. Detta kallas väglett lärande. Träningen går ut på att mata en algoritm med indata och utdata, som de träningsbara parametrarna i algoritmen justeras utefter. Från den färdigtränade algoritmen fås en modell, som utifrån indata som modellen inte tränats med ska kunna ge en prediktering.

TRÄNINGSDATA

För att träna en modell med indata och utdata måste algoritmen ha tillgång till en stor mängd träningsmaterial. Att tillhandahålla nuklearmedicinska bilder från kliniken är både omfattande och onödigt. Utdatan -vilket inom maskininlärning benämns labels- simulerades istället fram med virtuella patienter och en virtuell gammakamera, programvaran som användes var XCAT respektive SIMIND. En gammakamera kan simuleras eftersom radioaktivt sönderfall och fotoninteraktioner är slumpmässiga processer, som kan modelleras med Monte Carlo metoden. Via Monte Carlo metoden kan en foton spåras från sönderfallsplatsen ända in i den virtuella gammakameran. Labels som simulerades fram med SIMIND var brusfria. Indatan bestod av kopior av labels med Poisson distributerat brus. Inom maskininlärning benämns indata för samples. Varje sample hade en label, vilket motsvarade den brusiga respektive brusfria bilden. Label bilderna simulerades för radionukliden 111-In.

MASKININLÄRNINGS ARKITEKTUREN UNET

Maskininlärningsarkitekturen som valdes för att brusreducera de planara gammakamerabilderna kallas UNET, och består av en expansiv och en kontraherande del, vilket ger dess U-formade utseende. Den kontraherande delen minskar dimensionerna på indatan för att extrahera olika drag och mönster från samples via den matematiska operationen faltning. Den expanderande delen

förstorar dimensionerna istället. Även den expanderande delen består av faltningsoperationer.

RESULTAT OCH SLUTSATS

UNET modell tränad med normaliserad-data kunde brusreducera planara gammakamera bilder, men presterade sämre än Butterworth filtret om man tittar på de normaliserade profilerna. Butterworth filtret är ett standard filter som används inom kliniken. Problemet med att använda normaliserad data är att man behöver en konverteringsfaktor till aktivitetskoncentrationer. Det man måste ha i åtanke är att alla maskininlärnings modeller är till någon grad anpassad till träningsdatan (overfitting). För att träna en modell som presterar bättre och är mer generaliserad, och kan ge en godtycklig prediktering för data som modellen inte tränats på, behövs en stor mängd träningsdata med en stor spridning bland de virtuella patienterna. Att få en stor spridning bland de virtuella patienterna var utmanade, eftersom tillgången på basfantom i XCAT var begränsad. På basfantomen applicerades små ändringar för att skapa nya fantom. De ändringar som applicerades på basfantomen var eventuellt bristfälliga för att representera en stor patientpopulation.

3 Abbreviations

SPECT- Single-photon emission computed tomography
PMT- Photo-multiplier tubes
SNR- Signal to noise ratio
AI- Artificial intelligence
CNN- Convolutional neural network
ReLu- Rectified linear unit
GD- Gradient descent
MSE- Mean squared error
SGD- Stochastic gradient descent
MAE- Mean absolute error
FCN- Fully convolutional network
PET- Positron emission tomography
MSSIM- Mean structural similarity index
SSIM- Structural similarity index
NRMSE- Normalized root mean square error
PSNR- Peak signal to noise ratio

4 Abstract

Background: The project aims to reduce the noise in planar 111-In projections with a machine learning model.

Method: Sixteen base phantoms were used to create 696 phantoms in XCAT. An anterior planar projection was simulated with 111-In and ten uptake curves for each phantom in SIMIND. From the 6960 simulated planar projections, 55600 sample and label pairs were created by adding Poisson distributed noise, scaling, rotation, and shift. The label was the noiseless representation of the sample. The normalized sample and label pairs were used to train four UNETS with different hyperparameters. The UNETs were evaluated with the metrics mean NRMSE (normalized root mean square error), mean PSNR (peak signal to noise ratio), and mean MSSIM (mean structural similarity index). A fifth UNET was trained with 55600 sample and label pairs without normalization. A UNET trained with and without normalized training data was compared to a standard Butterworth filter. The comparison was performed using the normalized profiles from the different images.

Results: The smallest mean PSNR was 53. The largest mean NRMSE was 0.19 and the mean MSSIM was 0.81. The difference in the means between the UNETs for the different evaluation metrics was evaluated with a one-way ANOVA test for the test data containing no augmentation added except scaling. The one-way ANOVA test showed no statistical difference between the means for the different UNETs with regards to the calculated PSNR and NRMSE values. However, a statistically significant difference was found between the means of the different UNETs for the calculated MSSIM values. The Butterworth profile was more consistent with the ground truth label for the test data than the profiles from the UNETs trained with normalized and un-normalized data. The UNET from the un-normalized resulted in blurrier predictions compared to the UNET trained with normalized data.

Conclusions: The UNET can reduce the noise in scintillation camera images with a high noise level. However, the UNET trained could not recover details such as the ribs. Further work is needed to optimize the training data and the architecture to recover details lost in the imaging processing. The images from the UNET trained with normalized training data were not as blurry as the images filtered with the Butterworth filter.

5 Introduction

Scintillation camera images have limited spatial resolution and a large amount of Poisson distributed noise. A wide range of noise-reducing filters is available for example the low-pass Butterworth filter. However, most noise-reducing filters reduce the noise at the expense of resolution loss. Different convolutional neural networks have been proposed and have shown good results for photographic images. However, training a convolutional neural network for scintillation camera images is challenging, since noiseless images are required. Moreover, enhancement of medical images requires that no false information is added or crucial information is lost in the process. The enhancement should solely reconstruct information lost in the image acquisition.

6 Aim

This thesis aimed to:

1. investigate if a specific convolutional neural network, the UNET, can be trained to reduce the noise in 111-In planar projections simulated with a virtual gamma camera.

2. compare the UNET model's performance to a Butterworth filter.

7 Background

7.1 Nuclear Medicine Imaging

Nuclear medicine imaging utilizes the photon decay of a radioactive isotope labeled to a pharmaceutical. Since the pharmaceutical corresponds to an uptake mechanism in the patient, the image retrieved from a photon position-sensitive detector system reflects the organ function. The images have several clinical applications such as diagnosis of pulmonary embolism and tumor diagnosis.

7.2 Scintillation Camera Principles

The scintillation camera is a position-sensitive detector system for planar imaging and single-photon emission computed tomography (SPECT), see figure 2. The main components are a collimator, a scintillation crystal and photo-multiplier tubes (PMT) [1], see figure 1.

The collimator filters the incident photons by direction. The aperture is an array of lead consisting of holes and septa, with various geometric designs available. Nuclear medicine imaging utilizes, for example, parallel-hole collimators made of parallel holes and septa.[2] The parallel hole collimator attenuates photons with an oblique incidence angle, whereas photons with a direction parallel to the holes are transmitted through the collimator and can impinge the crystal.[1]

The photon interacts and ionizes atoms in the crystal, and energy is transferred to ejected electrons. The atoms are additionally excited by the electrons creating electron-hole pairs. In inorganic crystals, for instance, the frequently used NaI(TL), the electron-hole pairs migrate in the crystal lattice. Scintillation photons of visible wavelengths are emitted when electron-hole pairs recombine between discrete states in the forbidden band of the impurity atoms. These photons can not be reabsorbed since the energy is not enough to excite an electron from the valence band to the conduction band.[3]

The purpose of the PMT is to convert scintillation light to an electronic signal. The PMT consist of a photocathode, where weakly bound valence electrons absorb incident photons. If the absorbed energy exceeds the binding energy, the electron is ejected. The emitted electrons are multiplied in dynodes followed by charge collection in the anode. An array of PMTs is mapped to the crystal. The narrow beam of scintillation photons diverges, generating a signal in multiple PMT. The photon interaction site correlates mostly to the spatial coordinates of the PMT with maximum signal intensity. The centroid coordinates of all the signals decrease the spatial resolution compared to the PMT size, called the Anger logic centroid approach. Coordinates

are collected and placed in an image array. The image is essentially a map, showing the locations of radioactive decay.[2]

Figure 1: Dual headed gamma camera.

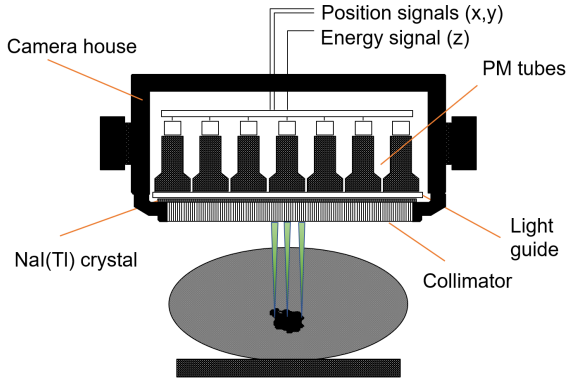

Figure 2: The main components of a scintillation camera based on Anger logic centroid approach.

7.3 Scintillation Camera Image Noise and Spatial Resolution

The statistical error from the randomness of radioactive decay creates Poisson distributed noise in SPECT projections and planar scintillation images.[2] The Poisson model estimates the number of events in a fixed time interval if the probability of an event for each trial is small, the events are independent and appear at a constant rate. An event in the scintillation camera can, for example, be detecting a radioactive decay. The constant rate of events is valid when the number of nuclei is large and the observation time is short compared to the specimens' half-life. According to the Poisson model, the mean value is equal to the variance, and the variance is the squared standard deviation, hence

$$\lambda = \sigma^2 \ , \tag{1}$$

where λ is the mean value and σ is the standard deviation.[3] The signal-to-noise ratio (SNR) can be expressed as,

$$\mathrm{SNR} = \frac{\lambda}{\sigma} = \frac{\lambda}{\sqrt{\lambda}} = \sqrt{\lambda} \ . \tag{2}$$

SNR is enhanced and image noise is reduced by increasing parameters proportional to the mean, namely the administered activity and the time per projection.[2] The former is restricted by the core value of targeted radionuclide imaging for therapies and diagnostic: acquire image information with an absorbed dose as low as reasonably achievable (ALARA).[4] The latter is at the expense of patient convenience and the risk of motion artifacts.[2] The count rate performance correlates to the image

mean. Therefore the collimator's geometric dimensions must be considered. However, an increase in the geometrical sensitivity can not avoid an increase in the geometrical spatial resolution. The trade-off between sensitivity and spatial resolution is essentially a compromise between SNR and spatial properties of the acquired image.

7.4 Low-Pass Filtering in Frequency Domain

Low-pass filters reduce noise, therefore often used as a post-acquisition filter. On the other hand, the filter also degrades the spatial resolution.[2] Low pass filtering in the frequency domain filters high-frequency components and retains the low. The former contains the details in the image and contributes to a significant amount of noise. The latter contains the main features of the image. Filtering in the frequency domain is defined as,

$$\mathrm{R(u,v)} = \mathrm{W(u,v)} \cdot \mathrm{G(u,v)} \ , \tag{3}$$

the filtered image R(u,v) is a multiplication between the unfiltered image G(u,v) and the filter W(u,v). Several low-pass filters in the frequency domain are available, for example, the ideal low-pass filter and the Butterworth low-pass filter. The Butterworth filter is preferred since the ideal low-pass filter is essentially a box function in the frequency domain. The box function is equivalent to a sinc function in the spatial domain, increasing the probability of ring artifacts, unlike the Butterworth filter with a more graded transition from 0 to 1. The Butterworth filter is given by,

$$\mathrm{W(u,v)_{Butter,lp}} = \frac{1}{1 + (\frac{\mathrm{D(u,v)}}{\mathrm{D_0}})^{2\mathrm{n}}} \ , \tag{4}$$

where n is the filter order controlling the slope of the amplitude, a higher value of n gives a steeper slope, increasing the probability for ring artifacts. $\mathrm{D_0}$ is the frequency where the amplitude has decreased from 1 to 0.5. The noise reduction increases for a lower $\mathrm{D_0}$ value since more of the high-frequency components are filtered out, see figure 3. However, the degradation of spatial resolution increases with the loss of high-frequency components.

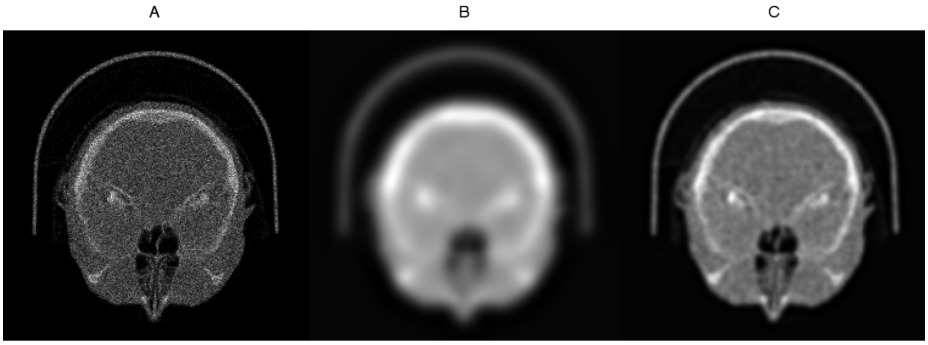

Figure 3: A is the noisy image, B and C is the noisy image filtered with different Butterworth filters. Image B is filtered with n=2 and $D_0 = 10$, while image C is filtered with n=2 and $D_0 = 30$.

7.5 Machine Learning

Machine learning is a field of artificial intelligence (AI), where the algorithm learns without human interference. Meaning, the algorithm filters out important information and gains knowledge from the training data. The algorithm has not been explicitly programmed, note explicitly programmed algorithms is another subcategory of AI. A trained model can make predictions on unseen data. In other words, data that the model has not been trained with.[5]

Supervised learning is a sub-field of machine learning, and the training data contain sample and label pairs. **The label is the prediction we desire from the model for a given sample.**[5]

7.6 Convolutional Neural Networks (CNN)

A convolutional neural network (CNN) is a machine learning model that can be applied to images since it is designed to extract array patterns. CNN is constructed of three types of layers, namely convolutional, pooling, and fully connected. The former are feature extractors and the latter maps extracted features in classification problems to a class output. Feeding the output from one layer as input in another can hierarchically extract more complex features. CNN architecture is often constructed of a repeating set of layers, for example, a convolution layer or layers followed by a pooling layer. In classification tasks, the repetition of layers is followed by at least one fully connected layer. In the forward propagation, the input data is modified through the layers giving output data.[6]

7.6.1 Convolutional Layer

The convolution layer for a three-dimensional image and filter is based on discrete convolution in two dimensions since striding is only performed in two dimensions. In 2d convolution, the filter is stridden over the image, and stride is the distance between two filter positions, denoted s. $s > 1$ shrinks the image. The image's third dimension represents the color channels. An RGB image has three color channels, while a gray-scale image has one. Hence, the dimensions of the input image, A, are given by.

$$\dim(\mathrm{A}) = (\mathrm{n_H}, \mathrm{n_W}, \mathrm{n_C}) \ , \tag{5}$$

where $\mathrm{n_H}$, $\mathrm{n_W}$ and $\mathrm{n_C}$ are the height, width, and the number of color channels, respectively. The convolutional layer takes the number of filters as an argument. Each filter, K, in the layer has the same dimension, given by,

$$\dim(\mathrm{K}) = (\mathrm{f}, \mathrm{f}, \mathrm{n_C}) \ , \tag{6}$$

where f is the width and the height of the filters. f is usually odd, often equal to three, five, or seven. The two-dimensional convolution on volumes, namely the input image and each filter is determined with,

$$\begin{aligned} &\mathrm{O}(\mathrm{x}, \mathrm{y}) = \mathrm{conv}(\mathrm{A}, \mathrm{K})_{\mathrm{x,y}} = \\ &\sum_{\mathrm{i}=0}^{\mathrm{f}-1} \sum_{\mathrm{j}=0}^{\mathrm{f}-1} \sum_{\mathrm{k}=0}^{\mathrm{n_C}-1} \mathrm{A}(\mathrm{x} + \mathrm{i} - \mathrm{n_H}/2, \mathrm{y} + \mathrm{j} - \mathrm{n_W}/2, \mathrm{k}) \cdot \mathrm{K}(\mathrm{i}, \mathrm{j}, \mathrm{k}) \ , \end{aligned} \tag{7}$$

where A is the image matrix, K is a filter matrix also called kernel and O is the resultant matrix or the feature map. x and y are the image matrix indices, whereas i, j, and k are the kernel indices. The filters' constitutes called weights are model-parameters or learnable parameters, while the size, f, and the number of filters are hyper-parameters. **Model-parameters also called learnable parameters are updated during training, while the hyper-parameters stay fixed.** For equation 7 the number of trainable weights, #w can be calculated with,

$$\#\mathrm{w} = \mathrm{f} \cdot \mathrm{f} \cdot \mathrm{n_C} \ . \tag{8}$$

According to equation (7), the kernel center never overlaps with the utmost elements. Therefore, the output image, O(x,y), is smaller than the input image, A(x,y), if zero padding is not added to A(x,y). Zero padding means surrounding the image A(x,y) with zeros. Another less common padding option is edge wrap. The padding parameter p is the number of elements added to each side of the width and height, see figure 4.

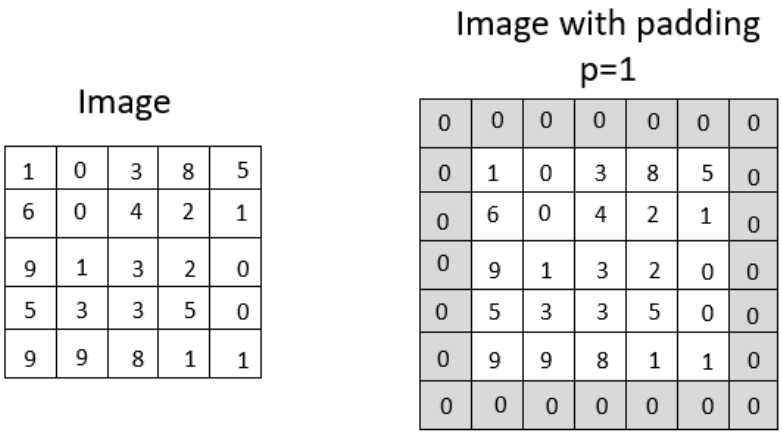

Figure 4: Image with and without padding p=1

The dimension of O(x,y) is a function of padding, p, and stride s, given by,

$$\dim(\mathrm{O}(\mathrm{x},\mathrm{y})) = \left(\lfloor \frac{\mathrm{n_H} + 2\mathrm{p} - \mathrm{f}}{\mathrm{s}} + 1 \rfloor, \lfloor \frac{\mathrm{n_W} + 2\mathrm{p} - \mathrm{f}}{\mathrm{s}} + 1 \rfloor \right) \ , \tag{9}$$

if $\mathrm{s} > 0$. p=0 means no padding is added. f=1 is essentially decreasing $\mathrm{n_C}$ to 1 without changing $\mathrm{n_H}$ and $\mathrm{n_W}$. For dim(A(x,y)) to be equal to dim(O(x,y)), p should be set to,

$$\mathrm{p} = \frac{\mathrm{f} - 1}{2} \ . \tag{10}$$

Multiple kernels are convolved with the input image forming a convolutional layer. The output from the convolutional layer is several feature maps stacked in a three-dimensional array. Each feature map contains different attributes extracted from the image. Therefore the filters are often called feature extractors or kernels. For example, a filter can be trained to recognize an edge and the feature map represents the spatial locations where an edge can be found in the input image A. The CNN consists of several layers, l, meaning the input image volume in a convolutional layer is the output volume from a previous layer if $\mathrm{l} > 1$. For layer $\mathrm{l} = 1$, the input image volume is equal to the input image volume in the CNN. The input and output image volumes for a convolutional layer are denoted $\mathrm{a}^{[\mathrm{l}-1]}$ and $\mathrm{a}^{[\mathrm{l}]}$ with dimensions $\dim(\mathrm{n_H^{[l-1]}}, \mathrm{n_W^{[l-1]}}, \mathrm{n_C^{[l-1]}})$ and $\dim(\mathrm{n_H^{[l]}}, \mathrm{n_W^{[l]}}, \mathrm{n_C^{[l]}})$, respectively. Meaning $\mathrm{a}^{[0]}$ is the input image volume for the CNN. Padding and stride can be adjusted between layers in a CNN, hence the notation $\mathrm{p}^{[\mathrm{l}]}$ and $\mathrm{s}^{[\mathrm{l}]}$. The number of filters in a layer l is given by $\mathrm{n_C^{[l]}}$, where each filter and convolution operation is denoted with $\mathrm{K}^{(\mathrm{n})}$ and n, respectively. Thus $\forall \mathrm{n} \in [1, 2, ..., \mathrm{n_C^{[l]}}]$. A convolutional layer, l, consist of an activation function, $\psi^{[\mathrm{l}]}$, and a bias $\mathrm{b_{(n)}^{[l]}}$ for each convolution n, see section 7.6.2. The n^{th} convolution operation in a layer l, can be expressed as,

$$\psi^{[l]}\big(b^{[l]}_{(n)} + \mathrm{conv}(a^{[l-1]}, K^{(n)})_{x,y}\big) = \psi^{[l]}\Bigg(b^{[l]}_{(n)} + \sum_{i=0}^{f^{[l]}-1}\sum_{j=0}^{f^{[l]}-1}\sum_{k=0}^{n_C^{[l-1]}-1} a^{[l-1]}(x+i-f^{[l]}/2, y+j-f^{[l]}/2, k)\cdot K^{(n)}(i,j,k)\Bigg) \ , \tag{11}$$

The output from layer, l, is then,

$$a^{[l]} = \big[\psi^{[l]}\big(b^{[l]}_{(1)} + \mathrm{conv}(a^{[l-1]}, K^{(1)})_{x,y}\big), ..., \psi^{[l]}\big(b^{[l]}_{(n)} + \mathrm{conv}(a^{[l-1]}, K^{(n)})_{x,y}\big)\big] \ , \tag{12}$$

and has the dimension,

$$\dim(a^{[l]}) = \Bigg(\lfloor\frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1\rfloor, \lfloor\frac{n_W^{[]l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1\rfloor, n_C^{[l]}\Bigg) \ , \tag{13}$$

if $s > 1$. Except for the weights, the bias $b_n^{[l]}$ is also a model parameter. The total learnable parameters for a convolutional layer is given by,

$$\#w + \#b = (f^{[l]} \cdot f^{[l]} \cdot n_C^{[l-1]}) \cdot n_c^{[l]} + n_c^{[l]} \ , \tag{14}$$

where #b is the bias contribution to the total number of model parameters.[7] The convolutional layer's main characteristic is weight sharing, namely the same kernel is stridden through the whole image. Weight sharing reduces the number of trainable parameters in comparison with the fully connected layer. Convolutional layers combined with downsampling, for example, pooling, can be used to extract spatial hierarchies.[6] The in-and output pixel values of a convolutional layer are called nodes.[8]

7.6.2 Non-linear Activation Function and Bias

Real-world problems can be complex, and the relation between inputs and correct outputs is often non-linear. Non-linearity in CNN is introduced through non-linear activation functions applied element-wise, see equation (7). The function can limit the output values from a convolutional or fully connected layer to be in a finite range, for example, the logistic function. The logistic functions have the characteristic sigmoid curve or S-shaped curve, see figure 5. The sigmoid functions are known to be bounded by two horizontal asymptotes $x \to \pm\infty$ and are differentiable with a

positive derivative for all real values. The logistic function is mainly used as the last layer activation function for normalized CNN input data since the function maps the values to be between (0,1). The logistic function is defined as,

$$f(x) = \frac{1}{1 + e^{-x}} \quad . \tag{15}$$

The activation function's derivative is essential for the back-propagation, the method used to update the weights. The ReLU (Rectified linear unit) activation is preferred over the logistic function since the logistic function has a more calculation demanding derivative and vanishing gradients for $x \to \pm\infty$. The ReLU function is given by,

$$f(x) = \max\{0, x\} \quad . \tag{16}$$

The dying ReLU problem is caused by weight updating resulting in values ≤ 0 being passed on to the ReLU function. The weights are not updated, since ReLU has a zero derivative for values ≤ 0. The issue can partly be solved with Leaky ReLU defined as,

$$f(x) = \max\{\alpha x, x\} \quad , \tag{17}$$

where α is a small constant.[9]

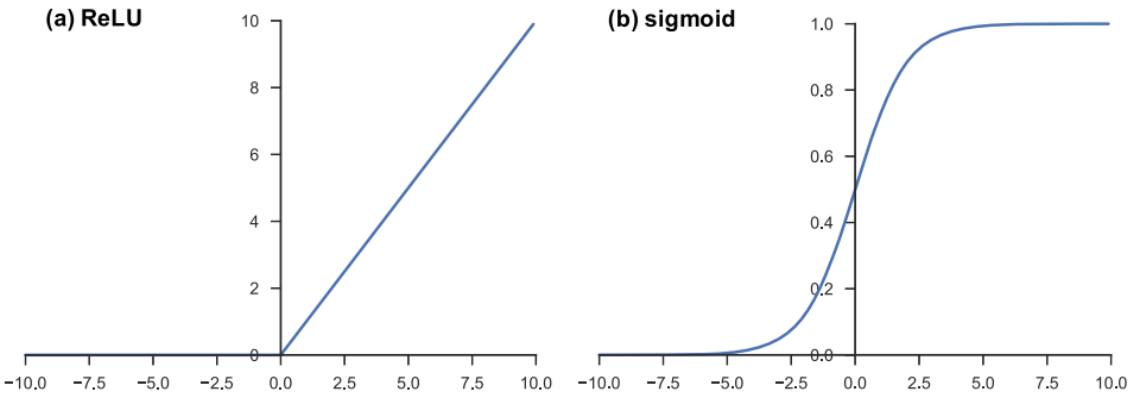

Figure 5: ReLU and sigmoid function.[6]

The bias is a learnable parameter that shifts the activation function with the term b, which was added element-wise to each value in a feature map, see equation (11). The bias makes the model more flexible.[6]

7.6.3 Kernel Initializers

Small kernel or filter values can result in small pixel values that can decrease as the values pass through the layers in the network. Small values being passed on to the

activation function can prevent weight updating. For values$\leq$0 and larger negative values have a small derivative for the ReLU and the sigmoid function, respectively. For larger weight values, the pixel values can increase for each layer and become massive especially with the ReLU activation function.[10] A suitable kernel initializer can avoid decreasing and increasing the pixel values exponentially.[11] He normal is a kernel initializer derived by He et al.[11] to resonate well with the ReLU activation function. The He normal kernel initializer draws the weights from a truncated normal distribution centered at 0 with the following standard deviation,

$$\text{stddev}_{\text{He}} = \sqrt{\frac{2}{\text{f}^{\text{l}} \cdot \text{f}^{\text{l}} \cdot \text{n}_{\text{C}}^{\text{l}-1}}} \ , \tag{18}$$

f is the filter size for layer l. $\text{n}_{\text{C}}^{\text{l}-1}$ represents the number of feature maps if $\text{l} > 1$ or the number of color channels if $\text{l} = 1$, for the image volume inserted into the convolutional 2d layer. The product $\text{f}^{\text{l}} \cdot \text{f}^{\text{l}} \cdot \text{n}_{\text{C}}^{\text{l}-1}$ represents the number of weights a filter or kernel contains in the convolutional 2d layer l.[11] He uniform is a kernel or filter initializer, which draws weights from a uniform distribution within the bounds $[-\text{limit}_{\text{He}}, +\text{limit}_{\text{He}}]$. The limit_{He} is set accordingly [12],

$$\text{limit}_{\text{He}} = \sqrt{\frac{6}{\text{f}^{\text{l}} \cdot \text{f}^{\text{l}} \cdot \text{n}_{\text{C}}^{\text{l}-1}}} \ . \tag{19}$$

Glorot and Bengio[13] pointed out that a random kernel or filter initialization is unsuited when using a logistic sigmoid activation function. Glorot and Bengio[13] further mention that the back-propagated gradients decrease while moving backward in the machine learning algorithm. Consequently, the later layers' learnable parameters or model parameters are updated to a larger degree than for the earlier layers. Subsequently proposed the glorot uniform kernel initializer, which draws filter or weight values from a uniform distribution within the bounds $[-\text{limit}_{\text{glorot}}, \text{limit}_{\text{glorot}}]$. The $\text{limit}_{\text{glorot}}$ is defined as,

$$\text{limit}_{\text{glorot}} = \sqrt{\frac{6}{\text{f}^{\text{l}} \cdot \text{f}^{\text{l}} \cdot \text{n}_{\text{C}}^{\text{l}-1} + \text{f}^{\text{l}} \cdot \text{f}^{\text{l}} \cdot \text{n}_{\text{C}}^{\text{l}}}} \ , \tag{20}$$

where $\text{n}_{\text{C}}^{\text{l}}$ represents the number of feature maps from convolutional 2d layer l.[13] Glorot normal filter or weight initialization samples filter values form the truncated normal distribution centered around 0 and the standard deviation is given by,

$$\text{stddev}_{\text{glorot}} = \sqrt{\frac{2}{\text{f}^{\text{l}} \cdot \text{f}^{\text{l}} \cdot \text{n}_{\text{C}}^{\text{l}-1} + \text{f}^{\text{l}} \cdot \text{f}^{\text{l}} \cdot \text{n}_{\text{C}}^{\text{l}}}} \ . \tag{21}$$

The truncated normal distribution draws filter values from the normal distribution and values acquired two standard deviations from the mean are re-sampled. The mean and the standard deviation are usually set to 0 and 1, respectively.[14]

7.6.4 Pooling Layer

The pooling layer takes a three-dimensional stack of feature maps or images as an argument. The layer consists of a downsampling operation that reduces the dimensions of n_H and n_W of each feature map or image; hence the number of channels n_C remains unchanged. The downsampled feature maps or images are more invariant to small shifts. The operation does not contain any learnable parameters but decreases the number of learnable parameters in the coming convolutional or fully connected layer.[6] A filter is stridden through each feature map. At a certain filter position several elements in the feature map are covered by the filter, and a pooling function φ is applied to the covered elements, accordingly,

$$a^{[l]}_{x,y,z} = \text{pool}(a^{[l-1]})_{x,y,z} = \varphi^{[l]}((a^{[l-1]}_{x+i-1,y+j-1})_{(i,j)\in[1,2,\ldots,f^{[l]}]^2}) \ , \tag{22}$$

where $a^{[l]}$ and $a^{[l-1]}$ is the output and the input from a pooling layer l in the CNN, with the dimensions $(n^{[l]}_H, n^{[l]}_W, n^{[l]}_C)$ and $(n^{[l-1]}_H, n^{[l-1]}_W, n^{[l-1]}_C)$, respectively. Note $n^{[l]}_C$ is equal to $n^{[l-1]}_C$. $a^{[0]}$ is the input image in the CNN. The filter has the dimension $(f^{[l]}, f^{[l]})$. φ pooling function can be a max function, that gives the maximum of a set of numbers. For a given stride $s^{[l]}$ and padding p^l(usually set to zero in pooling layers), $n^{[l]}_H$ and $n^{[l]}_W$ are given by,

$$n^{[l]}_H = \lfloor \frac{n^{[l-1]}_H + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \rfloor \ , \tag{23}$$

and,

$$n^{[l]}_W = \lfloor \frac{n^{[l-1]}_W + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \rfloor \ , \tag{24}$$

respectively. The filter size (f,f) and stride s, are hyperparameters, typically set to (2,2) and s=2 to reduce the image or feature map by a factor 2 in n_H and n_W direction.[7]

7.6.5 Fully Connected Layer

The feature map from the last convolutional or pooling layer is flattened. The one-dimensional array is inserted into a chain of one or several fully connected layers. In each layer every input is connected with a weight to an output. Each input and output are called node. The final number of outputs from the fully connected layers is equal to the number of classes in the classification task. Each output should contain the probability for a certain class.[6] The input vector's dimensions for layer l are defined as $(n^{[l-1]}_H \cdot n^{[l-1]}_W \cdot n^{[l-1]}_C, 1)$. The number of input elements in layer l is,

$$n_{l-1} = n_H^{[l-1]} \cdot n_W^{[l-1]} \cdot n_C^{[l-1]} \ . \tag{25}$$

The output from node j in layer l,

$$a_j^{[l]} = \psi^{[l]}(z_j^{[l]}) = \psi^{[l]}\left(b_j^{[l]} + \sum_{i=1}^{n_{l-1}} w_{j,i}^{[l]} a_i^{[l-1]}\right) \ , \tag{26}$$

where $a_i^{[l-1]}$ is an element, i, from the flattened one-dimensional vector from the previous layer, for example, pooling, convolutional, or fully connected layer. The activation function $\psi^{[l]}(z_j^{[l]})$ for the last fully connected layer should be defined to give the probability for a certain class. The number of learned parameters from layer l is [7],

$$\#w + \#b = n_{l-1} \cdot n_l + n_l \ . \tag{27}$$

7.6.6 Training a CNN Algorithm

Supervised learning is a frequently used method to train a CNN algorithm. The method maps CNN input data, samples, to the ground truth, labels. The trained model can in turn predict on unseen input data. In order to adjust the model parameters according to the samples and labels, the forward propagation, mentioned in section 7.6, is followed by backpropagation. Namely finding the gradients of a loss function with respect to the model parameters. Sequentially the model parameters are updated with a variant of gradient descent (GD). The loss function quantifies the compatibility between the prediction from the forward propagation and the ground truth and is a hyperparameter.[6] A commonly used loss function is the mean squared error (MSE),see figure 6,

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \ , \tag{28}$$

where y_i is the output from the CNN, prediction, for a given sample and $\hat{y}_i$ is the associated label. If N is equal to the number of sample and label pairs in the training set, the model parameters are updated with GD. Mini batch stochastic gradient descent (SGD) is a form of gradient descent, where $1 < N <$the size of the training data set. N is called batch size if the algorithm is trained with SGD. In plain SGD the model parameters are updated for a label and sample pair, meaning N=1.

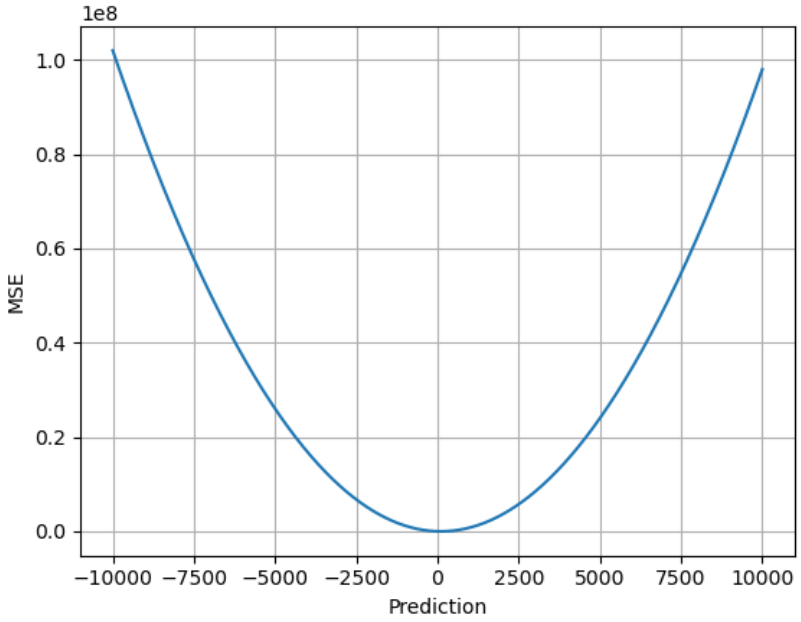

Figure 6: MSE for SGD with true value 100 and prediction in range (-10 000,10 0000).

MSE is sensitive to outliers since a large difference between the prediction and the label is squared. However, machine learning aims for a generalized model that performs well on average, and the MSE is not ideal since the function can put a lot of emphasis on a single outlier. Unlike the mean absolute error (MAE), see figure 7, which for GD is defined as,

$$\text{MAE} = \frac{1}{\text{N}} \sum_{\text{i}=1}^{\text{N}} |\text{y}_\text{i} - \hat{\text{y}}_\text{i}| \ , \tag{29}$$

MAE is not squared, therefore more robust against outliers than MSE. The issue with MAE is the constant gradient, which will be large even for small errors. Therefore more prone to overshoot the minima of the loss function, the ideal solution for the objective function. Unlike MSE, which has a dynamic gradient that decreases with the error. Besides the gradient of MAE is not continuous for all real values.[15]

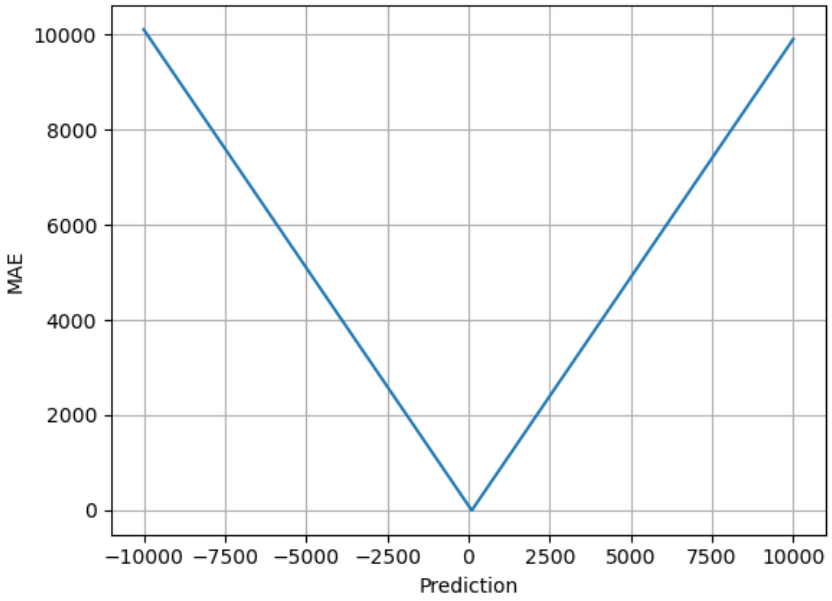

Figure 7: MAE for SGD with true value 100 and prediction in range (-10 000,10 0000).

The image quality metric the mean structural similarity index (MSSIM) has been used as a loss function. The MSSIM is estimated accordingly,

$$\mathrm{MSSIM} = \frac{1}{\mathrm{NM}} \sum_{\mathrm{i}=0}^{\mathrm{N}-1} \sum_{\mathrm{j}=0}^{\mathrm{M}-1} (\mathrm{SSIM}_{\mathrm{i,j}}) \ , \tag{30}$$

where the simplified version of SSIM is defined as,

$$\mathrm{SSIM}_{\mathrm{i,j}} = \frac{(2\mu_{\mathrm{a}}\mu_{\mathrm{b}} + \mathrm{C}_1)}{(\mu_{\mathrm{a}}^2 + \mu_{\mathrm{b}}^2 + \mathrm{C}_1)} \cdot \frac{(2\sigma_{\mathrm{a,b}} + \mathrm{C}_2)}{\sigma_{\mathrm{a}}^2 + \sigma_{\mathrm{b}}^2 + \mathrm{C}_2} = \mathrm{I}_{\mathrm{i,j}} \cdot \frac{(2\sigma_{\mathrm{a,b}} + \mathrm{C}_2)}{\sigma_{\mathrm{a}}^2 + \sigma_{\mathrm{b}}^2 + \mathrm{C}_2} \ , \tag{31}$$

C_1 and C_2 are constants. a and b are windows centered at indices (i,j) in the sample image x and label image y, respectively. μ, σ denotes the average and the variance, while $\sigma_{\mathrm{a,b}}$ denotes the covariance of a and b. $\mathrm{I}_{\mathrm{i,j}}$ is called the luminance component.[16] According to Nilsson and Akenine-Möller[17] the SSIM does not cohere to the characteristic of a human visual system. The luminance component is related to the normalized root mean squared error (NRMSE). Horé and Ziou[18] have found a relationship between MSE and SSIM, and MSE is not considered a perception-based metric in contrast to SSIM. Dosselmann and Yang[19] have found a relationship between SSIM, MSE, and PSNR (peak signal to noise ratio). Nilsson and Akenine-Möller[17] state that SSIM can give a nonintuitive result particularly when

pixel values differ little. Nilsson and Akenine-Möller[17] further mention that denoising and reconstruction algorithms often introduce small variations, and can therefore suffer from SSIM. SSIM should be employed with caution otherwise replaced. The metric can both distort and bias image quality evaluation.

The purpose of backpropagation is to find the partial derivative of the loss function for each learnable parameter. A learnable parameter is updated accordingly,

$$\vartheta_l := \vartheta_l - \eta_l \frac{\partial \mathrm{L}}{\partial \vartheta_l} \ , \tag{32}$$

where ϑ_l is a learnable parameter, L is the loss function and η_l is a hyperparameter called the learning rate. A CNN can consist of several layers, and for the early layers, the chain rule has to be applied multiple times to retrieve the partial derivative.[6] Consequently, the model parameters in the early layers are not updated to the same degree as the latter layers, due to a vanishing gradient from the chain rule. The algorithm is trained in several epochs. One epoch denotes when the algorithm has updated the weights once based on the training data also called the training set.

The validation set is used to evaluate the performance of the algorithm after each epoch. The validation set contains sample and label pairs not part of the training data. The loss function value is calculated for the validation set after each epoch to monitor the performance of the algorithm during training. Often used to optimize hyperparameters such as the number of epochs.

A test set containing sample and label pairs not part of the training or validation set is used to evaluate the performance of the trained algorithm (the model).

7.7 UNET

The UNET is a fully convolutional network (FCN) and does not contain fully connected layers compared to traditional CNNs. Hence, Traditional CNNs are used to map sample images to different categories/classes (labels), and fully convolutional networks map sample images to desired output images (labels). The UNET is U-shaped and consists of an encoder and decoder part. The encoder also called the contracting path, due to the repeated pattern of down-sampling (pooling layers) and convolutional layers, see figure 8. The down-sampling is meant to remove irrelevant details, inevitably significant information is also lost.[20] Milletari et al.[21] segmented Magnetic Resonance (MR) 3D prostate volumes found that replacing pooling layers with convolutional layers with stride 2 preserved details and reduced image size. The decoder is the expanding path with the series up-sampling and convolutional layers. Liu et al.[20] mention that the up-sampling operations are used to spatially locate the pixels and restore image size. Concatenation layers between the encoder and decoder,

where the image size is constant, can recover details lost in down-sampling. The concatenation layer concatenates the feature maps in the encoder with the feature maps in the decoder before the volume of feature maps is passed on to the consecutive layer. Besides, Milletari et al.[21] showed that the concatenation layer shortens the convergence time. The convergence time is the training time the algorithm requires for the validation loss or the accuracy after each epoch to converge.

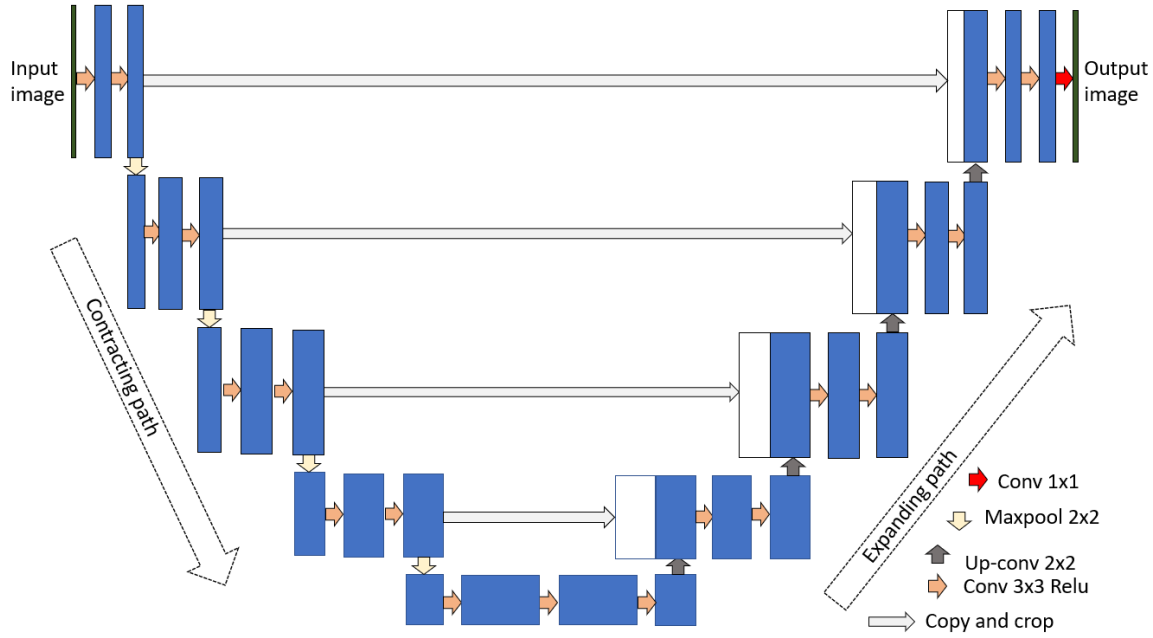

Figure 8: The UNET architecture.

7.8 Batch Normalization

The UNET can be combined with batch normalization.[20] Batch normalization was proposed by Ioffe and Szegedy[22], who trained batch normalized classification networks with training data from ImageNet. The validation error improved compared to earlier published results. According to Ioffe and Szegedy[22] the batch normalization was supposed to tackle the internal covariate shift. The internal covariate shift refers to the inputs of each layer being modified when the model parameters are updated in the previous layers. The algorithm converges slowly partly due to a required lower learning rate since larger learning rates can result in vanishing or exploding gradients. The exploding gradients can in one update have a huge impact on the model parameters. Pykes[23] states that exploding gradient makes the training unstable, for example, the loss function value will change drastically between consecutive epochs. Ioffe and Szegedy[22] found that batch normalization enables larger learning rates. Generally, larger learning rates increase the scale of the model parameters, which can amplify gradients to exploding gradients in the backpropagation. The backpropaga-

tion with batch normalization is uninfluenced by the scale of the model parameters, and can therefore stabilize the growth of the model parameters. The batch normalization layer in Keras, a machine learning framework in the programming language python, is based on the batch normalization suggested by Ioffe and Szegedy[22]. A batch normalization layer in Keras normalizes the batch of inputs with the mean and standard deviation, and is defined as,

$$\frac{\text{batch} - \text{mean(batch)}}{\text{var(batch} + \varepsilon) \cdot \gamma + \beta} \ , \tag{33}$$

where var denotes the standard deviation, epsilon is a constant set to 0.001 by default, γ is scaling factor initially 1 but modified during training and β is an offset factor also modified during training and initially 0.[24] However more recent research implies that the effectiveness of batch normalization is not due to tackling the internal covariate shift but other factors.[25]

7.9 Overfitting

The issue with supervised machine learning is overfitting. Meaning the model does not generalize from training data to unseen data. In contrast to unseen data, the model can accurately predict on training data. The model has memorized features specific to the training data, for example, unavoidable noise. However, the information is irrelevant to the ulterior task the model is given. 1) Overfitting is caused by training data to small, unrepresentative, or containing several noise sources. A generalized model should be able to separate noise from the underlying features. 2) Overfitting can rise from the compromise between variance and bias. Decreasing the number of learnable parameters generally decreases the variance and increases the bias. Hence, less variance between predictions but bias toward inaccurate predictions. Unlike increasing the number of learnable parameters, which has the opposite effect. Increasing the probability for underfitting and overfitting, respectively. Underfitting means the number of features recognized is insufficient to learn from the dataset. 3) The algorithm might have generalized better in the earlier iterations (epoch) compared to the last iteration.[26]

Overfitting can not be completely avoided. Although certain strategies can suppress overfitting, for example, 1) optimizing the number of learnable parameters. The strategy can be time-consuming compared to 2) early stopping, which can reduce the training time. Early stopping involves stopping the iteration at a point when the algorithm stops improving. After a certain epoch, the algorithm will stop improving or perform worse compared to the previous. The latter due to noise learning. If the iteration stops before or after the mentioned point can be a sign of under- and overfitting, respectively. The algorithm is evaluated with regard to the loss function value of a validation set after each epoch. The validation set consists of data from the

training set, which the algorithm is not trained with. 3) A more complex algorithm needs more training data than a simpler model to avoid overfitting. Training data can be difficult and expensive to acquire. However, training data can be expanded with data augmentation, which also improves the trained model's generalization. Data augmentation is simply to manipulate the existing data to create more training data. 4) Data cleaning: filter out less meaning full and irrelevant data in the training set. 5) Regularization: a complicated model memorizes several features even noise. Some of the features are irrelevant to the ulterior task. Different regularization methods exist to remove useless features or minimize the impact of the features.[26] Except for weight penalties, dropout is a frequently used regularization technique. According to Park and Kwak[27] dropout is an effective regularization method for convolutional layers. Dropout in convolutional layers drops activations, for example, the maximum value of each feature map. High activation values often contain details or key information. A stochastic dropout method drops randomly a predefined portion of the activations in each epoch.

7.10 Bayesian Optimization to Optimize Hyperparameters

A global optimization problem is finding the inputs that minimize or maximizes the cost of an objective function, the output. The objective function is often complex, non-convex, form unknown, nonlinear, has several dimensions, and is computationally expensive to evaluate. In machine learning, Bayesian optimization can find a set of hyperparameters that minimizes a cost function, for example, the validation loss. The validation loss is the loss function value of a validation set consisting of sample and label pairs excluded from the training set.[28] The hyperparameters of a machine learning algorithm are denoted $\vartheta_1, ..., \vartheta_n$, and the respective domains $\Theta_1, ..., \Theta_n$. Bayesian optimization algorithm initially evaluates the objective function f at t sets of hyperparameters, the acquired ⟨input,output⟩ pairs $\langle \vartheta_i, f(\vartheta_i) \rangle_{i=1}^{t}$. Called random iterations, since the t sets of hyperparameters are randomly sampled. Subsequently iterating over three steps.

- Step 1: A probabilistic model M of the objective function is created with $\langle \vartheta_i, f(\vartheta_i) \rangle_{i=1}^{t}$, called surrogate function.
- Step 2: Arbitrary inputs $\vartheta \in \Theta$ are evaluated with the surrogate function. The response is interpreted with an acquisition function, which selects the most promising set of hyperparameters ϑ_{t+1}.
- Step 3: The sampled set of hyperparameters ϑ_{t+1} are evaluated with the real objective function. $\langle \vartheta_{t+1}, f(\vartheta_{t+1}) \rangle$ are added to the acquired ⟨input,output⟩ pairs, used to access the probabilistic model.

The number of iterations governed by the acquisition function can be predefined or repeated until an extrema of the real objective function is found.[29]

7.11 SIMIND

SIMIND simulates a clinical SPECT scintillation camera. The program is based on uniformly distributed random numbers to model the radiation transport. The phantom in SIMIND consists of two volumes a source volume containing the radionuclides and the phantom volume where interactions occur. The source and phantom volume can be the same volume. A photon history is created by sampling a decay position in the source volume. The photon history is composed of a primary photon history and scattered photon histories. In the primary photon history, the photon penetrates the phantom and continues toward the collimator without interaction. The direction of the photon is within the collimator acceptance angle. The photon is terminated in the crystal. The probability for the simulated photon path is multiplied with the probability for the photon to escape the camera without interaction. At the same decay position, an isotropic direction is sampled for another photon creating a scattered photon history. A path length is sampled for the scattered photon. If the path length is within the phantom, the photon at the end of the path is Compton scattered toward the camera and escapes without further interactions in the phantom. The photon is followed till termination. At the end of the path a scattered photon history is created, where the interaction type Compton or coherent scattering is accurately sampled. The procedure is repeated until the selected number of scatter orders is reached. Meaning a path length is sampled for the scattered photon. A large number of photon histories are required to simulate a SPECT projection with a reasonable statistical error. The SIMIND program is coupled to the CHANGE program, where camera parameters such as collimator width, pixel width, etc. are defined.[30]

7.12 Virtual Phantoms

The imaging modality can be simulated with computer-based programs to improve and evaluate new imaging techniques. The computer-based programs require a model of the patient's anatomy and physiology, called phantoms. The phantom will provide the truth for a simulation since the organ volumes and tumor sizes are known.[31]

There exist three types of phantoms voxelized, mathematical, and hybrid. Voxelized phantoms are anatomically realistic since the phantoms are based on segmented patient data. However, the phantom is fixed to a patient's anatomy. The voxelized phantoms are not ideal for making anatomical changes to represent a population or modeling organ motion. In mathematical phantoms, anatomical objects are represented by equations or geometric primitives.[31] Geometric primitives are, for example, cuboids, cylinders, prisms, pyramids, spheres, cones, and ellipsoids. A lung can be constructed of half an ellipsoid with a section cut out. Unlike voxelized phantoms, equations can be easily modified to simulate different organ sizes or organ motion.[32] However, equations and geometric primitives can not imitate human anatomy realistically. Hybrid phantoms are a combination of voxelized phantoms and mathematical

phantoms. Surface contours for anatomical objects are segmented from patient data and converted to Non-uniform rational B-spline (NURBS) surface or polygon mesh. Both the surface and the mesh can accurately describe the anatomical objects while maintaining flexibility to anatomical variations. Latter, through the control points and the former through vertex points, that define the anatomical objects.[31]

XCAT is a program that provides virtual patients for research. XCAT has a detailed female and male anatomy, with several defined structures, each converted to a NURBS surface. The anatomies have parameterized models for motion, such as heart and respiratory. The software contains several changeable parameters to define motion, abnormalities, or variations to simulate a patient population. The finished phantom can be converted to a voxelized phantom. Other anatomies based on CT data are available. CT data have been acquired from patients from an age of 8 weeks to 78 years, with varying height and weight percentiles.[33]

7.13 Related Work

Denosing nuclearmedicine images with FCN (fully convolutional network) have previously been performed by Minarik et al.[34] and Xu et al.[16], and are summerized in table 1.

Table 1: Related work from David et al.[34] and Xu et al.[16] summarized.

	David et al.[34]	Xu et al.[16]
Task	Denoising	Denosing
Modality	Gamma camera	PET
Training data	Monte Carlo simulated planar projections	Reconstructed multi slices
Architecture	FCN	UNET
Loss function	quadratic	MAE
Evaluation metric	Mean MSE	NRMSE, PSNR and SSIM

Minarik et al.[34] simulated 99m-Tc anterior and posterior whole-body projections (256x1024) with virtual phantoms from XCAT and the virtual gamma camera SIMIND. The training material consisted of sample and label pairs, constructed of patches (128x128) from the anterior och posterior whole-body bone scans. Minarik et al.[34] trained FCN models with additional two training sets, one consisting of anterior and posterior projections of cylindric phantoms and a training set combining the mentioned training sets. According to Minarik et al.[34] the FCN trained on bone

scans and cylindrical phantoms were on average 20% better than Gaussian-filtered bone scans with to the mean squared error as an evaluation metric. The FCN most suited for the cylindrical phantom and the bone scans were the FCN trained with cylindrical phantoms and bone scans, respectively. Sample, label, and predictions acquired from the different FCN by Minark et al.[34] are shown in figure 9.

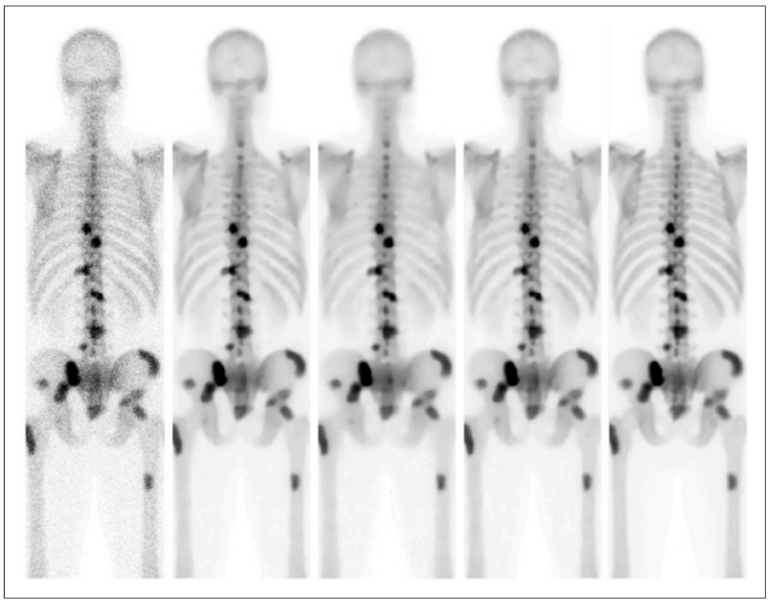

Figure 9: The posterior projections by Minarik et al. [34]. From left to right: sample, prediction from FCN trained with bone scans, prediction from CNN trained with cylinder projections, prediction from CNN trained with bone scans and cylinder projections, and the label. *Image originally published in The Journal of Nuclear Medicine (JNM) by Minark et al.[34] and is reprinted according to CC by NC.*

Xu et al.[16] trained a UNET to reconstruct standard-dose PET images from ultra-low-dose PET images. The training material was based on nine patients with glioblastoma, who had undergone PET/MRI scans with standard doses. From the standard dose images, noisy ultra-low-dose images were acquired by randomly and uniformly selecting 0.5% of the counts. Both the standard dose and the ultra-low dose images were reconstructed to 3D volumes. Apart from the standard UNET components, the structure consists of a residual connection, meaning the input is convolved with a filter of size (f,f,n_C)=(1,1,n_C) and added with the output image from the UNET, see figure 10. The input is three-dimensional and represents multiple slices instead of color channels. With volumetric information, consistent structures can be distinguished from noise. Xu et al.[16] used the normalized root mean squared error (NRMSE), the peak signal to noise ratio (PSNR), and the structural similarity index (SSIM) as evaluation metrics on test slices. The proposed UNET architecture was compared

with other FCN architectures. Xu et al.[16] found that the proposed architecture had superior performance, with regards to preserving resolution and reducing noise.

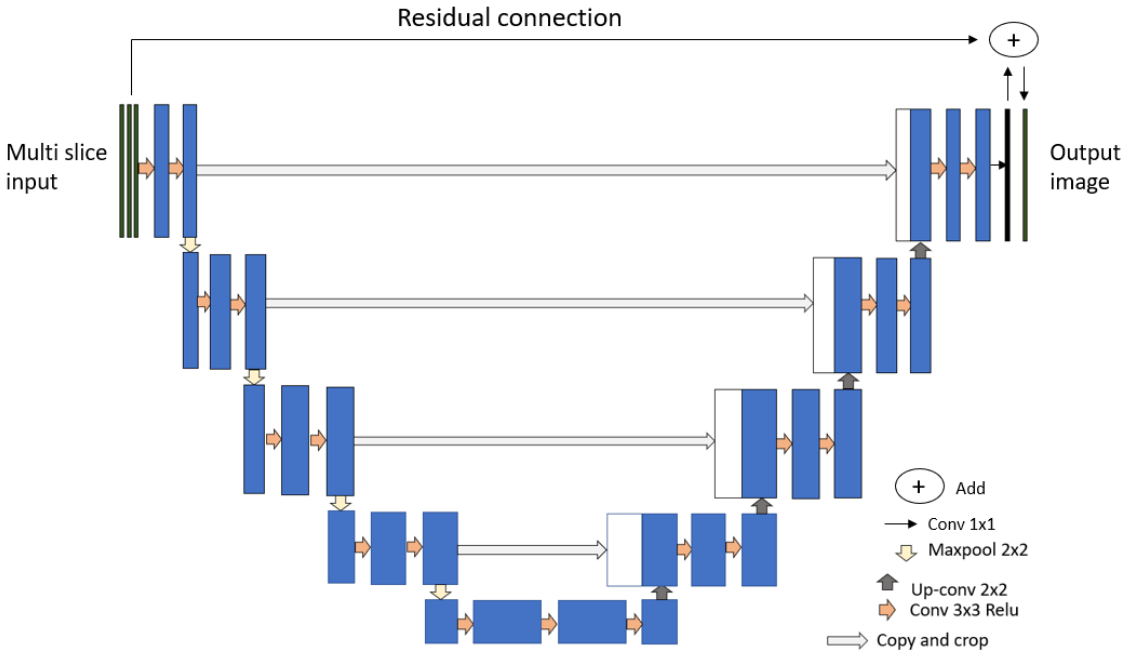

Figure 10: UNET with a residual connection and multi-slice input.

8 Method

8.1 Creating Phantoms in XCAT and Simulating Anterior Gamma Camera Projections in SIMIND

Sixteen base phantoms in XCAT were used, eight female and eight male. The base phantoms were altered with the XCAT program to create new phantoms contributing to a larger and more diverse training data set to avoid overfitting. The XCAT program was used to scale each base phantom with scale factors in intervals with an increment of 0.01. The upper and lower interval values were set individually, since the liver volume and height were characteristic to a base phantom, see table 9 in the appendix. The base phantoms were scaled with the same scale factor in height, long axis, and short axis. After scaling, the shortest, tallest. the largest liver volume and smallest liver volume were 150.0 cm, 192.7 cm, 2172 ml, and 859 ml, and were made from base phantoms femalepat_86, vmale_50, malepat_118, and femalepat_71 scaled with a scale factor of 0.89, 1.10 and 1.10, 0.94, respectively. From the scaling operation of the base phantoms, 232 scaled phantoms were created. Each of the created phantoms was copied three times, resulting in 696 phantoms. Random values of all the parameters listed in the table 10 were assigned to each of the 696 phantoms since copies were not of interest. The randomly modified phantoms were checked for multiples. Each modified phantom was constructed of 420 slices with a slice width of 0.15 cm covering parts of the thorax pelvis region. The slices had the dimensions 400x400 pixels with a pixel width of 0.15 cm.

For each phantom, ten anterior gamma-camera projections with labeled ^{111}In were simulated with SIMIND windows version 6.2 64 bits. The ten simulations were based on the estimated activity concentration in organs of five patients injected with labeled ^{111}In, at time points 4.4 h and 24 h after injection. Hence the ten simulated images acquired for each phantom were not copies. The projections were simulated with an acquisition time of 30 seconds. The simulations were performed with SIMIND windows version 6.2 64 bits and set up to mimic a standard Siemens camera. The camera had a 0.95 cm thick NaI crystal, an intrinsic spatial resolution of 0.37 cm, a medium energy collimator, an energy window of 20% centered around 245 keV, and an energy resolution in percentage full width at half maximum (FWHM) set to 9.5%. The simulated images had the dimension 128x128 pixels and a pixel size of 0.48 cm. A total of 6960 simulated images were obtained.

8.2 Applying Augmentations and Creating Sample and Label Sets

A larger data set is simulated by creating augmented copies to avoid overfitting. For each of the 6960 simulated images, three augmented copies were made besides

the original image, one with rotation, one with horizontal shift, and the third with rotation and shift. Hence, a total of 27840 images were obtained. The rotation and shift were randomly chosen to be an integer between -7 to 7 degrees and -20 to 20 pixels. Seeing that the original images were included and accounted for 0-pixel shift and 0-degree rotation, the rotated copy was not allowed to have zero degrees rotation, the shifted copy could not have zero pixel shift and the shifted and rotated image could have neither. Each of the 27840 images was scaled with two different scale factors randomly chosen from 0.1, .25, 0.5, 0.75, and 1.0. However, a copy of the unscaled images was set aside to form the labels for the training of the AI model. The samples were created by adding Poisson distributed noise to the scaled images. Each sample had a corresponding label. All samples and labels were normalized to the maximum pixel value in each image. Since the sigmoid function was used as the last layer activation function and the relu activation function in every other convolutional layer, all values ≤ 0.01 and $0.99 \leq$ were set to 0.01 and 0.99, respectively. Setting the pixel values to a maximum 0.99, is essential if the sigmoid function is used as the last layer activation function. The sigmoid function cannot return 1.00, due to its' converging property. Setting the pixel values to a minimum 0.01 is essential for the relu activation function since for pixel values ≤ 0, the gradient is zero and preventing weight updates.

8.3 Training UNETs with Bayesian Optimization

A UNET was programmed with Keras, a machine learning framework in python. The ordered layers of the UNET architecture are listed in table 11 in the appendix. The python library bayes_opt was applied and is based on the Bayesian optimization algorithm with gaussian processes. The objective function was to minimize the loss function value for the validation set, 10% of the sample, and label pairs in the training set. The UNET was not trained with the validation set, only evaluated. Fifty-five iterations were performed with the optimization algorithm, resulting in 55 models trained on different sets of hyperparameters. The iterations included 15 random iterations and 40 iterations governed by the Bayesian acquisition function. Between the iterations, the hyperparameters: learning rate Lr, the number of filters F, the dropout rate D, the number of epochs E, and the loss functions Lf were modified. The parameters' bounding values are listed in table 2. The bayes_opt sampled a float of the listed hyperparameters. Therefore, the models were trained on the integer of the sampled epochs. By default, Keras took the integer of the number of filters. The sampled loss function value was rounded to either 1 or 0. Hyperparameters fixed for all iterations was the stochastic gradient descent with a batch size of 20 sample and label pairs. The callback tf.Keras.callbacks.ReduceLROnPlateau reduced the learning rate with a factor 0.5 after five epochs if the validation loss function was not reduced between epochs. The tf.Keras.callbacks.EarlyStopping, stopped the model training if the loss function value for the validation set did not reduce in 10 epochs.

The model with the smallest loss function value for the validation set out of the 55 models was evaluated.

Table 2: Optimization intervals for the hyperparameters learning rate LR, the number of filter F, the dropout rate D, the number of epochs E and the loss function Lf.

Hyperparameter	Interval	Description
Lr	[0.001,0.05]	-
F	[8,64]	-
D	[0.0,0.5]	-
E	[25,300]	-
Lf	[0,1]	0 is mse and 1 is ssim.

8.4 Different Kinds of UNETs

The kernel initializer is a hyperparameter, although inserting it as a parameter in the Bayesian optimization algorithm was not solved. The kernel initializer was manually changed. Therefore 55 UNET models were trained for each kernel initializer or combination of initializers examined. The procedure in section 8.3 was repeated for the kernel initializer truncated normal and the combination of kernel initializers he normal and glorot normal. The truncated normal had a mean at 0 and the standard deviation at 0.05. He normal and glorot normal was applied on the convolutional layers with the ReLu and sigmoid activation function, respectively. A batch normalization layer was added manually after each convolutional layer with the relu activation function, see table 11. Hence, further 55 models were trained with batch normalization layers and the he normal and glorot normal as kernel initializers. Another set of 55 models were trained without the batch normalization layers, leaky relu replaced relu, MAE instead of SSIM, and the combination of he uniform and glorot uniform as kernel initializers. α was set to a hyperparameter in the domain (0.1,0.3).

8.5 Evaluation Methology

The UNETs with different characteristics were evaluated using ten simulated anterior gamma camera projections that were not part of the training set, called test data. Augmented copies were made for each image, one with rotation, one with horizontal shift, and one with both. Each of the augmented copies was scaled with 0.1,0.24,0.5, 0.75, and 1.0 before Poisson distributed noise was added, resulting in 200 samples. A copy of the unscaled images was kept to form the labels. Hence every label had a corresponding sample.

Similarity metrics normalized root mean square error (NRMSE), peak signal to noise ratio (PSNR), and mean structural similarity index (MSSIM) were used to quanti-

tatively evaluate image quality. The MSSIM and the simplified version of SSIM is defined in equations (30) and (31), respectively. The NRMSE is defined as,

$$\mathrm{NRMSE} = \sqrt{\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(x_{i,j}-y_{i,j})^2}{\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}y_{i,j}^2}} \ , \tag{34}$$

where N,M denotes the number of rows and columns in images y and x. Image y is the label, whereas $y_{i,j}$ denotes the pixel intensity at indices (i,j). The same applies to image x, which is the either the sample or the prediction retrieved from the AI. PSNR is given by,

$$\mathrm{PSNR} = 20 \cdot \log_{10}\left(\frac{\mathrm{MAX}}{\sqrt{\mathrm{MSE}}}\right) \ , \tag{35}$$

where MAX denotes the peak intensity of the noise-free label image. MSE is the mean squared error, and can be calculated accordingly,

$$\mathrm{MSE} = \frac{1}{NM}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(x_{i,j}-y_{i,j})^2 \ . \tag{36}$$

NRMSE, PSNR, and MSSIM were calculated for each prediction and the corresponding label as well as each sample label pair. Subsequently, the mean for each similarity index was evaluated for the different UNETs. One-way ANOVA test was used to examine if a statistically significant difference for the means of NRMSE, PSNR, and MSSIM existed between the different UNETs for predictions with scaling and without rotation and shifting. A one-way ANOVA test was performed for each scale factor, namely 0.1,0.25,0.5,0.75 and 1.0 and each evaluation metric. The null hypothesis, H_0, and the alternative hypothesis, H_1, are defined as,

H_0: No statistically significant difference between the means of the different UNETs for a certain evaluation metric and scale factor.

H_1: A statistically significant difference between the means of the different UNETs for a certain evaluation metric and scale factor.

The significance level was set to 0.05.

8.6 Comparing UNET to a Butterworth Filter

An ideal filter redistributes the pixel values, while the image total is retained. The image total refers to the sum of all the pixel values in an image. The UNETs in section 8.4 were trained on samples and labels with image totals that differed, due to the normalization and the Poisson distributed noise added to the samples. The

UNET models in section 8.4 were not trained to preserve the image total. The image total of a sample and its associated label with a scale factor of 1.0 from the test set used in section 8.4, are listed in table 3.

Table 3: The image total of a sample and its associated label with a scale factor of 1.0. The sample and label pair is from the test set used in section 8.4.

Image total the sample	Image total of the label
1590	1857

Except for the models generated in section 8.4, another 35 models were trained, and the sample and label pairs were created according to section 8.2 with four exceptions. Firstly, the labels were scaled. Secondly, the samples nor the labels were normalized. Thirdly, since normalization was excluded the last layer activation function was replaced by relu all values $0.99\leq$ were not set to 0.99. However, all values ≤ 0.01 were set to 0.01. Fourth, the noised samples were scaled with the respective quotient between the total of the sample and label image. The image total of the sample and label were equivalent with the applied changes. Without normalization, the pixel values had the unit kBq / ml. The 35 UNETs were trained according to section 8.3 but with 15 random iteration and 20 iterations governed by the Bayesian acquisition function. The kernel initializer was set to He normal for each convolutional layer with the relu as activation function.

The UNET was compared with a Butterworth filter with $n = 4$ and $D_0 = 15$. The evaluation metrics used were the mean absolute error (MAE), which was calculated accordingly,

$$\mathrm{MAE}(A, B) = \frac{1}{N}\sum_{i=1}^{N}|\mathrm{total}(A_i) - \mathrm{total}(B_i)| \ , \tag{37}$$

where $\mathrm{total}(A_i)$ is image total of a sample, A_i, and $\mathrm{total}(B_i)$ is the associated image total of the filtered image, B_i. N denotes the number of samples and filtered images evaluated. MAE was calculated for the respective scale factor and filtering method. The test data consisted of 200 samples and labels. The MAE for each scale factor is based on 40 samples from the test data and respective filtered images. The 40 samples contain 10 different phantoms, and four copies of each phantom. Namely, a copy with no rotation or shifting, with rotation added, with shifting added, and with both rotation and shifting. Between the different scale factors, the same images were used but the samples were multiplied with different scale factors. The mean of the image total for the 40 samples and labels for each scale factor was listed as a reference.

9 Results

9.1 Evaluating Different Kinds of UNETs

The Bayesian optimization algorithm trained four sets of 55 models, meaning a total of 220 models were trained. The four sets had different characteristics namely, the first set of 55 models were trained with the kernel initializer truncated normal, the second set with he normal in combination with the glorot normal without batch normalization layers, the third the same as the second but with batch normalization layers and the fourth used he uniform in combination with glorot uniform, where ssim and relu were replaced by MAE and leaky relu respectively. The hyperparameters for the best-performed model in each set with regards to the loss function value of the validation set are listed in table 4. The number of filters F and the dropout rate D are compatible with the notation in table 11 in the appendix, which describes the UNET architecture. Inferior performance was received with the SSIM loss function compared to the MSE with regards to the loss function value of the validation set. The best-performed models with SSIM had validation loss function values of 0.02985, 0.01657, and 0.01756 for the truncated normal, the he normal in combination with glorot normal without batch normalization, and the he normal in combination with glorot normal with batch normalization, respectively. Superior performance was acquired with the MSE in the fourth set, whereas the best-performed model with MAE had a validation loss of 0.15997.

Table 4: Hyperparameters learning rate LR, number of filters F, the dropout rate D, the number of epochs E, and the loss function LF for the best-performed model for each set of models trained with the Bayesian optimization algorithm. Listed alongside the loss function value for the validation set after the last epoch.

	1st set	2nd set	3rd set	4th set
Hyperparameter	Truncated normal	He normal and glorot normal	He normal and glorot normal with batchnormalization layers	He uniform, glorot uniform, MAE replaced SSIM and leaky relu replaced relu
Lr [1]	0.025	0.049	0.050	0.007
F [1]	19	39	38	44
D [1]	0.0097	0.0000	0.0000	0.0000
E [1]	51	73	52	29
Lf [1]	MSE	MSE	MSE	MSE
Validation loss after last epoch	0.00087	0.00058	0.00079	0.00091
α	-	-	-	0.101

A plot for each augmentation applied to the samples, therefore indirectly to the predictions, namely scaling and scaling combined with rotation, shifting, and a combination of rotation and shifting is visualized in figures 11, 12 and 13. In each subfigure of figure 11, the mean PSNR of 10 predictions for the models listed in table 4 is plotted against the scale factor applied to the samples. The mean PSNR increases with the scale factor and is independent of the augmentations rotation, shifting, and a combination of both. Moreover saturates with an increasing scale factor.

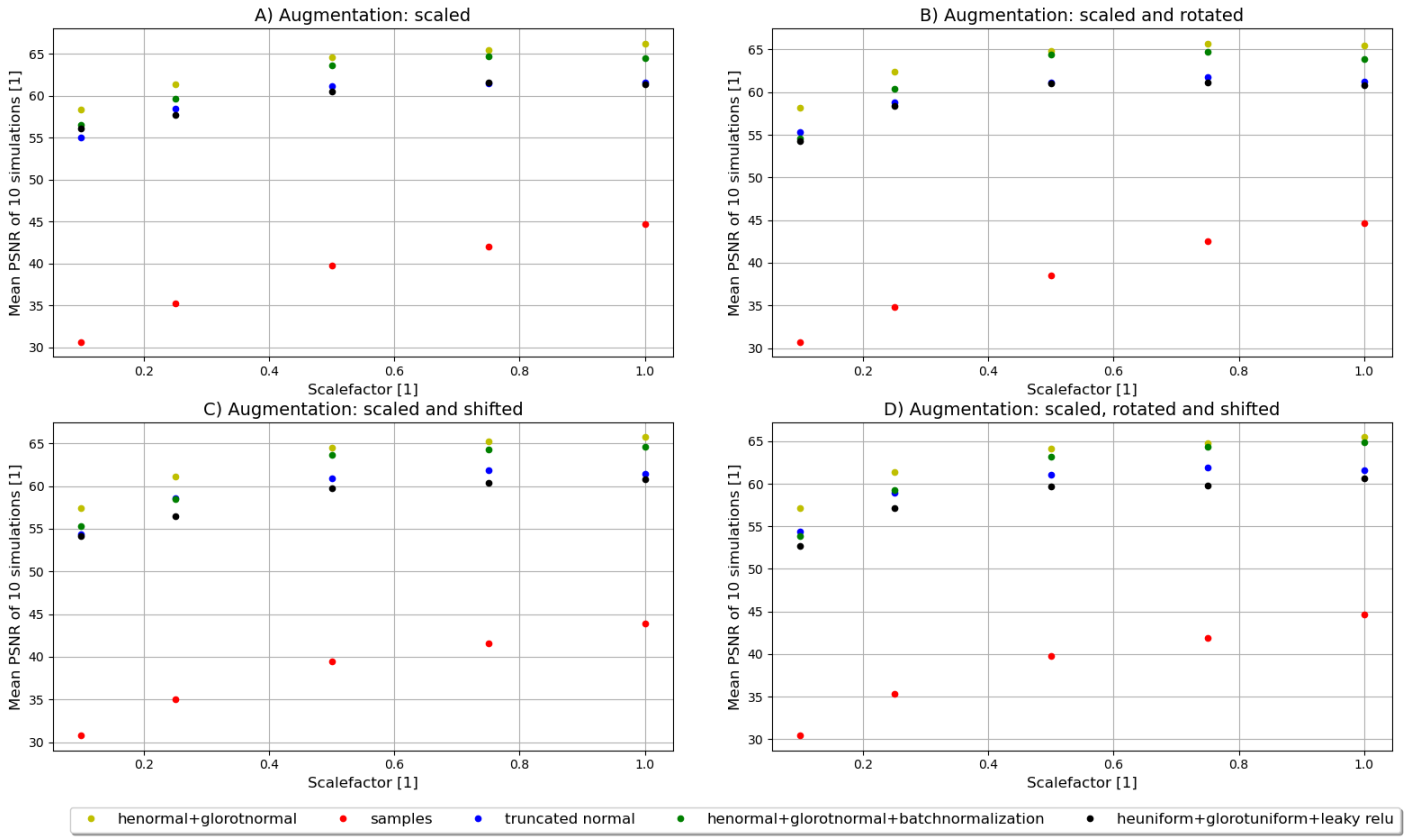

Figure 11: The mean PSNR of 10 predictions for the models listed in table 4 is plotted against the scale factor. In subfigures A-D, no augmentations except scaling were applied to the samples, scaling combined with rotation, scaling combined with shifting, and scaling combined with both rotation and shifting, respectively.

In the subfigures of figure 12, the mean NRMSE of 10 predictions is plotted against the scale factor applied to the samples of the models listed in table 4. The mean NRMSE decreases for increasing scale factor, while also converging. The mean NRMSE is independent of the augmentations rotation, shifting, and a combination of both.

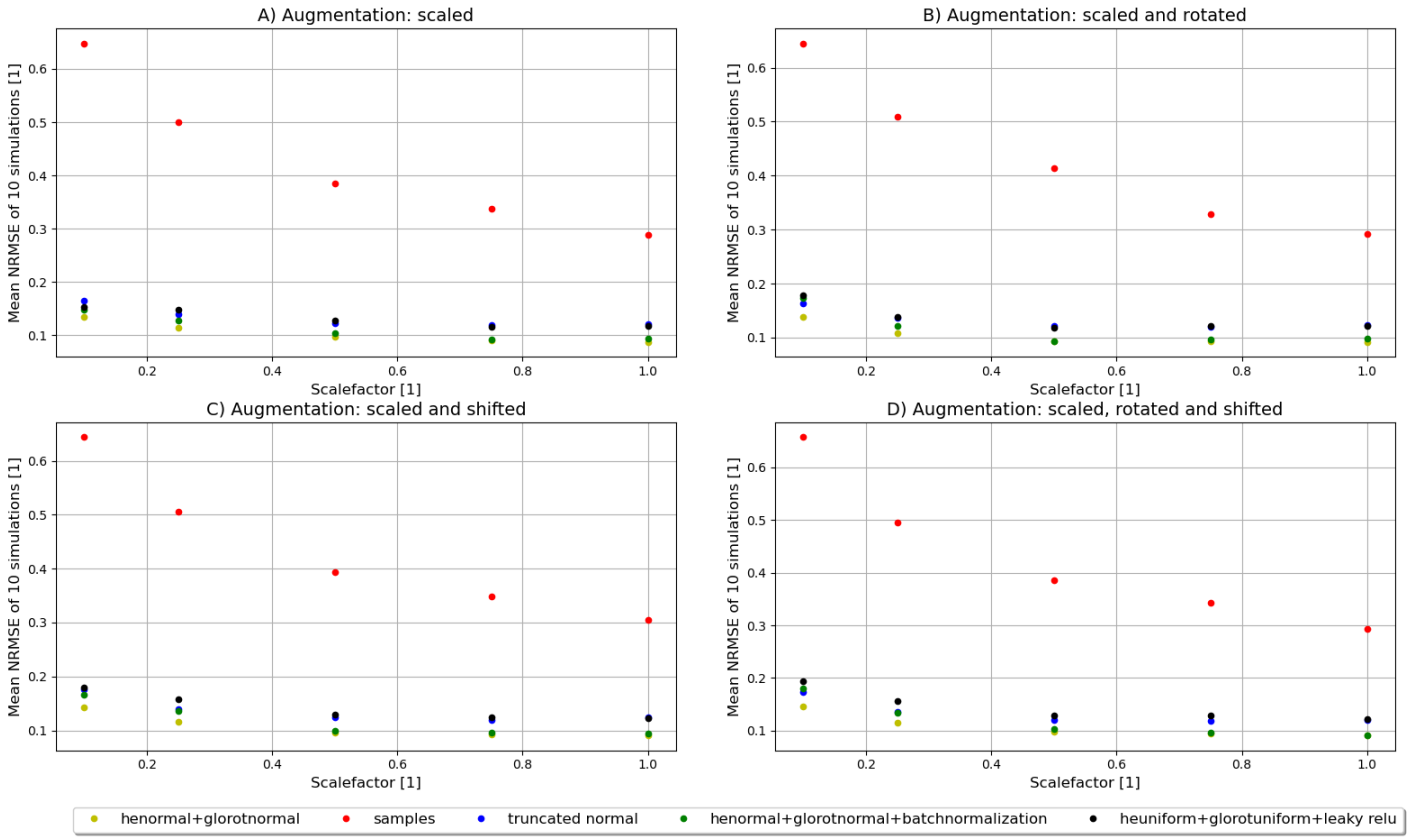

Figure 12: The mean NRMSE of 10 the predictions for the models listed in table 4 is plotted against the scale factor. In subfigures A-D, no augmentations except scaling were applied to the samples, scaling combined with rotation, scaling combined with shifting, and scaling combined with both rotation and shifting, respectively.

In the subfigures of figure 13, the mean MSSIM of 10 predictions for the models listed in table 4 is plotted against the scale factor. The mean MSSIM increases with the scale factor, while also saturating. According to the mean MSSIM, the models are invariant to rotation, shifting, and the combination of both.

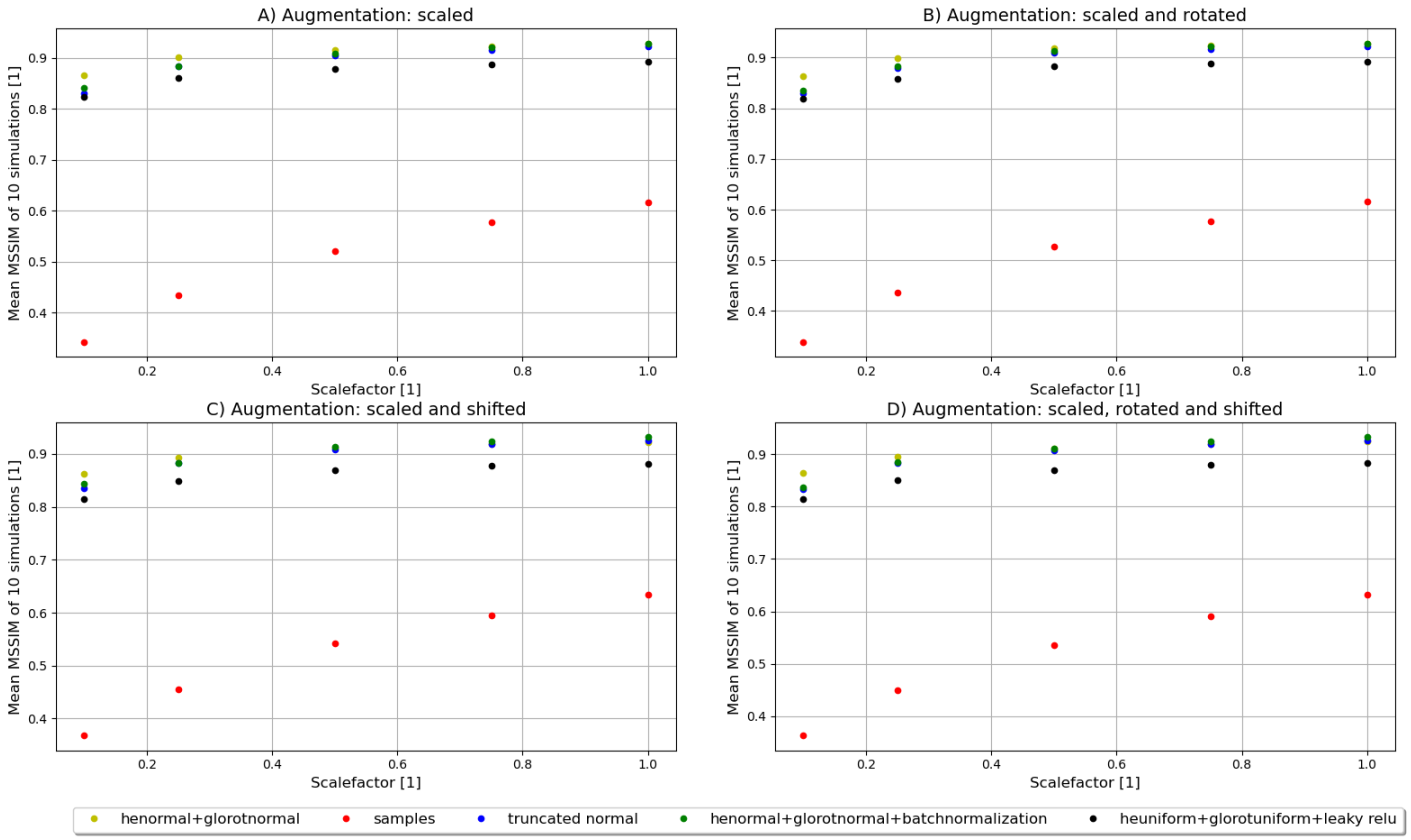

Figure 13: The mean MSSIM of 10 the predictions for the models listed in table 4 is plotted against the scale factor. In subfigures A-D, no augmentations except scaling were applied to the samples, scaling combined with rotation, scaling combined with shifting, and scaling combined with both rotation and shifting, respectively.

In table 5 the P-statistics are listed for each evaluation metric and scale factor are listed. Note the predictions the one-way ANOVA tests were performed on contained no rotation or shifting. The P-statistics are above the significance level for PSNR and NRMSE, while below for the MSSIM for each scale factor.

Table 5: The P-statistics for one-way ANOVA tests of different evaluation metrics and scale factors for predictions without rotation and shifting.

	Scalefactor				
Evaluation method	0.1	0.25	0.5	0.75	1.0
PSNR	0.270	0.264	0.343	0.173	0.113
NRMSE	0.412	0.595	0.597	0.332	0.294
MSSIM	0.026	0.006	0.002	0.000	0.000

In figure 14 a sample and label pair from the test set is shown alongside the predictions from the listed models from table 4. The sample label pair had no shifting or rotation applied. The sample was scaled with 1.0. The prediction from the best-performed model with regards to the validation loss from the 3rd set had an artifact, a white blob in the liver. The artifact occurred only for the model evaluated from the 3rd set and was present in 55/200 predictions from the test set. The artifact consisted of pixel value 1.0. In figure 14 the vertebrae in the spine are visible in the four predictions, which was not the case for all the predictions from the test set.

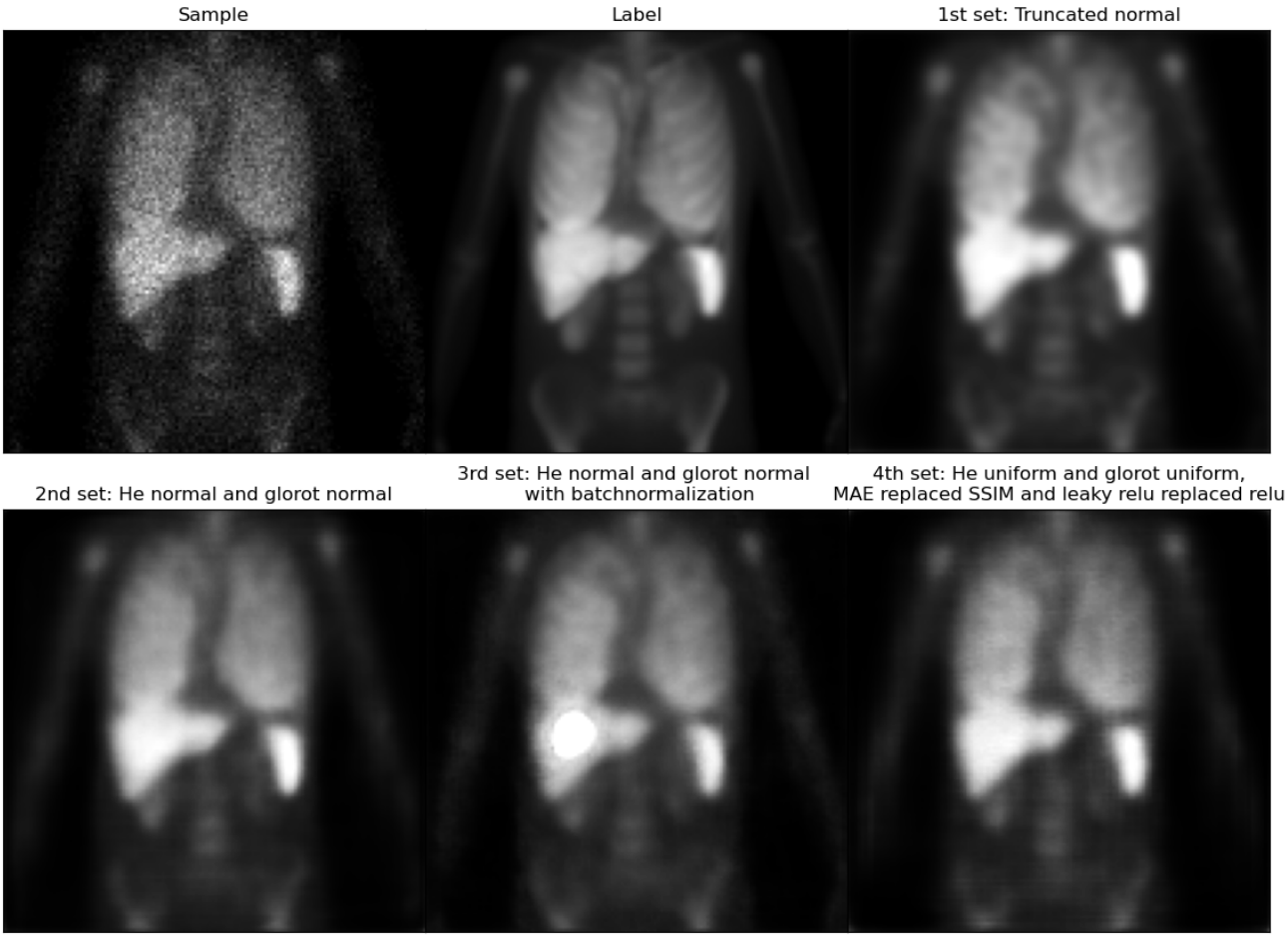

Figure 14: The sample, label, and the predictions from the models listed in table 4. The sample and label pair contained no augmentations such as shifting and rotation. The sample had a scale factor of 1.0.

9.2 Comparing UNET to a Butterworth Filter

The last set of models were trained with non-normalized data and the kernel initializer He normal. The hyperparameters for the best-performed model with regards to the validation loss are listed in table 7. The validation loss after the last epoch was 0.99984, however, a validation loss of 0.21120 was recorded after the second epoch. The model was saved if the validation loss had improved compared to the previous epochs. Therefore the model was saved when the validation error was 0.21120 and not 0.99984. The best-performed model with MSE as loss function had a validation loss of 61.22852.

Table 7: Hyperparameters learning rate LR, number of filters F, the dropout rate D, the number of epochs E, and the loss function LF for the best-performed model for the 5th set of models trained with the Bayesian optimization algorithm. Listed alongside the loss function value for the validation set after the last epoch.

	5th set
Hyperparameter	No normalization on the training data and He normal
Lr [1]	0.014
F [1]	21
D [1]	0.3902
E [1]	13
Lf [1]	SSIM
Validation loss after last epoch	0.99984

Table 8 lists the MAE for the different filtering methods and scale-factors alongside MAE between the samples and labels. Higher and lower values for MAE can be observed for UNET and the Butterworth filter, respectively. MAE increases with the scale factor for both filtering methods.

Table 8: MAE is listed for the different filtering methods and scale factors. Alongside MAE calculated between the samples and labels. The mean of the image total of the samples for different scale factors are also listed.

	Scalefactor				
	0.1	0.25	0.5	0.75	1.0
Mean($\text{tot}_{\text{sample}}$)	13823 ±3122	34511 ±7808	68995 ±15619	103481 ±23429	137969 ±31240
MAE(sample,label)	0 ±0	0 ±0	0 ±0	0 ±0	0±0
MAE(sample,prediction)	360 ±113	718 ±153	1115 ±238	1493 ±354	1875 ±467
MAE(sample,butter)	44 ±15	81 ±35	139 ±67	198 ±92	255 ±126

In figure 15 the UNET filtered and Butterworth filtered images are shown alongside the sample and the label from the test data, both scaled with 1.0. The phantom is also visualized in image 14. Note in figure 15 the label and sample are shifted -6 pixels and rotated 4 degrees. Furthermore, the sample and label pair were scaled and normalization was not performed. 15 includes the normalized profiles at index y = 75 for the label, the Butterworth filtered image and the UNET predicted image. Each profile was normalized against the maximum value in the respective profile. The backbones are more distinguishable in the Butterworth filtered image compared to the UNET prediction. The two larger peaks at index 40 and 80 in the normalized profiles represents the liver and the spleen, respectively. The two smaller peaks are the arms.

In figure 16 the UNET filtered and Butterworth filtered images are shown for a sample scaled with 1.0 and label from the test data created in section 8.5 for the normalized training data. The UNET prediction is from the model listed in table 4 as the 4th set. The kernelinitializer for the model was the He normal and glorot normal. The sample and label are created from the same phantom used in image 14 and 14. Note the sample and label are shited with 20 pixels and rotated with -5 degrees rotation. In 16 the normalized profiles at index y = 80 are plotted for the label, Butterworth filtered image and the AI predicted image. Each profile was normalized against the maximum value in the respective profile.

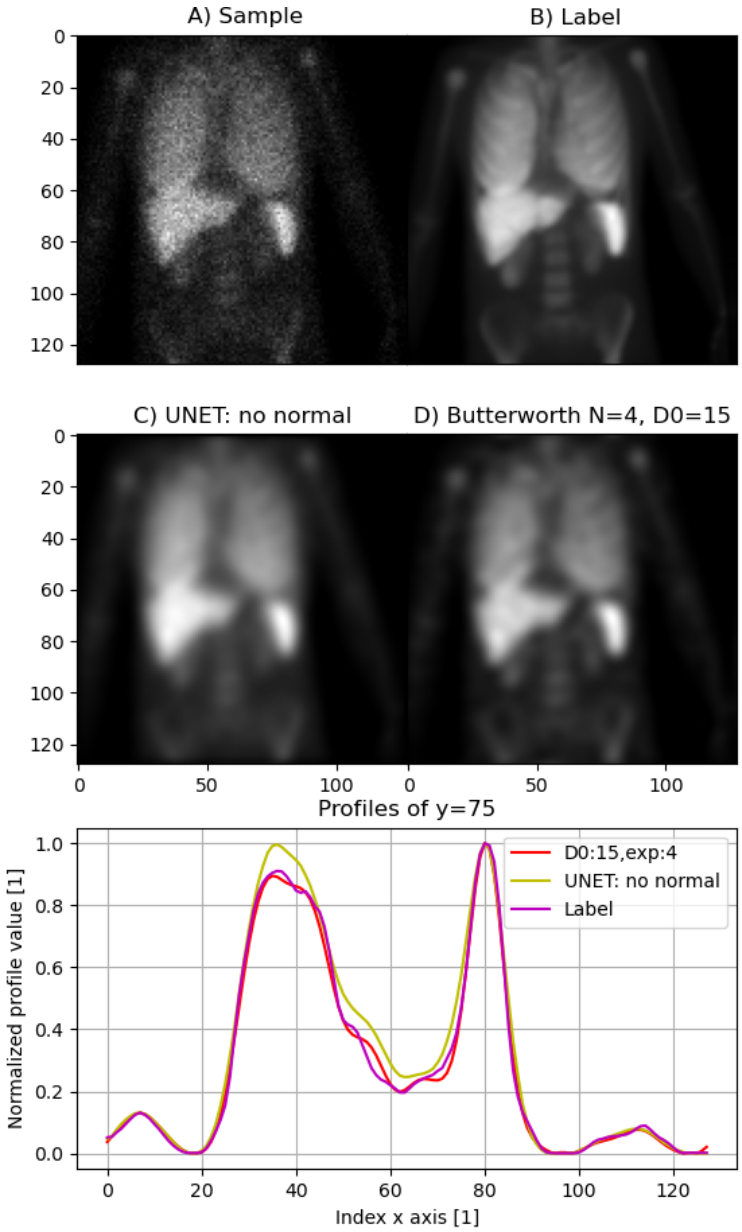

Figure 15: A sample and the respective label from the test data are shown with the UNET filtered and the Butterworth filtered images. The sample and label are scaled with 1.0 and normalization were not performed. The normalized profile values for the label, the UNET prediction, and the Butterworth filter at index y = 75 are plotted against the x-axis index. The profiles were normalized against the max value in each profile.

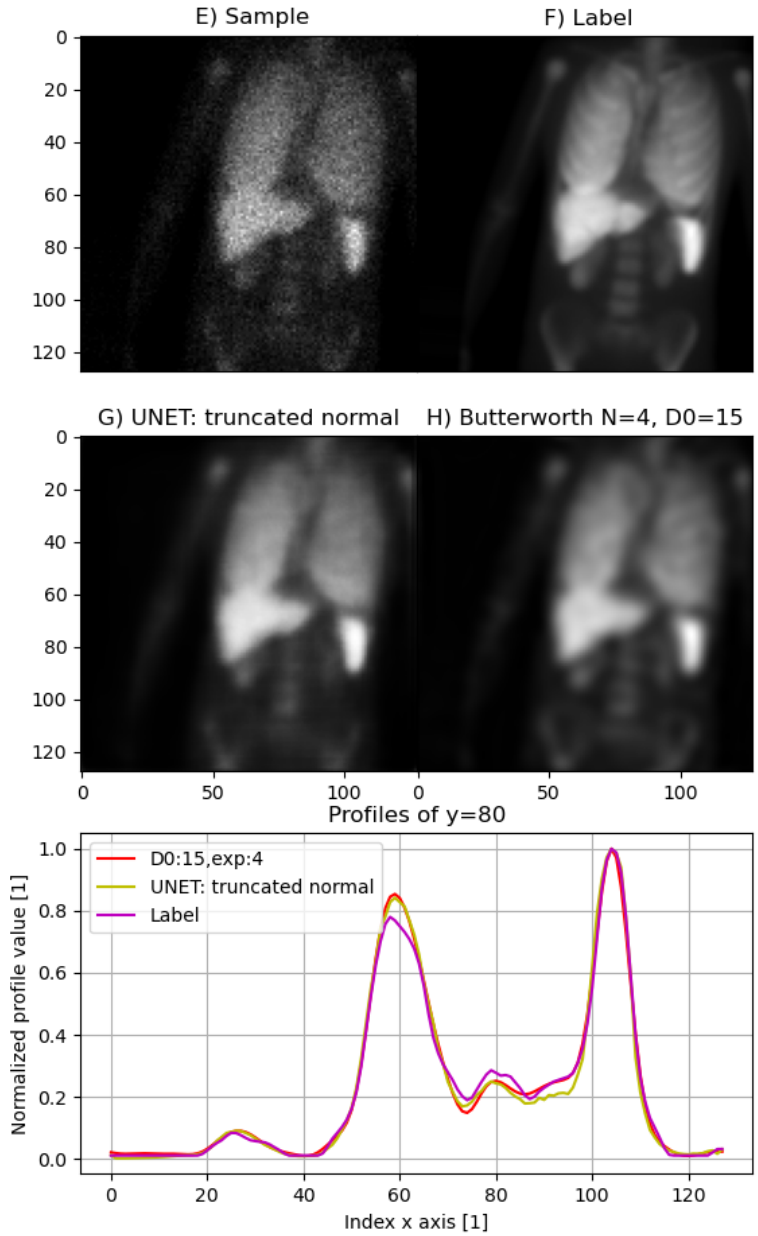

Figure 16: A sample and the respective label from the test data are shown with the UNET filtered and the Butterworth filtered images. The sample is scaled with 1.0 and normalization was performed. The UNET prediction is from the model listed in 4 as the 2nd set. The normalized profile values for the label, the AI prediction, and the Butterworth filter at index y = 75 are plotted against the x-axis index. The profiles were normalized against the max value in each profile.

10 Discussion

10.1 Evaluating Different Kinds of UNETs

In figures 11, 12 and 13 the mean PSNR, mean NRMSE and mean MSSIM does not deviate significantly for the different combination of augmentation for a fixed scale factor and UNET. Indicating that the models evaluated are invariant to the augmentations rotation, shifting, and a combination of rotation and shifting introduced in the training data. The mean PSNR and mean MSSIM increase with the scale factor and the mean NRMSE decreases with the scale factor since a lower scale factor introduces more Poisson noise and represents an image acquired with 30 seconds scaled with the scale factor. The mean PSNR, mean MSSIM, and mean NRMSE saturates with an increasing scale factor, a contributing factor is the high noise level even for a scale factor of 1.0, see figure 14. The mean PSNR, mean NRMSE, and the mean SSIM was used as evaluation metrics instead of the mean MSE, and is consistent and inconsistent with Xu et al.[16] and Minarik et al.[34], respectively. The mean MSE was not used as an evaluation metric, since it is sensitive to outliers, due to the squared term.

The noise level in Minarik et al.[34] was low compared to the noise level acquired with ^{111}In with an acquisition time of 30 seconds in SIMIND. Minarik et al.[34] reduced the noise while preserving the image details, see figure 9. The noise was reduced with different UNETs, see figure 14. However, the UNET could not distinguish, for example, the ribs from the noise. To recover more image details such as the ribs one can try training the UNET with larger training data with a larger spread. For example by creating phantoms from the base phantoms with more variation between the phantoms or by using more base phantoms and patient data to estimate activity concentration in organs. Another option is to use volumetric input data, like Xu et al.[16] used, since structural information can help the algorithm distinguishing noise.

In the one-way ANOVA tests, the evaluation metrics PSNR and NRMSE showed no statistical difference between the means of the different UNETs, since the P-statistics were below the significance level for the listed scale factors, see table 5. With regards to predictions with no rotation and shifting added. The results received with the statistical tests can be misleading, for example in figure 14 a visible difference can be seen between the different UNETs. The shape and size of the spleen differ between the UNETs, also the fact that the liver of the model trained with he normal, glorot normal, and batch normalization has an artifact, a white blob. The one-way ANOVA tests performed on the MSSIM showed a statistically significant difference between the means of the different UNETs, but for which UNET model or models is unknown. The source of the observed difference is unknown, for example, if the difference was due to the different hyperparameters Lr, F, E, Lf, and α between the UNETs or

the manual modification performed between the Bayesian optimization algorithms. Namely changing the kernel initializers and the activation function to leaky relu or adding batch normalization layers. The mean MSSIM was the second-highest for the model evaluated from the 3rd set compared to the other sets for all sub-figures, see figure 13. Not expected, considering that 55/200 predictions had the white blob artifact. However, Nilsson and Akenine-Möller[17] stated that the use of MSSIM as an evaluation metric should be performed with caution since it can introduce bias and distortion in the assessment.

None of the models evaluated had the MSSIM as loss function, which is considered a nonoptimal loss function and evaluation metric according to Nilsson and Akenine-Möller[17]. The MSE was found the most suitable loss function by the Bayesian optimization algorithm from sets 1-3 compared to MSSIM. In the fourth set, the MSE was found superior compared to MAE. The gradient of MSE is dynamic and less likely to overshoot the minima compared to MAE. However, Xu et al.[16] used MAE as the loss function and the architecture proposed was superior compared to the other architectures examined in the article.

The artifact, the white blob, for the 3rd set with he normal, glorot normal, and batch normalization can be due to overfitting, see figure 14. The Bayesian optimization algorithm chose a dropout rate of 0.0000. However, using dropout rates is a regularization method to prevent overfitting. The test images should have been evaluated with a model containing dropout rates > 0, to determine if the artifact can be reduced or prevented with regularization. Note, neither Minarik et al.[34] or Xu et al.[16] had any white blob artifacts. Both Minarik et al.[34] and Xu et al.[16] used batch normalization layers before the activation function in the UNETs. Unexpectedly the white blob artifact had pixel values 1.0. The sigmoid function was used as the last layer activation function, the function value can not be $1.0 \leq$, see equation (15). The x term in the sigmoid function must have been large for the pixel values in the white blob. Thus, the function values were rounded to 1.0. For large x values, the sigmoid function has a small derivative of approximately 0, which inhibits the weights from being updated. The large x-values can be a consequence of large weight values in the last convolutional layer. A batch normalization layer followed each convolutional 2d layer except for the last convolutional 2d layer. Therefore large weight values in the previous layers should have a smaller impact in comparison to the weight values in the last layer.

The size of the spleen is oversized for the model trained with kernel initializer truncated normal even though the model has a dropout rate of 0.0097. A model trained with a higher dropout rate could have decreased the over-sizing of the spleen but might have been at the cost of the validation error and noise reduction.

10.2 Comparing UNET to a Butterworth Filter

The training data created without normalization set values ≤ 0.01 to 0.01 before the multiplication with quotient between the total of the sample and label image. Consequently, the min value in the samples was 0.0 and the min value in the labels was 0.01. Zero values can make the training inefficient, with the ReLU activation function the zero values receive a zero derivative, preventing the weights from updating. If the redefining of values ≤ 0.01 had been performed after the multiplication with the mentioned quotient, the image total of the sample and label had not been equivalent. To improve the performance of the model, the UNET algorithm can be trained with sample and label pairs with equivalent image total and the min value defined as 0.01.

The UNET model trained without normalized training data performed worse than the UNET trained with normalized training data. The subfigure C) in 15 has poor spatial resolution, and is blurrier compared to the subfigure G) in 16. Furthermore, the validation loss for the UNET trained without normalization is significantly much higher compared to the UNETs trained with normalized data, see tables 4 and 7. Moreover the right side of the peak for the UNET(no norm) at index 40 in figure 15 is less consistent with the label compared the UNET(truncated normal) in figure 16. However, the profiles can be misleading, since the label profile is consistent with the profiles from the UNET and the Butterworth filter at index x = 80 in figure 15. The peak at index x = 80 represents the spleen, and appears quite different between the label image B), the UNET filtered image C) and the Butterworth filtered image D). The same argument applies to the profiles in figure 16.

The Butterworth filter seems to perform better than the UNET for data that has not been normalized. The MAE calculated between the total of the samples and the Butterworth filtered images is smaller than between the samples and the predictions acquired with the UNET. Hence, the Butterworth filter is better at preserving the image total compared to the UNET. Furthermore the prediction from the UNET image C) is blurrier than the Butterworth filtered image D) in figure 15. Consequently harder to distinguish the backbones from the UNET prediction compared to the Butterworth filtered image. Besides the Butterworth profile is more consistent with the label profile around the liver (the peak at index x = 40) in comparison to the UNET profile, see figure 16.

The UNET trained with normalized data and the Butterworth filter used on normalized data can not be compared with regards to preserving the image total, since the UNET was not trained to preserve the image total. The Butterworth profile is more consistent with the label to the left of the spleen (at index x = 105), compared to the UNET, see figure 16. Otherwise, the profile from the Butterworth filter is consistent with the profile from the UNET. In subfigure G) and H) the Butterworth filtered spleen edges are blurrier than in the UNET prediction, see figure 16. In general,

the spatial resolution is enhanced in the prediction from the UNET compared to the Butterworth filtered profile. Note, one prediction and Butterworth filtered image are compared. To draw conclusions, more predictions and Butterworth filtered images must be compared, for example, with different phantoms and different scale factors. The UNET performs sufficient for a larger scale factor but the Butterworth filter might be superior for lower scale factor values. A Butterworth filter with a smaller D0 value might have a spatial resolution comparable with UNET prediction, for the sample E) in figure 16. But with a larger D0 value, the noise in the image increases since fewer frequencies are cut off.

11 Conclusions

The UNET can reduce the noise in scintillation camera images with a high noise level. However, the UNET trained could not recover details such as the ribs and was overfitted since the size of the spleen in the predictions was not consistent with the labels. Further work is needed to optimize the training data and the architecture to recover details lost in the imaging processing and minimize overfitting. The images from the UNET trained with normalized training data were not as blurry as the images filtered with the Butterworth filter.

12 Appendix

12.1 Base Phantom Modifications

In table 9, the base phantoms are listed alongside respective original height and liver volume. The table contains the interval for each phantom, where-from scaling factors were sampled. The sampled scaling factor was used to scale the short, long, and height axis. All intervals had an increment of 0.01. The number of differently scaled phantoms acquired for each base phantom are listed in column scaled phantoms.

Table 9: The original height and liver volume, number of scaled phantoms acquired and scale interval for each base phantom.

Phantom	Height [cm]	Scale interval [1]	Liver [ml]	scaled phantoms [1]
malepat_118	173.5	(0.96,1.10)	1632	15
malepat_144	170.2	(0.96,1.02)	1937	7
malepat_145	172.1	(0.96,1.10)	1498	15
malepat_150	174.9	(0.94,1.01)	2059	8
malepat_159	173.9	(0.96,1.00)	2063	5
malepat_167	177.6	(0.94,1.08)	1387	15
malepat_169	182.1	(0.91,1.02)	1929	12
vmale_50	175.2	(0.94,1.10)	1776	17
femalepat_71	161.0	(0.94,1.12)	1034	19
femalepat_86	168.6	(0.89,1.07)	1283	19
femalepat_89	161.6	(0.94,1.12)	1315	19
femalepat_117	158.3	(0.95,1.12)	1619	18
femalepat_140	163.1	(0.93,1.10)	1479	18
femalepat_142	164.5	(0.93,1.10)	1481	18
femalepat_143	169.3	(0.92,1.07)	1369	16
femalepat_147	163.9	(0.93,1.03)	1823	11
			Total:	232

Parameters changed to modify the base phantoms are listed in table 10, alongside a description, the interval within the parameter was sampled and respective interval increment.

Table 10: Parameters adjusted in XCAT.

Parameter	Description	Interval	Interval increment
Thickness skin [cm]	If greater than 0, it adds a skin layer to the body	[0.0,0.2]	0.1
Thickness backbone [cm]	No description	[0.3,0.6]	0.1
Start and end slice [slice]	The base phantom is constructed of slices. Since the thorax and pelvis region was of interest, certain slices of the original phantom was extracted. For more variation, the end and begin slice varied between the modified phantoms.	[-25,25]	5
Phantom rot z [degrees]	Degree to rotate the entire phantom by the z-axis	[-7.5,7.5]	2.5
Air in small intestine [1]	0 = do not include air and 1 = include air in small intestine	[0,1]	1
Air in large intestine [1]	Location of air in the large intestine and rectum, 5 = air visible in the entire large intestine and rectum, 2 = air visible in ascending and transverse portions of the large intestine, 1 = air visible in ascending portion of the large intestine only, 0 = no air visible (entire large intestine and rectum filled with contents)	No interval. Possible values 5,2,1,0	-
Left and right diaph liv scale [1]	Height of right and left diaphragm/liver dome, 0 is flat, 1 original height and >1 raises the diaphragm.	[0.95,1.00]	0.05

12.2 UNET Architecture Trained

Table 11, gives a detailed description of UNET architecture trained. The output size does not account for the number of label and sample pairs. The convolutional layer is defined as Conv2D(f×f,F), where f is the width and height of the filter and F is the number of filters. Padding was added in x- and y- axis in every convolutional layer to keep the dimensions of the input image. The stride was set to 1 for the convolutional layers. The dropout layer and the dropout rate are denoted Dropout and D, respectively. The filter and stride size for the max-pooling layers are denoted (s×s,S), where s is the width and height of the filter and S is the stride. The upsampling layer UpSampling2D, enlarges the image with a factor U for rows and columns given in the format (U,U). Concatenate is the concatenation layer between two output volumes of feature maps from the convolutional layers (LX,LY), where the X and Y are the layer indices.

Table 11: UNET architecture trained. Each layer has an index, given in the layer column. The activation function, output shape, layer type and the number of model parameters are listed for each layer.

Layer	**Layer type**	Activation function	Output shape	Params
0	InputLayer	-	(128,128,1)	0
1	Conv2D (3×3, F)	relu	(128,128, F)	$9 \cdot F + F$
2	Conv2D (3×3, F)	relu	(128,128, F)	$9 \cdot F^2 + F$
3	MaxPooling2D (2×2,2)	-	(64,64,F)	0
4	Dropout (0.5D)	-	(64,64,F)	0
5	Conv2D (3×3, 2·F)	relu	(64,64,2·F)	$18 \cdot F^2 + 2 \cdot F$
6	Conv2D (3×3, 2·F)	relu	(64,64,2·F)	$36 \cdot F^2 + 2 \cdot F$
7	MaxPooling2D (2×2,2)	-	(32,32,2·F)	0
8	Dropout (D)	-	(32,32,2·F)	0
9	Conv2D (3×3, 3·F)	relu	(32,32,3·F)	$54 \cdot F^2 + 3 \cdot F$
10	Conv2D (3×3, 3·F)	relu	(32,32,3·F)	$81 \cdot F^2 + 3 \cdot F$
11	MaxPooling2D (2×2,2)	-	(16,16,3·F)	0
12	Dropout (D)	-	(16,16,3·F)	0
13	Conv2D (3×3, 4·F)	relu	(16,16,4·F)	$108 \cdot F^2 + 4 \cdot F$
14	Conv2D (3×3, 4·F)	relu	(16,16,4·F)	$144 \cdot F^2 + 4 \cdot F$
15	MaxPooling2D (2×2,2)	-	(8,8,4·F)	0
16	Dropout (D)	-	(8,8,4·F)	0
17	Conv2D (3×3, 8·F)	relu	(8,8,8·F)	$288 \cdot F^2 + 8 \cdot F$

(To be continued)

Layer	Layer type	Activation function	Output shape	Params
18	Conv2D (3×3, 8·F)	relu	(8,8,8·F)	$576 \cdot F^2 + 8 \cdot F$
19	UpSampling2D (2,2)	-	(16,16,8·F)	0
20	Concatenate (L19,L14)	-	(16,16,8·F+4·F)	0
21	Dropout (D)	-	(16,16,8·F+4·F)	0
22	Conv2D (3×3, 4·F)	relu	(16,16,4·F)	$432 \cdot F^2 + 4 \cdot F$
23	Conv2D (3×3, 4·F)	relu	(16,16,4·F)	$144 \cdot F^2 + 4 \cdot F$
24	UpSampling2D (2,2)	-	(32,32,4·F)	0
25	Concatenate (L24,L10)	-	(32,32,4·F+3·F)	0
26	Dropout (D)	-	(32,32,4·F+3·F)	0
27	Conv2D (3×3, 3·F)	relu	(32,32,3·F)	$189 \cdot F^2 + 3 \cdot F$
28	Conv2D (3×3, 3·F)	relu	(32,32,3·F)	$81 \cdot F^2 + 3 \cdot F$
29	UpSampling2D (2,2)	-	(64,64,3·F)	0
30	Concatenate	-	(64,64,3·F+2·F)	0
31	Dropout (D)	-	(64,64,3·F+2·F)	0
27	Conv2D (3×3, 2·F)	relu	(64,64,2·F)	$90 \cdot F^2 + 2 \cdot F$
28	Conv2D (3×3, 2·F)	relu	(64,64,2·F)	$36 \cdot F^2 + 2 \cdot F$
29	UpSampling2D (2,2)	-	(128,128,2·F)	0
30	Concatenate (L29,L2)	-	(128,128,2·F+F)	0
31	Dropout	-	(128,128,2·F+F)	0
32	Conv2D (3×3, F)	relu	(128,128,F)	$27 \cdot F^2 + F$
33	Conv2D (3×3, F)	relu	(128,128,F)	$9 \cdot F^2 + F$
34	Conv2D (1×1, 1)	sigmoid	(128,128,1)	$1 \cdot F + 1$

References

[1] Ljungberg M, Gleisner KS. Hybrid Imaging for Patient-Specific Dosimetry in Radionuclide Therapy. Diagnostics. 2015 Jul;5(3):296–317.

[2] Khalil M. Basic Sciences of Nuclear Medicine. Berlin and Heidelberg: Springer; 2011.

[3] Knoll G. Radiation Detection and Measurement. 4th ed. New York: Wiley; 2010.

[4] ICRP. The 2007 Recommendations of the International Commission on Radiological Protection. Elsevier Ltd; 2007.

[5] Hurwitz J, Kirsch D. Machine Learning for Dummies. Wiley, New Jersey; 2018.

[6] Yamashita R, Nishio M, Do RKG, Togashi K. Convolutional neural networks: an overview and application in radiology. Insights into Imaging. 2018 Jun;9:611–629.

[7] Mebsout I. Convolutional Neural Networks' mathematics. towards data science; 2020 [updated 2020 Feb; cited 2020 Dec]. Available from: https://towardsdatascience.com/convolutional-neural-networks-mathematics-1beb3e6447c0.

[8] Robinson R. Convolutional Neural Networks - Basics An Introduction to CNNs and Deep Learning; 2017 Apr[cited 2020 Dec]. Available from: https://mlnotebook.github.io/post/CNN1/.

[9] Indolia S, Goswami AK, Mishra S P , Asopa P. Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. Elsevier. 2018;132(2018):679–688.

[10] Brownlee J. Use Weight Regularization to Reduce Overfitting of Deep Learning Models; 2018 Nov [cited 2021 Jan]. Available from: https://machinelearningmastery.com/weight-regularization-to-reduce-overfitting-of-deep-learning-models/.

[11] He K, Zhang X, Ren S, Sun J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. CoRR. 2015;abs/1502.01852. Available from: http://arxiv.org/abs/1502.01852.

[12] Tensorflow. tf.keras.initializers.HeUniform; [date unknown][updated 2021 Jan; cited 2021 Jan]. Available from: https://www.tensorflow.org/api_docs/python/tf/keras/initializers/HeUniform.

[13] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Teh YW, Titterington M, editors. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. vol. 9 of Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR; 2010. p. 249–256. Available from: http://proceedings.mlr.press/v9/glorot10a.html.

[14] Keras. Layer weight initializers; [date unknown][cited 2021 Jan]. Available from: https://keras.io/api/layers/initializers/#truncatednormal-class.

[15] Grover P. 5 Regression Loss Functions All Machine Learners Should Know. Heartbeat; 2018 [updated 2018 Jun; cited 2020 Dec]. Available from: https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0.

[16] Xu J, Gong E, Pauly JM, Zaharchuk G. 200x Low-dose PET Reconstruction using Deep Learning. CoRR. 2017;abs/1712.04119. Available from: http://arxiv.org/abs/1712.04119.

[17] Nilsson J, Akenine-Möller T. Understanding SSIM. 2020. Available from: https://arxiv.org/abs/2006.13846.

[18] Horé A, Ziou D. Image Quality Metrics: PSNR vs. SSIM. In: 2010 20th International Conference on Pattern Recognition; 2010. p. 2366–2369.

[19] Dosselmann R, Yang XD. Comprehensive Assessment of the Structural Similarity Index. SIViP. 2011 Mar;5:81–91. Available from: https://doi.org/10.1007/s11760-009-0144-1.

[20] Liu L, Cheng J, Quan Q, Wu FX, Wang YP, Wang J. A survey on U-shaped networks in medical image segmentations. Neurocomputing (Netherlands). 2020 Oct;409:244 – 58. Available from: http://dx.doi.org/10.1016/j.neucom.2020.05.070.

[21] Milletari F, Navab N, Ahmadi S. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. CoRR. 2016;abs/1606.04797. Available from: http://arxiv.org/abs/1606.04797.

[22] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. CoRR. 2015;abs/1502.03167. Available from: http://arxiv.org/abs/1502.03167.

[23] Pykes K. The Vanishing/Exploding Gradient Problem in Deep Neural Networks. towards data science; 2020 May [cited[2021 Jan].

[24] Keras. BatchNormalization layer; [date unknown][cited 2020 Dec]. Available from: https://keras.io/api/layers/normalization_layers/batch_normalization/.

[25] Santurkar S, Tsipras D, Ilyas A, Madry A. How Does Batch Normalization Help Optimization? 2019. Available from: https://arxiv.org/abs/1805.11604.

[26] Ying X. An Overview of Overfitting and its Solutions. Journal of Physics: Conference Series. 2019 Feb;1168:022022. Available from: https://doi.org/10.1088/1742-6596/1168/2/022022.

[27] Park S, Kwak N. Analysis on the Dropout Effect in Convolutional Neural Networks. vol. 10112. Lai SH, Lepetit V, Nishino K, Sato Y, editors. Springer, Cham; 2017.

[28] Brownlee J. How to Implement Bayesian Optimization from Scratch in Python; 2019 Oct [updated 2020 Aug; cited 2020 Dec]. Available from: https://machinelearningmastery.com/what-is-bayesian-optimization/.

[29] Feurer M, Springenberg JT, Hutter F. Initializing Bayesian Hyperparameter Optimization via Meta-Learning. In: AAA1'15: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI Press; 2015. p. 1128–1135.

[30] Ljungberg M, Strand SE, King MA. The SIMIND Monte Carlo program Monte Carlo Calculation in Nuclear Medicine: Applications in Diagnostic Imaging. 2nd ed.; 2012.

[31] Segars WP, Sturgeon G, Mendonca S, Grimes J, BM Tsui. 4D XCAT phantom for multimodality imaging research. Med Phys. 2010;37(9):4902–4915.

[32] Kainz W, Neufeld E, Bolch WE, Graff CG, Kim CH, Kuster N, et al. Advances in Computational Human Phantoms and Their Applications in Biomedical Engineering—A Topical Review. IEEE Transactions on Radiation and Plasma Medical Sciences. 2019;3(1):1–23.

[33] Chang D. XCAT Medical Imaging Simulation using Computational Phantoms; [date unknown][cited 2020 Dec]. Available from: https://olv.duke.edu/industry-investors/available-technologies/xcat/.

[34] Minarik D, Enqvist O, Trägårdh E. Denoising of Scintillation Camera Images Using a Deep Convolutional Neural Network: A Monte Carlo Simulation Approach. J Nucl Med. 2020 Feb;61(2):298–303.