Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Technical Physics
Technomathematics

Angelina Senchukova

LEARNED IMAGE RECONSTRUCTION IN X-RAY COMPUTED TOMOGRAPHY

Master's Thesis

Examiners: Professor Heikki Haario
Adjunct Professor Tuomas Eerola

Supervisors: Professor Heikki Haario

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Technical Physics
Technomathematics

Angelina Senchukova

LEARNED IMAGE RECONSTRUCTION IN X-RAY COMPUTED TOMOGRAPHY

Master's Thesis

2020

101 pages, 81 figures.

Examiners: Professor Heikki Haario
Adjunct Professor Tuomas Eerola

Keywords: learned image reconstruction, X-ray tomography, Radon transform

The basic idea behind X-ray computed tomography (CT) is, given the change in the intensities of X-ray beams passing through a target object, to reconstruct the image of the object's density that is characterized by the attenuation function. The CT scanning process results in the Radon transform of the attenuation function, therefore the reconstruction problem leads to the inversion of this transform. The inversion can be performed by applying, for example, the filtered backprojection (FBP) formula. However, this classical reconstruction approach often results in images corrupted by artefacts. The learned image reconstruction, based on artificial neural networks, is employed to solve the reconstruction quality issue. This thesis aimed to compare the generalization capabilities of two learned image reconstruction approaches: a two-step approach when FBP-reconstruction is followed by the U-net neural network post-filtering, and a one-step approach when image reconstruction is fully automated with the AUTOMAP deep neural network. The two-step approach was shown to outperform the one-step approach since the latter one requires highly diverse training sets to be able to generalize that poses a problem in the context of CT where the amount of real data available for training is usually limited.

PREFACE

First of all, I would like to thank my supervisor Professor Heikki Haario for his guidance and unwavering support throughout the research process. Also, a huge thanks to Alexey Kazarnikov for his insightful comments and suggestions that kept me motivated day by day. Additionally, I wish to thank Assistant Professor Andreas Hauptmann, whose work gave me the inspiration and impetus to immerse myself in the topic of this thesis.

I really appreciate the opportunity to be a part of the Lappeenranta-Lahti University of Technology during this year and study in the calm, warm atmosphere of peer-support. Thanks to the University, I had the privilege of being a student of perfectly qualified professionals, including Peter Jones and Hwei-Ming Boey who encouraged my interest in the English language and gave me an immense number of first-class, extraordinary lessons I will never forget.

My parents deserve a particular note of thanks: they taught me the greatest lesson in my life — to chase my dream and never give up. Their unconditional love and kind words saved me the minute I was frustrated and exhausted.

Finally, I would like to thank my English mentor, 'partner in crime' and bosom friend — Aakif Nawaz for reviewing this manuscript and his belief in my success.

Lappeenranta, June 21, 2020

Angelina Senchukova

CONTENTS

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
AUTOMAP	Automated Transform by Manifold Approximation
CNN	Convolutional Neural Network
CT	Computed Tomography
DNN	Deep Neural Network
EIT	Electrical Impedance Tomography
FBP	Filtered Backprojection
GPU	Graphical Processing Unit
HDF	Hierarchical Data Format
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
RGB	Red, Green and Blue

1 INTRODUCTION

1.1 Background

X-ray computed tomography (CT) is a medical imaging technique that allows performing non-invasive diagnostics, i.e. assessment of the internal structure of a scanned object without cutting [1]. The technique is based on the measurement of changes in the intensities of multiple X-ray beams that cross the object from different angles (projections). While scanning, X-ray beams are emitted from a machine (with known initial intensities), next the beams are passed through the object, and their final intensities are recorded by a CT detector. Given the measurements of initial and final intensities for enough number of X-ray beams, the two-dimensional slices (cross-sectional images) of the scanned object can be obtained and displayed on a screen.

The quantity which characterizes the density of the object, together with the amount of intensity it causes X-ray beams to lose, is known as the attenuation coefficient [2]. Let $A(x, y)$ be the attenuation coefficient of the scanned object at the point (x, y). A single X-ray beam can be considered as a straight line l crossing the body. Therefore the scanning process results in the set of the line integrals of the function $A(x, y)$ along each of the lines l. The transform that maps a function on $\mathbb{R}^2$ into the set of its line integrals is known as the Radon transform. Thus, the reconstruction problem in CT leads to the inversion of the two-dimensional Radon transform in $\mathbb{R}^2$.

One possible way to recover an image of the function $A(x, y)$ from the values of its Radon transform is to apply the backprojection formula. However, this approach results in the smoothed version of the original attenuation coefficient, and consequently in highly blurred images. That is why in CT image reconstruction, filtered backprojection (FBP) is commonly used to correct the smoothing effect of the backprojection. In this approach, the data received from a CT detector are filtered at first (e.g. with low-pass filters), and next, the backprojection transform is applied [1].

The assumption behind the filtered backprojection formula is that all the possible CT projections are known. However, in practice, only a finite number of X-ray projections are available and can be used to reconstruct the image from the detector measurements. The limited data at the disposal usually result in artefacts that impair the quality of reconstructions and therefore impede accurate medical diagnostics. Consequently, there is a need for post-filtering of the reconstructed images.

Along with the classical image processing techniques [3], machine learning and artificial neural networks (ANNs) are widely used nowadays to post-process the reconstructions obtained by traditional algorithms such as FBP, aiming at enhanced reconstruction quality (see [4] for general information). Such an approach can be called a two-step approach since it consists of two successive stages: conventional reconstruction stage and post-filtering stage performed by a neural network. Since the approach is based on ANNs, it refers to learned image reconstruction.

For instance, the two-step approach applied for reconstruction in the context of real-time electrical impedance tomography (EIT) is proposed in [5, 6]. In the first step, the direct reconstructions are obtained using the D-bar method that provides reliable image recovery but suffers from a blurring (loss of sharp features). The issue of blurred reconstructions is solved in the second step by applying a convolutional neural network (CNN) that successfully learns the deblurring, resulting in images with sharp features important in medical applications.

An alternative approach related to learned image reconstruction can be called a one-step approach since it implements reconstruction and post-processing stages jointly. An example of this approach has been proposed recently in [7]. The method is applied to photo-acoustic tomography data and provides higher quality reconstructions as compared to other learned and non-learned techniques.

One more example of the one-step approach called Learned Primal-Dual Algorithm is introduced in [8] as a framework that integrates the knowledge of a forward model of computed tomography into the design of a deep neural network for solving the inverse problem of CT. The proposed technique involves learning the whole reconstruction operator, mapping directly raw measured data to reconstruction, rather than just post-processing. Thus, the algorithm does not depend on initial reconstructions, for instance, FBPs.

Another example of the one-step approach called automated transform by manifold approximation (AUTOMAP) allows learning a reconstruction mapping between the detector domain (input sinogram) and the image domain (output reconstruction). The AUTOMAP reconstruction framework is implemented with a deep neural network and is shown to be resistant to noise and cause fewer artefacts compared to conventional handcrafted reconstruction approaches [9].

1.2 Objectives and delimitations

This Master's thesis focuses on evaluating two different approaches to learned image reconstruction in computed tomography: a two-step approach when conventional image reconstruction is followed by convolutional neural network post-filtering, and a one-step approach when image reconstruction is fully automated with a deep neural network. In this thesis, the U-net convolutional neural network is used in the realization of the two-step approach, while the one-step approach is implemented using the AUTOMAP deep neural network. The limiting factor in the selection of network architectures is the type of input data inherent to the context of computed tomography. The development of new learned image reconstruction approaches is beyond the scope of the thesis. The objective is to compare the generalization capabilities of two learned image reconstruction approaches:

- the two-step approach when U-net is used to post-process the filtered backprojection reconstructions;
- the one-step approach with AUTOMAP applied to learn the full reconstruction process, mapping directly input raw data collected by a CT detector (sinograms) to the output reconstructed image.

1.3 Structure of the thesis

The thesis is organized as follows. Section 2 introduces the main concepts underlying the Radon transform applications in X-ray computed tomography: the mathematical model of X-ray tomography, the definition of the Radon transform and how it is implemented in MATLAB, the idea of backprojection filtering and examples of low-pass filters in computerized tomography. Section 3 presents a detailed description of deep neural networks (DNNs), including the structure of biological and artificial neurons, the major stages in the development of artificial neural networks, the concept of multilayer perceptron and its learning rule, most commonly used activation functions and adaptive gradient methods applied for optimization of the loss function in the training process, and finally architectures of convolutional neural networks and U-net particularly. The experiments on two different approaches in learned image reconstruction are described in Section 4. Finally, the results are summarized in Section 5, and the conclusion is given in Section 6.

2 RADON TRANSFORM IN X-RAY TOMOGRAPHY

2.1 Forward and inverse problems of mathematical modeling

Modern engineering is based on modeling of various phenomena and processes (research objects) arising in different spheres of human activity. To determine and predict the behaviour of a research object it is necessary to develop its mathematical model, i.e. describe its main features using mathematical concepts. The process of creating a mathematical model is known as mathematical modeling [10].

Investigation of a research object is grounded in two main information sources: observation and experiment. Observation allows establishing behavioural characteristics of a research object. During an experiment, it is possible to affect the object (input of a research object) and record the response (output of a research object). The main goal of mathematical modeling is to determine the connection between the input and the output of a research object [11].

In terms of the cause-and-effect relationship between input and output of a research object, all the problems of mathematical modeling can be divided into two classes: forward problems and inverse problems. The purpose of forward problems is to find the effects given the causes, while for the class of inverse problems, it is required to determine the causes using the information about the effects as shown in Fig. 1 (see [12, 13] for details). Inverse problems have various applications in computer vision [14], machine learning [15], medical imaging [16–18] and many other fields of science. One of the classical examples of the inverse problem is X-ray tomography described in Section 2.2.

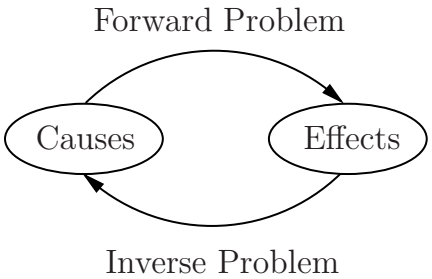


Figure 1. Definition of forward and inverse problems. The forward problem starts with the the causes and then calculates the effects. The inverse problem is the reverse of the forward model: it begins with the effects and then calculates the causes.

2.2 X-ray tomography and its mathematical model

The theoretical aspects presented in Sections 2.2—2.6 closely follow the material introduced in [2, 19]. Tomography is a technique imaging a structure of two- or three-dimensional object from many one-dimensional slices of the object. For example, in computed tomography (CT) these slices are determined by parallel X-ray beams perpendicular to the object. CT detector records intensity loss of the beams to produce a two-dimensional image that can be displayed on a screen (see [1, 2] for details).

It is a well-known fact that the amount of energy decrease of an X-ray beam when passing through an object depends on the density of the object. The density of the object along with the amount of intensity it causes a beam to lose is characterized by the attenuation coefficient. The attenuation coefficient $A(x)$ is the proportion of photons absorbed per millimeter of substance at a distance x from the origin. The Beer-Lambert Law describes a relationship between the attenuation coefficient $A(x)$ and the intensity $I(x)$ of an X-ray beam that crosses an inhomogeneous material along a distance x from the origin

$$\frac{dI}{dx} = -A(x)I(x). \tag{1}$$

Let $I(X_0) = I_0$ denote the initial intensity of a beam at the point X_0 (X-ray source) and $I(X_1) = I_1$ denote its final intensity at the point X_1 (X-ray detector). After integrating equation (1) along the line X_0X_1 the following result is obtained

$$\int_{I_0}^{I_1} \frac{dI}{I} = -\int_{X_0}^{X_1} A(x)dx.$$

Thus, the equation connecting the initial and final intensities to the attenuation coefficient is obtained

$$\ln\left(\frac{I_0}{I_1}\right) = \int_{X_0}^{X_1} A(x)dx. \tag{2}$$

In other words, the scanning process results in the line integrals of the function $A(x)$ along each of the lines X_0X_1. The attenuation coefficient $A(x)$ can be reconstructed from all these integrals. In the context of CT scans, two-dimensional slices of a three-dimensional objects are considered. The slices are obtained as an intersection of the object and some plane, and the attenuation coefficient is understood as a function of two variables $A(x, y) \in \mathbb{R}^2$ within the particular slice [2]. The transform that maps a function on $\mathbb{R}^2$ into the set of its line integrals is known as the two-dimensional Radon transform. Consequently, the reconstruction problem of CT leads to the inversion of the

two-dimensional Radon transform.

2.3 Radon transform

The Radon transform is an important mathematical tool for a large class of reconstruction problems [1]. This integral transform and its inversion formula were introduced by Johann Radon (see original German text [20] and its English translation [21]).

Before introducing the definition of the Radon transform, it is necessary to adopt a specific coordinate system (Fig. 2). It is a well-known fact that the equation of a line AA' in $\mathbb{R}^2$ can be written in the standard form

$$ax + by = c, \tag{3}$$

where $a, b, c \in \mathbb{R}$, and $a^2 + b^2 \neq 0$. Equation (3) can be rewritten as follows

$$\frac{a}{\sqrt{a^2+b^2}}x + \frac{b}{\sqrt{a^2+b^2}}y = \frac{c}{\sqrt{a^2+b^2}}. \tag{4}$$

If the following notation is used

$$\vec{\omega} = (\omega_1, \omega_2) = \left(\frac{a}{\sqrt{a^2+b^2}}, \frac{b}{\sqrt{a^2+b^2}}\right),$$

then $\vec{\omega}$ is a point lying on the unit circle, since

$$\left(\frac{a}{\sqrt{a^2+b^2}}\right)^2 + \left(\frac{b}{\sqrt{a^2+b^2}}\right)^2 = 1.$$

Thus, there exists an angle $\alpha \in [0, 2\pi)$ such that $\vec{\omega} = (\cos\alpha, \sin\alpha)$. If $\rho = \frac{c}{\sqrt{a^2+b^2}}$ and $\vec{z} = (x, y)$ in equation (4), then

$$\langle \vec{\omega}, \vec{z} \rangle = \rho,$$

where $\langle \cdot, \cdot \rangle$ denotes inner product, or, equivalently

$$x\cos\alpha + y\sin\alpha = \rho. \tag{5}$$

It should be noticed that in equation (5) the parameters α and ρ are fixed and uniquely determine a specific line AA' in the plane (Fig. 2), and $\vec{z}$ determines a point on this line,

therefore, the line AA' can be represented as

$$AA' = \{\vec{z} = (x, y) \in \mathbb{R}^2 : \langle \vec{\omega}, \vec{z} \rangle = \rho\}.$$

An alternative representation of a specific point lying on the line AA' can be given. Since the vector $\vec{v} = (-\sin\alpha, \cos\alpha)$ is perpendicular to the normal vector $\vec{n} = (\cos\alpha, \sin\alpha)$, it is possible to describe a specific point (x, y) on AA' as follows

$$(x(s), y(s)) = \rho\vec{n} + s\vec{v} = (\rho\cos\alpha, \rho\sin\alpha) + (-s\sin\alpha, s\cos\alpha), \tag{6}$$

where $s \in \mathbb{R}$ is a parametrization parameter. Thus, the alternative representation for the set of all points on the line AA' is

$$AA' = \{(x(s), y(s)) = (\rho\cos\alpha - s\sin\alpha, \rho\sin\alpha + s\cos\alpha) : s \in \mathbb{R}\}. \tag{7}$$

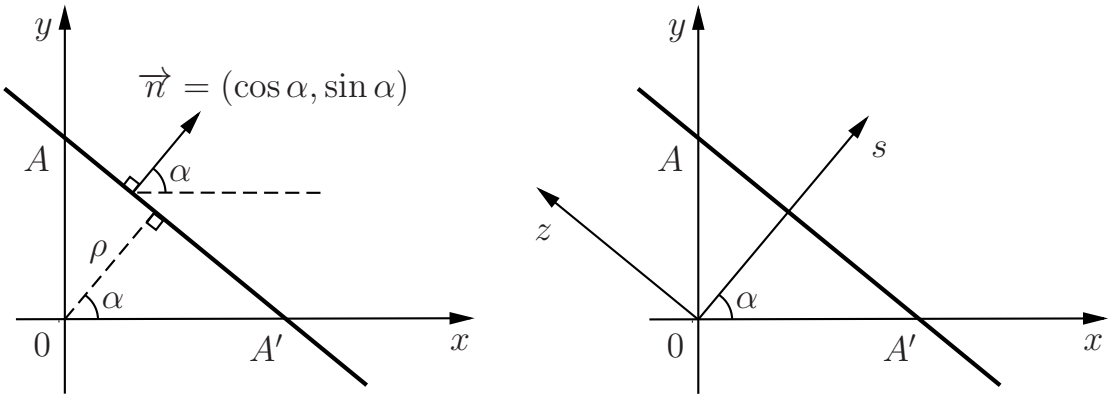


Figure 2. Coordinate transformation. The coordinate system Osz is derived from the coordinate system Oxy by rotating it α degrees counterclockwise. The parameters α (inclination angle) and ρ (distance from the origin measured along the normal vector $\vec{n}$ with the corresponding sign) determine a point $(\rho\cos\alpha, \rho\sin\alpha)$ in the space and when $\vec{z}$ changes the point travels up and down along the line AA'. The line AA' is perpendicular to the normal vector $\vec{n} = (\cos\alpha, \sin\alpha)$.

Finally, the definition of the Radon transform can be introduced. For a function f on $\mathbb{R}^2$ with compact support its Radon transform $\mathcal{R}f$ is defined as an integral [22]

$$\mathcal{R}f(\rho, \alpha) = \int_{-\infty}^{+\infty} f(x(s), y(s))ds, \tag{8}$$

where $\alpha \in [0, 2\pi)$, $\rho \in \mathbb{R}$, and $s \in \mathbb{R}$ are defined by (7).

In geometrical terms, the Radon transform is an integral of $f(x, y)$ over the line AA' that is perpendicular to the normal vector $\vec{n} = (\cos\alpha, \sin\alpha)$ and lies at a distance ρ (measured along the normal vector $\vec{n}$ with the corresponding sign) from the origin (Fig. 2). Taking into account representation (6), an alternative definition of the Radon transform is obtained

$$\mathcal{R}f(\rho, \alpha) = \int_{-\infty}^{+\infty} f(\rho\cos\alpha - s\sin\alpha, \rho\sin\alpha + s\cos\alpha)ds. \tag{9}$$

2.4 Connection between the Radon and Fourier transforms

The Fourier transform is used to derive the inverse formula of the Radon transform. The one-dimensional Fourier transform $\mathcal{F}f$ of an absolutely integrable function f on $\mathbb{R}$ is defined for all $\omega \in \mathbb{R}$ as

$$\mathcal{F}f(\omega) = \int_{-\infty}^{+\infty} f(x)e^{-i\omega x}dx.$$

The corresponding inverse Fourier transform is given by the formula

$$\mathcal{F}^{-1}f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \mathcal{F}f(\omega)e^{i\omega x}d\omega.$$

The two-dimensional Fourier transform is defined for all $(u, v) \in \mathbb{R}^2$ as

$$\mathcal{F}_2 f(u, v) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x, y)e^{-i(ux+vy)}dxdy.$$

The corresponding inverse Fourier transform is defined as follows

$$\mathcal{F}_2^{-1}f(x, y) = \frac{1}{4\pi^2}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mathcal{F}_2 f(u, v)e^{i(ux+vy)}dudv. \tag{10}$$

The following theorem is proved to provide the connection between the Fourier and Radon transforms [23].

Theorem 1 *(The Central Slice Theorem) For an absolutely integrable function f defined*

on $\mathbb{R}^2$*, all* $t \in \mathbb{R}$*, and* $\alpha \in [0, 2\pi)$ *the following equality holds*

$$\mathcal{F}_2 f(t\cos\alpha, t\sin\alpha) = \mathcal{F}(\mathcal{R}f)(t, \alpha).$$

Proof. The two-dimensional Fourier transform of a function f is considered

$$\mathcal{F}_2 f(t\cos\alpha, t\sin\alpha) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x, y)e^{-it(x\cos\alpha + y\sin\alpha)}dxdy. \tag{11}$$

A change of variables according to the coordinate system defined in Section 2.3 results in

$$x(s) = \rho\cos\alpha - s\sin\alpha, \quad y(s) = \rho\sin\alpha + s\cos\alpha.$$

Making substitution of variables (ρ, s) in the integral (11) and taking into account that

$$x\cos\alpha + y\sin\alpha = \rho\cos^2\alpha - s\sin\alpha\cos\alpha + \rho\sin^2\alpha + s\sin\alpha\cos\alpha = \rho,$$

and the Jacobian determinant for $x(s)$ and $y(s)$ is given by

$$\det\left(\begin{bmatrix} \frac{\partial x}{\partial \rho} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial \rho} & \frac{\partial y}{\partial s} \end{bmatrix}\right) = \det\left(\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}\right) = \cos^2\alpha + \sin^2\alpha = 1,$$

equation (11) can be rewritten as follows

$$\mathcal{F}_2 f(t\cos\alpha, t\sin\alpha) = \int_{-\infty}^{+\infty}\left(\int_{-\infty}^{+\infty} f(\rho\cos\alpha - s\sin\alpha, \rho\sin\alpha + s\cos\alpha)e^{-it\rho}ds\right)d\rho.$$

Taking into account the definition of the Radon transform (9), the following equation is obtained

$$\mathcal{F}_2 f(t\cos\alpha, t\sin\alpha) = \int_{-\infty}^{+\infty} e^{-it\rho}\mathcal{R}f(\rho, \alpha)d\rho, \tag{12}$$

that is the Fourier transform of the Radon transform $\mathcal{R}f$. •

Thus, Theorem 1 states that the two-dimensional Fourier transform of the function f along a line AA' at the inclination angle α is the one-dimensional Fourier transform of the Radon transform of this function.

2.5 Filtered backprojection formulas

When the connection between the one-dimensional Radon and two-dimensional Fourier transform is established (see Theorem 1), it is possible to introduce the backprojection formula that allows recovering the attenuation coefficient $A(x)$ described in Section 2.2. In physical terms the Radon transform $\mathcal{R}f(\rho, \alpha)$ provides the total density of the object f along some line (Fig. 2). This density can be determined by measuring the initial and final intensities of an X-ray beam passing through the object (see equation (2)). A single slice of an object is obtained when X-ray beams are shot along many different lines. Changing the inclination angle α of these beams gives multiple slices of the object. If the backprojections of the densities are calculated the object can be reconstructed (see [24] for general information and [25] for technical details).

The backprojection [2] of a function $h(t, \alpha)$ (where t and α denote polar coordinates) at the point (x, y) is defined as follows

$$\mathcal{B}h(x, y) := \frac{1}{\pi} \int_0^\pi h(x \cos\alpha + y \sin\alpha, \alpha) d\alpha. \tag{13}$$

Applying equation (13) to the Radon transform of the attenuation-coefficient function f in the context of CT results in the formula for the backprojection of $\mathcal{R}f$ at a point (x, y)

$$\mathcal{B}\mathcal{R}f(x, y) = \frac{1}{\pi} \int_0^\pi \mathcal{R}f(x \cos\alpha + y \sin\alpha, \alpha) d\alpha. \tag{14}$$

Thus, using formula (14), the backprojections of the slices can be obtained. However, this approach results in the smoothed version of the original function f. The filtered backprojection (FBP) is introduced to correct this smoothing effect [2].

Theorem 2 *If f is an absolutely integrable function defined on $\mathbb{R}^2$, then it can be found as the following filtered backprojection*

$$f(x, y) = \frac{1}{2} \mathcal{B}[\mathcal{F}^{-1}[|t| \mathcal{F}(\mathcal{R}f)(t, \alpha)]](x, y). \tag{15}$$

Proof. After taking into account equation (10), for the two-dimensional Fourier transform

and its inverse transform the following holds

$$f(x,y) = \mathcal{F}_2^{-1}\mathcal{F}_2 f(x,y) = \frac{1}{4\pi^2}\int\limits_{-\infty}^{+\infty}\int\limits_{-\infty}^{+\infty}\mathcal{F}_2 f(u,v)e^{i(xu+yv)}dudv. \tag{16}$$

The change of variables from the Cartesian coordinates (u,v) to polar coordinates (t,α) results in

$$u = t\cos\alpha, \quad v = t\sin\alpha,$$

where $t \in \mathbb{R}$, $\alpha \in [0,\pi]$. The Jacobian determinant of this coordinate transformation is

$$\det\left(\begin{bmatrix} \frac{\partial u}{\partial t} & \frac{\partial u}{\partial \alpha} \\ \frac{\partial v}{\partial t} & \frac{\partial v}{\partial \alpha} \end{bmatrix}\right) = \det\left(\begin{bmatrix} \cos\alpha & -t\sin\alpha \\ \sin\alpha & t\cos\alpha \end{bmatrix}\right) = t\cos^2\alpha + t\sin^2\alpha = |t|,$$

consequently, $dudv = |t|dtd\alpha$. Substituting variables (t,α) into (16) results in the inverse Fourier transform in polar coordinates

$$f(x,y) = \frac{1}{4\pi^2}\int\limits_{0}^{\pi}\int\limits_{-\infty}^{+\infty}\mathcal{F}_2 f(t\cos\alpha, t\sin\alpha)e^{it(x\cos\alpha+y\sin\alpha)}|t|dtd\alpha. \tag{17}$$

After taking into account Theorem 1, equation (17) can be rewritten as

$$f(x,y) = \frac{1}{4\pi^2}\int\limits_{0}^{\pi}\left(\int\limits_{-\infty}^{+\infty}\mathcal{F}(\mathcal{R}f(t,\alpha))e^{it(x\cos\alpha+y\sin\alpha)}|t|dt\right)d\alpha. \tag{18}$$

Equation (18) can be rewritten by considering its inner integral

$$\int\limits_{-\infty}^{+\infty}\mathcal{F}(\mathcal{R}f(t,\alpha))e^{it(x\cos\alpha+y\sin\alpha)}|t|dt = 2\pi\left(\frac{1}{2\pi}\int\limits_{-\infty}^{+\infty}\mathcal{F}(\mathcal{R}f(t,\alpha))e^{it(x\cos\alpha+y\sin\alpha)}|t|dt\right) =$$

$$= 2\pi\mathcal{F}^{-1}[|t|\mathcal{F}(\mathcal{R}f)(t,\alpha)](x\cos\alpha + y\sin\alpha,\alpha).$$

Thus, equation (18) will be as follows

$$f(x,y) = \frac{1}{2\pi}\int\limits_{0}^{\pi}\mathcal{F}^{-1}[|t|\mathcal{F}(\mathcal{R}f)(t,\alpha)](x\cos\alpha + y\sin\alpha,\alpha)d\alpha.$$

Taking into account the fact that the integral in the right-hand side of the above equation is $\frac{1}{2}$ the backprojection (14) for the function $[|t|\mathcal{F}(\mathcal{R}f)(t,\alpha)]$, the above equation can be simplified as follows

$$f(x,y) = \frac{1}{2}\mathcal{B}[\mathcal{F}^{-1}[|t|\mathcal{F}(\mathcal{R}f)(t,\alpha)]](x,y).$$

2.6 Filtering factor

In the filtered backprojection formula (15), the multiplier $|t|$ is called the filtering factor. This section investigates the contribution of $|t|$ to equation (15). Under the assumption that there exists a function φ such that its Fourier transform is equal to the filtering factor $|t|$, i.e.

$$\mathcal{F}\varphi(t) = |t|.$$

it is possible to rewrite the filtered backprojection formula (15) as follows

$$f(x,y) = \frac{1}{2}\mathcal{B}[\mathcal{F}^{-1}[\mathcal{F}\varphi \cdot \mathcal{F}(\mathcal{R}f)(t,\alpha)]](x,y). \tag{19}$$

According to the convolution theorem [26] the Fourier transform of a convolution of two functions is the pointwise product of their Fourier transforms

$$\mathcal{F}(f * g) = \mathcal{F}f \cdot \mathcal{F}g. \tag{20}$$

Taking into account equation (20), equation (19) can be rewritten as follows

$$f(x,y) = \frac{1}{2}\mathcal{B}[\mathcal{F}^{-1}[\mathcal{F}(\varphi * \mathcal{R}f)(t,\alpha)]](x,y).$$

Since the inverse Fourier transform of the Fourier transform is equivalent to identity transformation, the following equation is obtained

$$f(x,y) = \frac{1}{2}\mathcal{B}(\varphi * \mathcal{R}f)(x,y). \tag{21}$$

In the context of computed tomography, $\mathcal{R}f$ in equation (21) represents the data measured by a CT detector, the function φ filters the data, then the backprojection $\mathcal{B}$ is applied.

However, it turns out that there is no function φ such that its Fourier transform is exactly equal to the absolute value function $|t|$ (see [1]). Indeed, $|t|$ tends to infinity as $t \to \infty$,

but the function $\mathcal{F}\varphi$ defined as follows

$$\mathcal{F}\varphi(\omega) = \int_{-\infty}^{+\infty} \varphi(x) e^{-i\omega x} dx,$$

tends to zero as $\omega \to \infty$. This problem can be avoided by focusing on band limited functions. A function φ is called a band limited function if for some real number $M > 0$

$$\mathcal{F}\varphi(\omega) = \int_{-\infty}^{+\infty} \varphi(x) e^{-i\omega x} dx = 0, \quad \forall \omega \notin [-M, M],$$

i.e. the Fourier transform of a band limited function is equal to zero outside a finite interval $[-M, M]$. Under the assumption that the function φ is band limited, the filtering factor $|t|$ in the filtered backprojection formula (15) can be replaced with the function S with compact support defined as

$$S = \mathcal{F}\varphi(\omega). \tag{22}$$

This function belongs to the class of low-pass filters that are discussed in detail in Section 2.7. It should be noticed that if function S is used then there is no longer equality in equation (21), i.e.

$$f(x, y) \approx \frac{1}{2} \mathcal{B}(\mathcal{F}^{-1} S * \mathcal{R}f)(x, y).$$

2.7 Filtering in computed tomography

The usage of a specific filter depends on different purposes such as artefact reduction (the high-pass filters), noise suppression (the low-pass filters), or signal enhancement (the enhancement filters). One of the applications of the high-pass filters is to eliminate star artefacts that result from the limited number of backprojections. The specific example of these filters is the ramp filters which block low frequencies that make an image blurry. The high-pass filters sharpen parts of an image where the signal change is rapid (i.e. edges). However, these filters suffer from the amplification of statistical noise. In order to reduce the exaggeration of high frequencies, the high pass filters are usually accompanied by the low-pass filters [27].

The low-pass (smoothing) filters are used to remove statistical noise in images. These filters maintain the low frequencies while eliminating the high frequencies. The parameter characterizing low-pass filters is called cut-off frequency that defines the frequency above which the noise is blocked. The filter function is defined to be zero for all the frequen-

cies above cut-off frequency [27]. Since the low-pass filters are of particular interest in imaging, they are considered in detail below.

In terms of functions, low-pass filters are generally of the following form

$$S(\omega) = |\omega| \cdot F(\omega) \cdot \chi_M(\omega),$$

where S is defined by equation (22), F is a filter function that determines its properties, constant $M > 0$ sets cut-off frequency, and function χ_M is defined as a characteristic function of the interval $[-M, M]$

$$\chi_M(\omega) = \begin{cases} 1, & |\omega| \leq M, \\ 0, & |\omega| > M. \end{cases}$$

The low-pass filters widely used in imaging are listed below (see [28, 29] for details). In *the Ram-Lak filter* [29] the function F is replaced with the constant function

$$F \equiv 1,$$

therefore the filter is defined as

$$S(\omega) = |\omega| \cdot \chi_M(\omega) = \begin{cases} |\omega|, & |\omega| \leq M, \\ 0, & |\omega| > M. \end{cases}$$

The Hann (Hanning) filter [30] is as follows

$$S(\omega) = |\omega| \cdot \frac{1}{2}\left(1 + \cos\frac{2\pi\omega}{M}\right) \cdot \chi_M(\omega),$$

where the function

$$F(\omega) = \frac{1}{2}\left(1 + \cos\frac{2\pi\omega}{M}\right),$$

is called the Hann window.

The Hamming filter [30] utilizes the function

$$F(\omega) = \frac{25}{46} + \frac{21}{46}\cos\frac{2\pi\omega}{M},$$

called the Hamming window. Hence, this filter is defined as follows

$$S(\omega) = |\omega| \cdot \left(\frac{25}{46} + \frac{21}{46} \cos \frac{2\pi\omega}{M} \right) \cdot \chi_M(\omega).$$

The Shepp-Logan filter [31] has the following representation

$$S(\omega) = |\omega| \cdot \frac{\sin \frac{\pi\omega}{2M}}{\frac{\pi\omega}{2M}} \cdot \chi_M(\omega) = \begin{cases} \frac{2M}{\pi} \left| \sin \frac{\pi\omega}{2M} \right|, & |\omega| \leq M, \\ 0, & |\omega| > M. \end{cases}$$

This filter results in the least smoothing and has the highest resolution [27].

The low-pass cosine filter [32] is defined as follows

$$S(\omega) = |\omega| \cdot \cos \frac{\pi\omega}{2M} \cdot \chi_M(\omega).$$

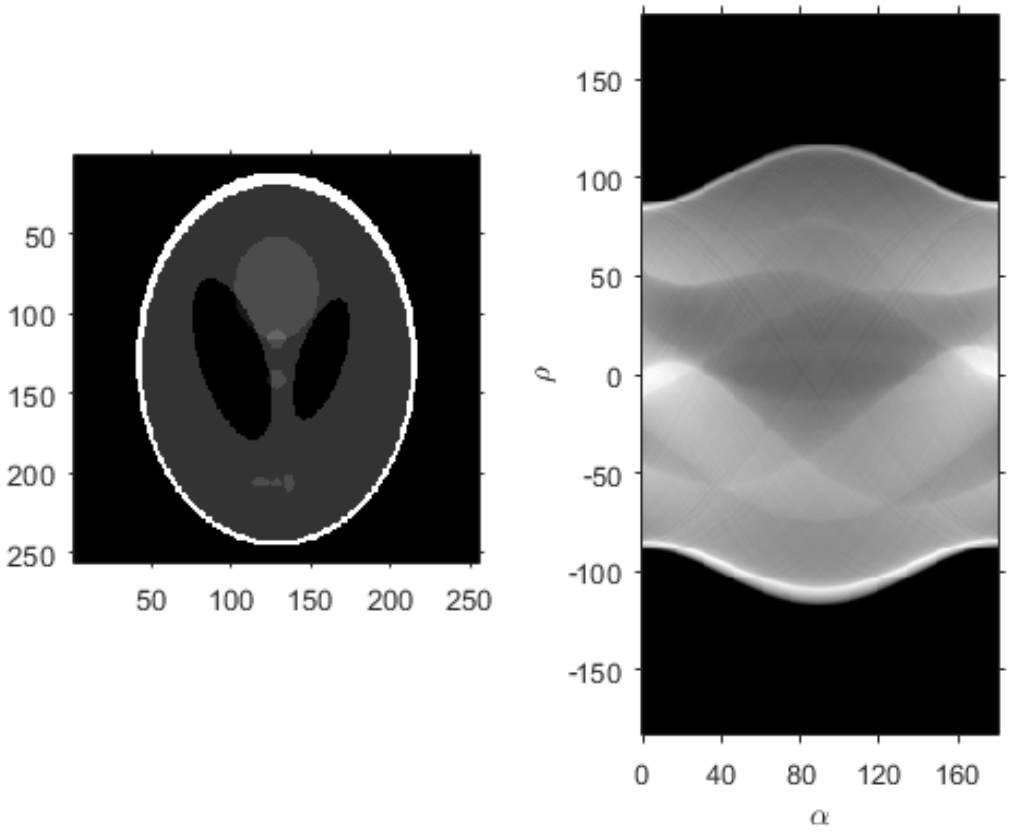
The effectiveness of the low-pass filters discussed in this section with different number of projections is shown in Fig. 5 (see Section 2.8 for details about usage of these filters in MATLAB).

The low-pass filters may result in high degree smoothing of an image leading to contrast loss [27]. On that ground enhancement filters are introduced in imaging. These filters intensify the signal and reduce noise simultaneously. The examples of enhancement filters are the Metz [33] and Wiener [34] filters that are combinations of the inverse (responsible for resolution recovery) and low-pass (suppress the noise) filters.

2.8 Radon transform in MATLAB

While passing through an object, an X-ray beam loses its intensity. The density of an object can be assessed by evaluating the initial and final intensities of a beam. A denser object causes a greater decrease of intensity of the beam than a less dense object. The intensity losses can be given as a grayscale value. A value of 0 (black) refers to zero alteration in intensity whereas a value of 1 (white) represents the beam being completely absorbed by an object. The collection of these values between 0 and 1 is called a sinogram (Fig. 3).

In a sinogram the horizontal axis corresponds to the angle at which slice is measured,



(a) (b)

Figure 3. Modified Shepp-Logan phantom (a) and its sinogram (b).The phantom is a simplified representation of a human head comprising one large ellipse imitating the brain and several smaller ellipses imitating the brain features. The sinogram shows a change in intensity for a given angle and distance: the horizontal axis corresponds to the angle α at which slice is measured, and the vertical axis corresponds to the distance ρ between an X-ray beam and the origin.

and the vertical axis corresponds to the distance between various beams and the origin (radial distance). In other words, a particular point of a sinogram shows an alteration in intensity for a given angle and distance. Fig. 3 shows an example of a sinogram obtained from the modified Shepp-Logan phantom (the Shepp-Logan phantom [31] with improved contrast for better visual perception) image using a MATLAB function `radon`. This function returns the Radon transform R of the intensity image for angles specified by an input vector parameter. As shown in Fig. 4, each element of the matrix R is the projection of the image intensity along a radial line oriented at a specific angle. In other words, the columns of the matrix R relate to angles at which projections are computed and the rows refer to the corresponding radial lines along which the projections are computed. The number of radial lines can be specified as an input parameter, otherwise, it is set by default.

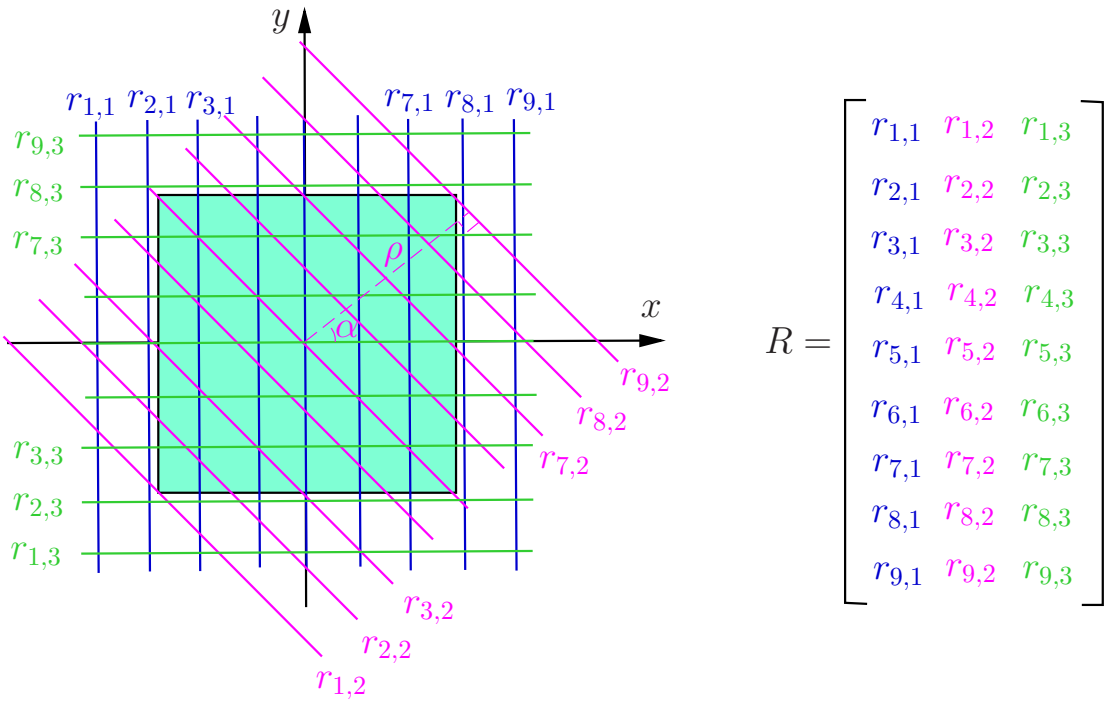


Figure 4. Radon transform in MATLAB. The `radon` function takes an intensity image and returns its Radon transform in the form of matrix R with columns corresponding to angles α at which the projections are computed (vector of angles is additionally specified as an input parameter). In this particular example, the first, second and third columns contain the projections taken at angles $0°$, $45°$ and $90°$, respectively. The rows of matrix R refer to the projections taken at different radial distances ρ. The number of radial distances (equal to the number of rows in matrix R) can be specified as a parameter, otherwise, it is set by default. The shown arrangement of projections is called the parallel scanning geometry, since projection lines (parallel and evenly spaced) are taken for a number of equally distributed directions.

It should be noted that the implementation of `radon` function corresponds to parallel scanning geometry, i.e. projections are arranged as a set of equally spaced parallel lines taken for a number of evenly distributed directions [1], as shown in Fig. 4. In practice, such scanning geometry means that a single X-ray source and corresponding CT detector move in parallel and rotate in accordance with a given set of angles.

The modified Shepp-Logan phantom in Fig. 3(a) is generated using a MATLAB function `phantom`. This function creates a grayscale intensity image of a head phantom that consists of one large ellipse (representing the brain) and several smaller ellipses (representing features in the brain). This phantom is most commonly used in biomedical image reconstruction [35].

In MATLAB the inverse Radon transform is performed by the function `iradon`, its input parameters are projection data and the angles in degrees at which the projections were taken. The function `iradon` uses the filtered backprojection algorithm described in Section 2.5. The Ram-Lak filter (see Section 2.6 for details) is the default. Other low-pass filters for backprojection algorithm (e.g. the Hann filter, the Hamming filter, the Shepp-Logan filter, the cosine filter) can be specified as parameters. The images reconstructed from the phantom sinogram (Fig. 3(b)), using `iradon` function with different filters, are presented in Fig. 5.

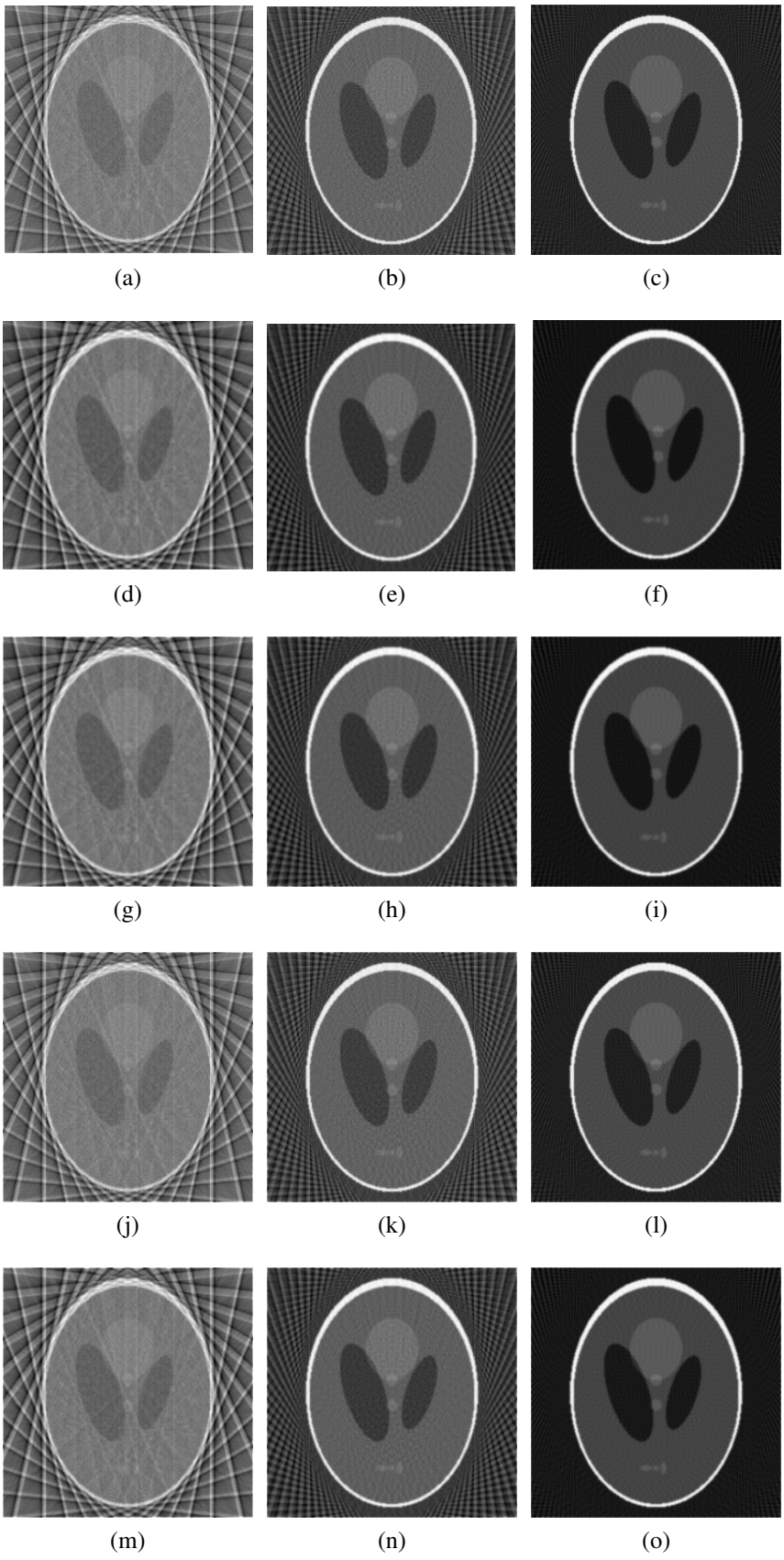


Figure 5. The modified Shepp-Logan phantom reconstructions obtained using different low-pass filters: the Ram-Lak filter (a)–(c), the Hann filter (d)–(f), the Hamming filter (g)–(i), the Shepp-Logan filter (j)–(l), the cosine filter (m)–(o). Images in the left, middle and right columns correspond to 15, 45 and 90 projections respectively.

3 DEEP NEURAL NETWORKS

3.1 Biological and artificial neurons

Artificial neural networks (ANNs) are an example of mathematical construction inspired by neuroscience [36]. These networks consist of processing units called artificial neurons that resemble biological neurons in a brain. A biological neuron contains dendrites that receive input signals from the neighbouring neurons and pass them to the neuron cell body (soma), where these signals are accumulated (Fig. 6). If the input signal is strong enough (exceeds threshold limit) neuron activates and its axon transmits the signal to the dendrites of other neighbouring neurons.

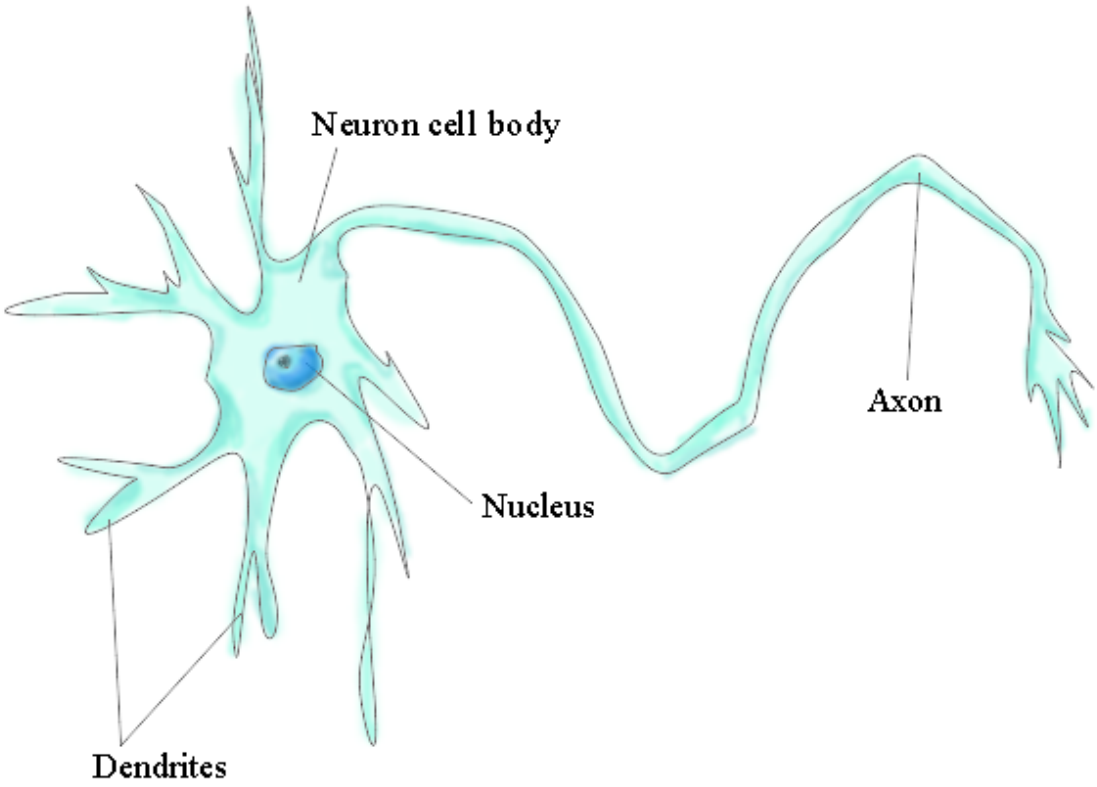


Figure 6. Structure of a biological neuron. A typical neuron consists of a cell body (soma), dendrites and a single axon. Dendrites receive input signals from other neurons, next the signals are summed up in the cell body and finally (if the neuron was activated) passed through the axon to the neighbouring neurons.

An artificial neuron has the structure similar to that of a biological neuron (Fig. 7). The

input connections of an artificial neuron play the role of dendrites, accepting the input signals from other neighbouring neurons. Let $\vec{x'} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ denote a vector of the input signals and $\vec{\omega'} = (\omega_1, \ldots, \omega_n) \in \mathbb{R}^n$ denote a vector of weights. Firstly, in the neuron body, the signals $\vec{x'}$ are summed up with weights $\vec{\omega'}$, and then an extra constant ω_0 known as bias is added as follows

$$\sum_{i=1}^{n} x_i \omega_i + \omega_0. \tag{23}$$

Often for convenience the first dummy coordinate x_0 (always equal to 1) is added to vector $\vec{x'}$, then linear combination (23) can be rewritten in shorter notation as the following inner product of vectors $\vec{x} = (1, x_1, \ldots, x_n) \in \mathbb{R}^{n+1}$ and $\vec{\omega} = (\omega_0, \omega_1, \ldots, \omega_n) \in \mathbb{R}^{n+1}$

$$\langle \vec{x}, \vec{\omega} \rangle = \sum_{i=0}^{n} x_i \omega_i = \vec{\omega}^\tau \vec{x}. \tag{24}$$

Next, the non-linear activation function $f : \mathbb{R} \to \mathbb{R}$ operates on linear combination (24) to calculate the neuron output

$$y = f(\vec{\omega}^\tau \vec{x}), \tag{25}$$

based on input received (see [36] for details).

The initial neurobiology inspiration to artificial neural networks dates back to the 1940s when Warren McCulloch and Walter Pitts proposed a concept of the first artificial neural network [37]. Their neuron model has n excitatory binary inputs $\vec{x} = (x_1, \ldots, x_n)$, where $x_k \in \{0, 1\}$, $k = 1, ..., n$, and a single binary inhibitory input $i \in \{0, 1\}$, where $i = 1$ means that the neuron cannot fire, and $i = 0$ stands for the opposite (the inhibition rule). The model has a binary output $y \in \{0, 1\}$, where $y = 0$ shows that the neuron is at rest and $y = 1$ that it is excited. A threshold value ζ is used to decide whether the neuron fires: if the cumulative input signal exceeds ζ, then the neuron is activated, otherwise it stays at rest. Thus, the output signal y can be calculated as follows

$$y(\vec{x}) = \begin{cases} 1, & \sum_{k=1}^{n} x_k > \zeta \text{ and } i = 0, \\ 0, & \text{otherwise.} \end{cases}$$

The McCulloch-Pitts artificial neurons can be used to represent logical functions such as AND, NOT and OR. It is also possible to implement more sophisticated functions such as flip-flop by combining multiple neurons. Despite the versatility of the model the McCulloch-Pitts neurons have considerable shortcomings [38], namely, they are not

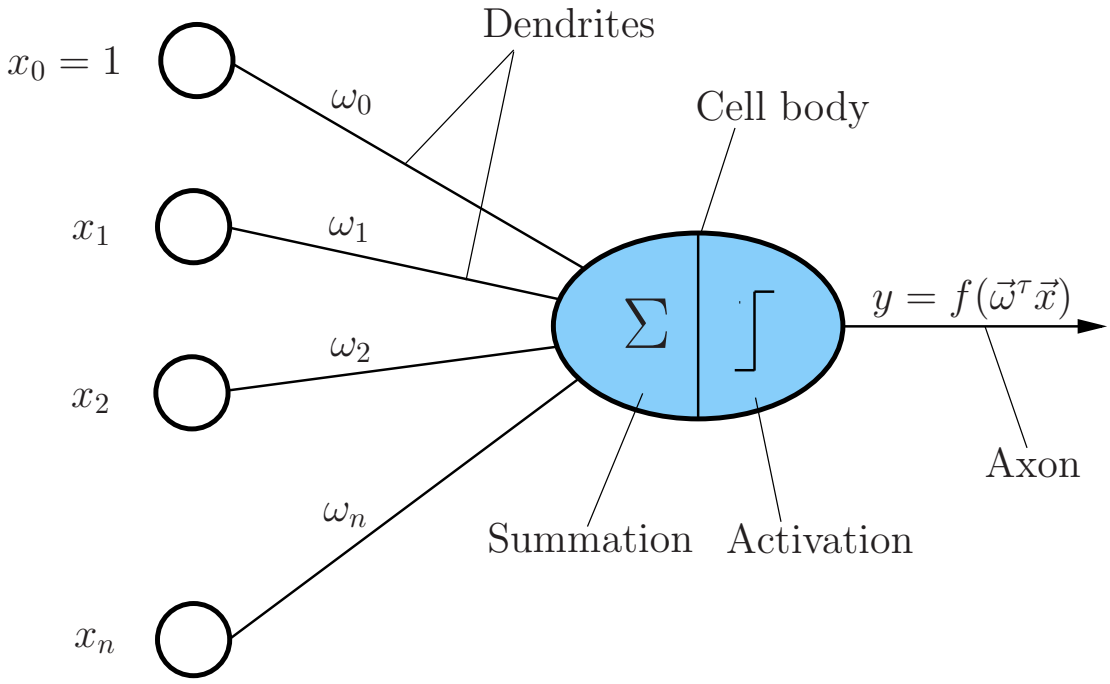


Figure 7. Structure of an artificial neuron. Input signals x_1 through x_n are summed up with weights ω_1 through ω_n, then a bias ω_0 is added. Next, the activation function f operates on the obtained linear combination $\vec{\omega}^\tau \vec{x}$ to produce the neuron output y.

designed to be trained (i.e. weights cannot be learned from data); all weights are assumed to be equal to unity, this implies that all the inputs make an equal contribution to the output.

3.2 Perceptron

Modern perceptrons are largely based on the first construction of a linear perceptron proposed by Rosenblatt in the 1950s [39]. Rosenblatt's perceptron was inspired by Donald O. Hebb's theory of synaptic plasticity [40]. This theory describes the basic principle of biological neuron training: the more often the connection between two neurons is used, the stronger it becomes.

Rosenblatt's perceptron is essentially a linear binary classification model. The term *binary* means that the model allows classifying the input data as belonging to one of two classes (the classes usually have labels 0, 1, or equivalently -1, 1), while the term *linear* implies that the model splits the data into two classes by a hyperplane (a line in the two-dimensional case), and the rule in adopting decisions (to which of two classes the input

data belong) depends linearly on the input data.

Let $\vec{x'} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ denote the input data represented as a real vector, and $y(\vec{x'}) \in \{-1, 1\}$ denote the corresponding output . Then the linear classification model seeks for the values of weights $\vec{\omega} = (\omega_0, \omega_1, \ldots, \omega_n) \in \mathbb{R}^{n+1}$ such that the sign of the linear function

$$\hat{y}(\vec{x'}, \vec{\omega}) = \text{sgn}(\omega_0 + \omega_1 x_1 + \ldots + \omega_n x_n), \tag{26}$$

coincide with the sign of the correct answer $y(\vec{x'})$ as often as possible. If the first dummy coordinate $x_0 = 1$ is added to the vector $\vec{x'}$ in a manner described in Section 3.1, then equation (26) can be rewritten as

$$\hat{y}(\vec{x}, \vec{\omega}) = \text{sgn}(\vec{\omega}^\tau \vec{x}),$$

where $\vec{x} = (1, x_1, \ldots, x_n) \in \mathbb{R}^{n+1}$.

Let $\mathcal{X}$ denote the training set (the dataset used to fit the model) of N elements. Rosenblatt's perceptron utilizes the following error function

$$E_P(\vec{\omega}) = -\sum_{\vec{x} \in \mathcal{M}} y(\vec{x})(\vec{\omega}^\tau \vec{x}),$$

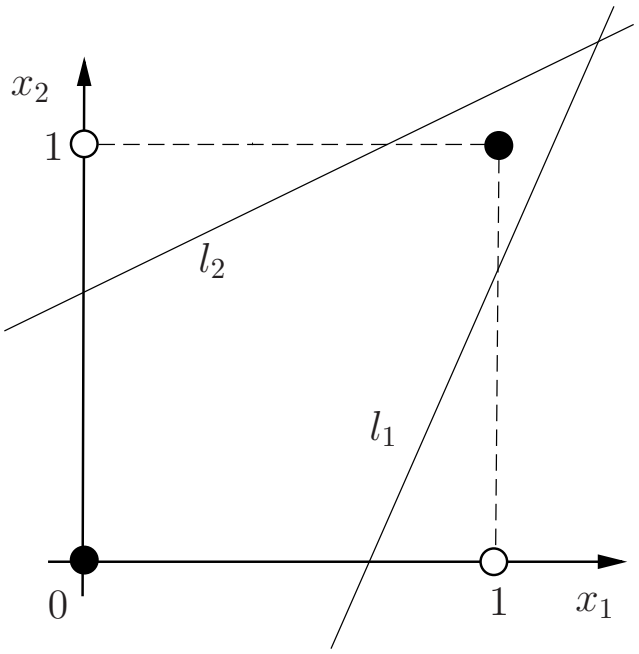
where $\mathcal{M} \subset \mathcal{X}$ is a set of input data which perceptron with the weighs $\vec{\omega}$ classifies incorrectly.

The error function can be minimized by the gradient descent method according to the Perceptron learning rule

$$\vec{\omega}_{old} = \vec{\omega}_{new} - \eta \nabla_{\vec{\omega}} E_P(\vec{\omega}),$$

where η is a learning rate parameter. In other words, the algorithm goes through the elements $\vec{x}_1$, $\vec{x}_2$, ..., $\vec{x}_N$, from the training set and for each $\vec{x}_k$ if it is classified correctly, does not change weights $\vec{\omega}$, otherwise adds term $\eta \cdot y(\vec{x}_n) \cdot \vec{x}_n$ to weights $\vec{\omega}$.

In 1969, Marvin Minsky and Seymour Papert showed the significant limitations of Rosenblatt's perceptron [41]. Particularly, the perceptron is able to classify the data as belonging to one of two sets only if these two sets are linearly separable, i.e. they can be separated by a single line (in two-dimensional case). For example, Rosenblatt's perceptron cannot be trained to solve the XOR (exclusive-OR) problem, i.e. to predict the outputs of XOR logical function given two binary inputs. That happens because XOR inputs are not linearly separable (cannot be separated by a single line), as shown in Fig. 8.



x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Figure 8. Logical exclusive-OR (XOR) function: its graphical representation (left) and truth table (right). Two classes of the XOR values (black and white bullets stand for 0 and 1 respectively) cannot be separated with a single line but can be logically classified with two lines l_1 and l_2.

To overcome this limitation, modern perceptrons are equipped with non-linear activation functions that are discussed in detail in Section 3.3. Thus, modern perceptrons have the structure shown in Fig. 7: the inputs are summed up with weights, and then the non-linear activation function is applied.

3.3 Activation functions

An activation function is an integral part of any perceptron (or neuron) that determines whether it is activated based on the input of the perceptron. For neural units, there exist linear and non-linear activation functions, the usage of a specific function depends on the problem to be solved and the network architecture.

If a neuron uses *a linear activation function* its output is linearly dependent on the input, as shown in Fig. 9. Hence, the range of the function is $(-\infty, +\infty)$. Linear activation functions are not used in modern neural networks, since they have two significant limitations [39]. Firstly, the derivative of a linear function is constant, consequently it is not possible to use backpropagation (or gradient descent) that is the main learning technique nowadays (see Section 3.4). Secondly, a linear activation function turns any multi-layered

network into one-layered network, since a linear combination of any number of linear functions is a linear function.

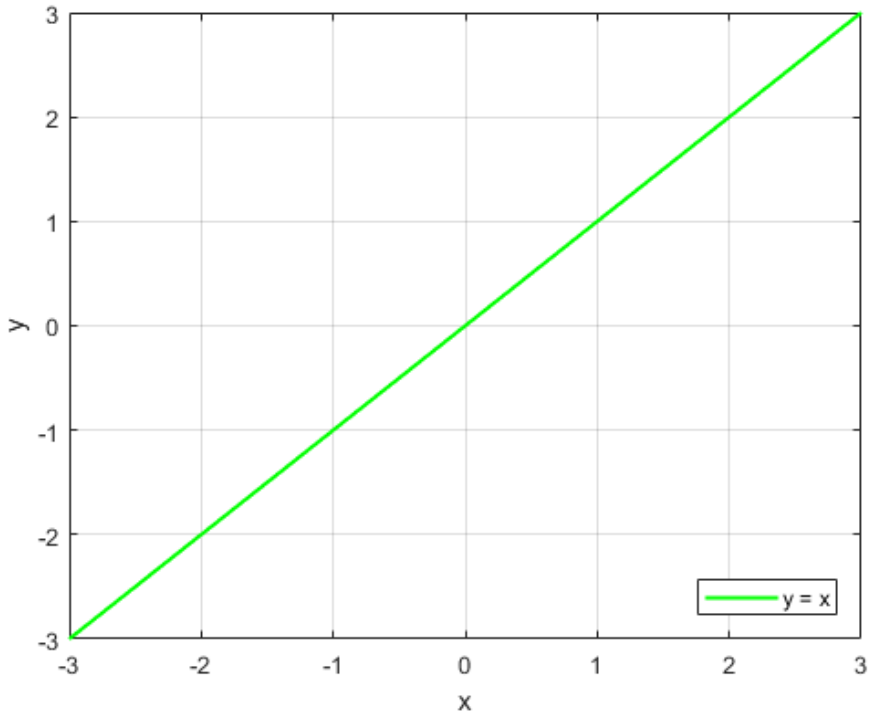


Figure 9. Linear activation function. The output signal y is proportional to the input x.

In most modern networks, non-linear activation functions are used [39]. Non-linearity allows the artificial neural network to learn more intricate features from non-linear or high-dimensional data through creating complex mapping between the input and output.

A logistic sigmoid activation function is as follows

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

As shown in Fig. 10, logistic sigmoid is a continuous monotonically increasing function with the range between 0 and 1: it tends to 0 as x tends to $-\infty$ and to 1 as x tends to $+\infty$. The function has continuous derivative $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ that is important for training a neural network. However, the main drawback of sigmoid is that for very low and high arguments the values of the function do not change significantly that causes a vanishing gradient problem. This issue prevents network from further learning, since at some point the gradient is extremely small and weights do not change.

The properties of *hyperbolic tangent*

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

are similar to those of sigmoid activation function. It is continuous and bounded, but as x tends to $-\infty$ it tends to -1 (not to 0 like sigmoid), as shown in Fig. 10. Its derivative is $\tanh'(x) = 1 - \tanh^2(x)$. Hyperbolic tangent approaches its asymptotes faster than sigmoid. One of the advantages of hyperbolic tangent over the sigmoid is that it maps zero values near zero and negative input into strongly negative values. However, hyperbolic tangent is not able to solve the vanishing gradient problem as well as sigmoid.

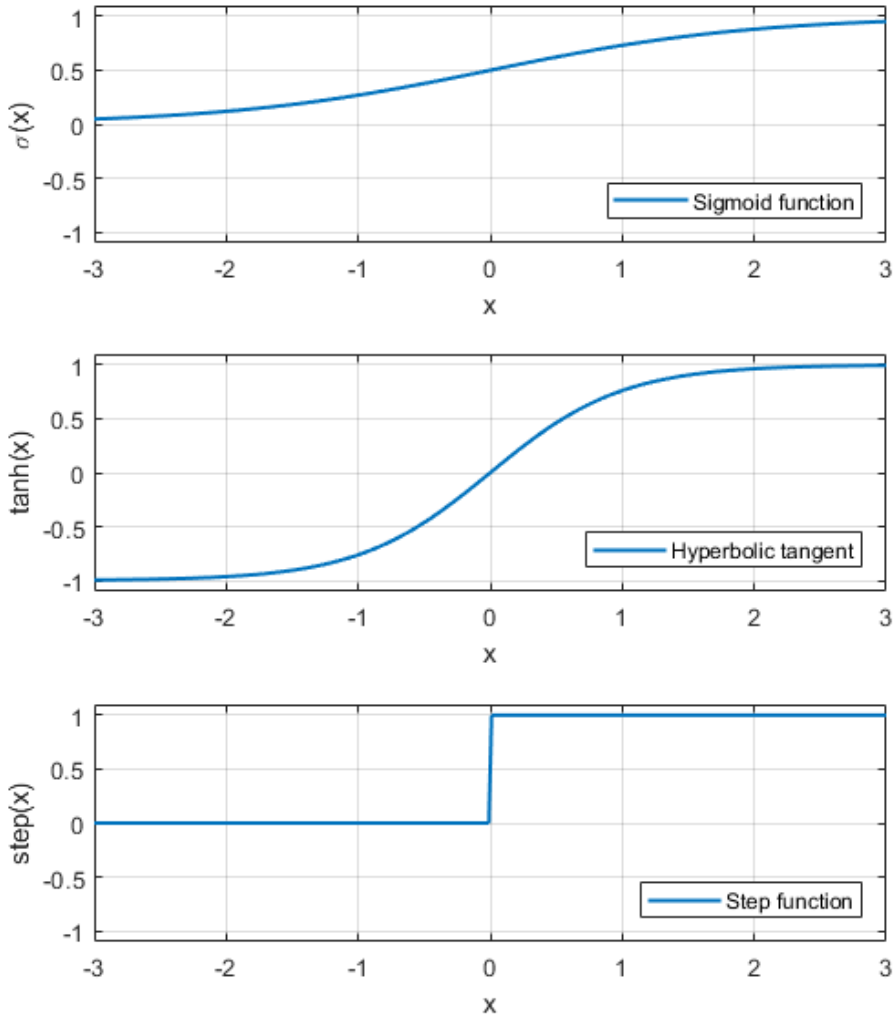


Figure 10. Activation functions. From the top down: logistic sigmoid $\sigma(x)$, hyperbolic tangent $\tanh(x)$ and Heaviside step(x) function.

Another function that can be used as activation is *the Heaviside step function*

$$\text{step}(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases}$$

It is a threshold-based activation function which means that the neuron is activated and its output is 1 if its input value is above a certain threshold, otherwise the output value is equal to zero and neuron is at rest (Fig. 10). The main shortcoming of this activation function is that it allows only binary outputs (0 or 1), thus it cannot implement classifying the inputs into more than two categories.

Another activation function is *a rectifier* that is employed by a rectified linear unit (ReLU)

$$\text{ReLU}(x) = \begin{cases} 0, & x < 0, \\ x, & x \geq 0, \end{cases}$$

or equivalently $\text{ReLU}(x) = \max\{0, x\}$. Neurons with ReLUs are more computationally efficient than those based on the logistic sigmoid or hyperbolic tangent, since to compute the derivative $\text{ReLU}'(x)$ only one comparison is needed: if x is less than zero the derivative is equal to 0, otherwise it is equal to 1. An important advantage of rectifier is that, unlike three previous non-linear functions, it allows distinguishing between strongly and slightly activated neurons. However, the shortcoming of assuming zero for all the negative values is a situation when a ReLU neuron produce zero output for any input and becomes inactive forever (the Dying ReLU problem).

Some modifications of ReLU are designed to solve this issue. For example, *Leaky ReLU (LReLU)* [42] has a slight positive slope for the negative arguments and coincides with ReLU for the positive input values

$$\text{LReLU}(x) = \begin{cases} \alpha x, & x < 0, \\ x, & x \geq 0, \end{cases}$$

where α is a small positive constant. For instance, Fig. 11 demonstrates LReLU with parameter $\alpha = 0.1$. Due to its constructions, Leaky ReLU enables learning through backpropagation even for the negative inputs and thus solves the Dying ReLU problem.

The idea of LReLU evolved to the Parametric ReLU (PReLU) [43] that has the same construction, but the constant α can be learnt for every particular dataset that results in better performance [39].

Another modification of a rectifier is *an Exponential linear unit (ELU)* [44]

$$\mathrm{ELU}(x) = \begin{cases} \alpha(e^x - 1), & x < 0, \\ x, & x \geq 0, \end{cases}$$

that is an exponential function for negative inputs and linear for the positive arguments. Parameter α is an arbitrary chosen small positive constant.

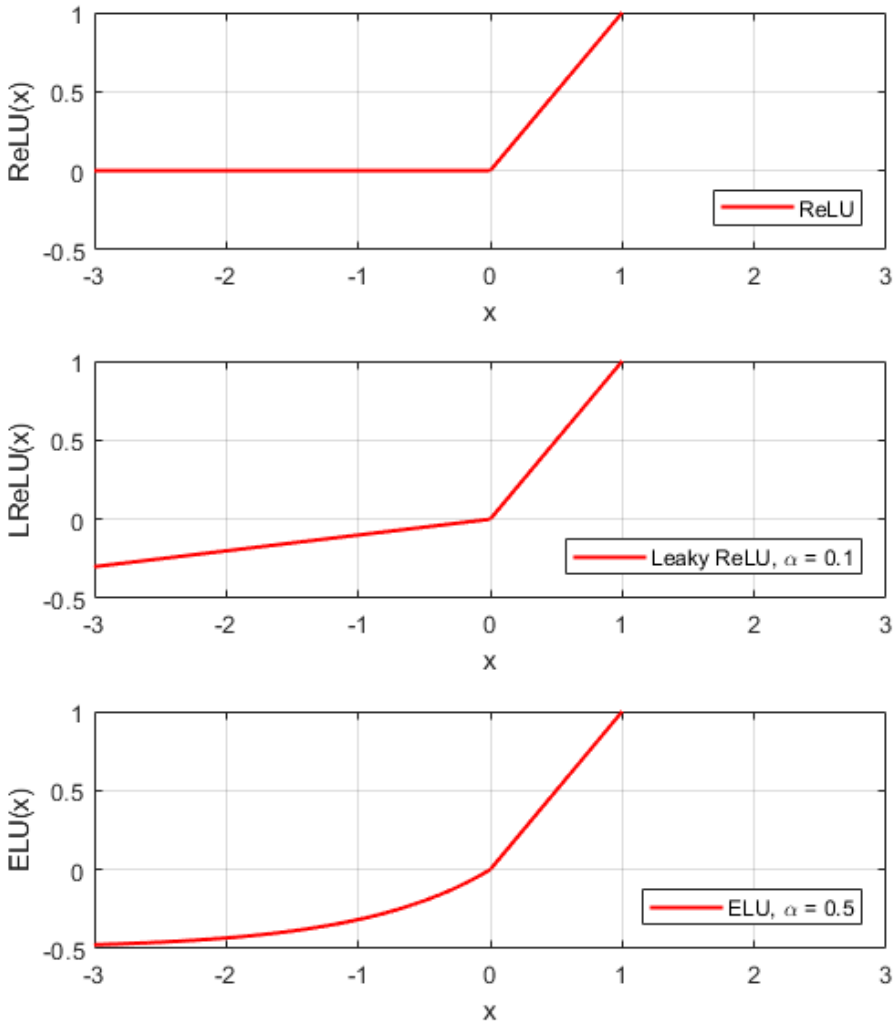


Figure 11. Rectifier activation function and its modifications. From the top down: rectified linear unit (ReLU), Leaky ReLU with $\alpha = 0.1$ and exponential linear unit (ELU) with $\alpha = 0.5$.

3.4 Multilayer perceptron and its learning rule

A multilayer perceptron (MLP) is an artificial neural network consisting of an input layer, one or more hidden layers and an output layer. For example, the MLP with two hidden layers is shown in Fig. 12. It should be noted that the MLP refers to fully connected networks, that is, each neuron in the preceding layer is connected to all neurons in the next layer. In contrast to a linear perceptron, an MLP has non-linear activation functions. Non-linearity enables an MLP to differentiate between the data that are not linearly separable.

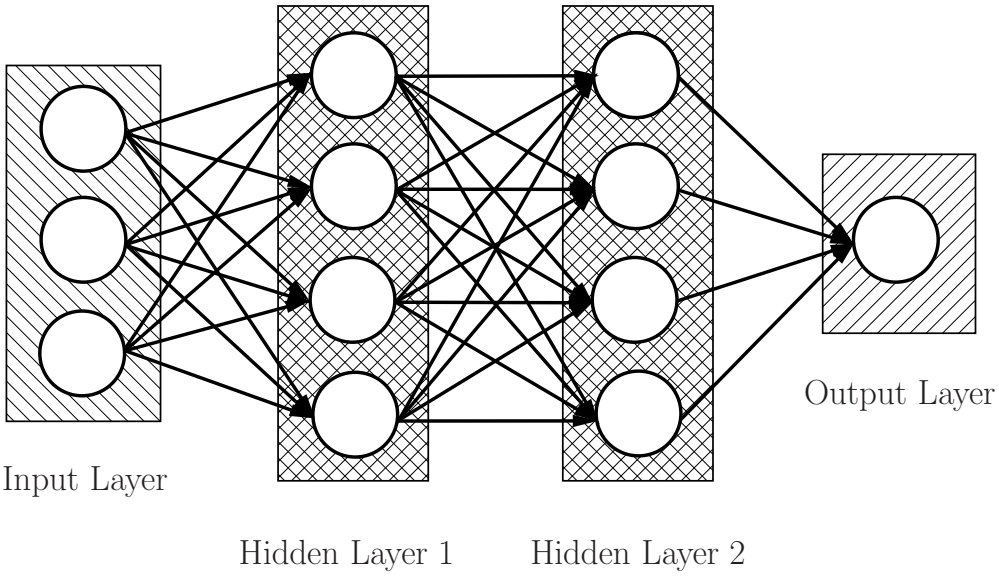


Figure 12. A multilayer perceptron (an artificial neural network) with an input layer, two hidden layers, and an output layer. Since all neurons in one layer are connected to all neurons in the next layer, this neural network is fully connected.

The Perceptron learning rule used for training a linear perceptron is not applicable to learning an MPL model owing to its different structure. A commonly used technique for training an MLP is to design a loss function (or error) according to, for example, the error of misclassification and next, minimize this function with respect to the model weights [36].

In most cases optimization and hence training a neural network is based on algorithm using the gradient to minimize the loss function [45]. The main of these algorithms is a gradient descent method, all the others are its various modifications.

In a classification problem, a training set D composed of pairs $(\vec{x}, \vec{y})$ is considered, where vector $\vec{x}$ represents the input data and $\vec{y}$ is a correct classification response. The neural network with weights $\vec{\theta}$ makes predictions $f(\vec{x}, \vec{\theta})$, where $\vec{x} \in D$, and the error function $E(f(\vec{x}, \vec{\theta}), \vec{y})$ is defined for all the elements in the training set. Then, the total error function is a sum of errors for all elements in the training set

$$E(\vec{\theta}) = \sum_{(\vec{x},\vec{y})\in D} E(f(\vec{x}, \vec{\theta}), \vec{y}).$$

One step of the gradient descent method is implied according to the following rule

$$\vec{\theta}_t = \vec{\theta}_{t-1} - \eta \nabla E(\vec{\theta}_{t-1}) = \vec{\theta}_{t-1} - \eta \sum_{(\vec{x},\vec{y})\in D} \nabla E(f(\vec{x}, \vec{\theta}_{t-1}), \vec{y}),$$

where $\vec{\theta}_t$ is an updated vector of weights and parameter η is a learning rate that regulates the size of the step in the method so that the weights converge rapidly to a correct response.

However, one step of the gradient descent algorithm requires going through the whole training set that is not cost-efficient, therefore practically stochastic gradient descent is used, that is, weights are updated after going through each element in the training set rather than the whole set

$$\vec{\theta}_t = \vec{\theta}_{t-1} - \eta \nabla E(f(\vec{x}_t, \vec{\theta}_{t-1}), \vec{y}_t),$$

thus, this modification fastens the training process.

The method called backpropagation is applied to simplify the computation of gradient. This method propagates the error from the output layer backward so that the gradients at the preceding layers can be found using the chain rule of derivatives for the function composition [36].

A useful tool to represent a composite function as a composition of simple functions is a directed acyclic graph called computation graph [39]. In this graph vertices (nodes) correspond to functions, while edges connect functions and their arguments. The process of transmitting values from the input nodes to the output is known as forward pass (or forward propagation).

Fig. 13(a) shows a computation graph for a function $f(x, y) = 3x^3 + 5xy^2$ and also partial

derivatives of the nodes with respect to their inputs. Backpropagation and computation graph allow for calculation of all partial derivatives of the function $f(x, y)$ using the chain rule of derivatives, as shown in Fig. 13(b). In this case, the computation of derivatives starts from the output node with partial derivative $\frac{\partial f}{\partial f} = 1$ and moves in the direction opposite to the edges of a graph. For example, a partial derivative of f with respect to variable x is calculated as follows

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial c}\frac{\partial c}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial f}{\partial e}\frac{\partial e}{\partial d} = 9x^2 + 5y^2.$$

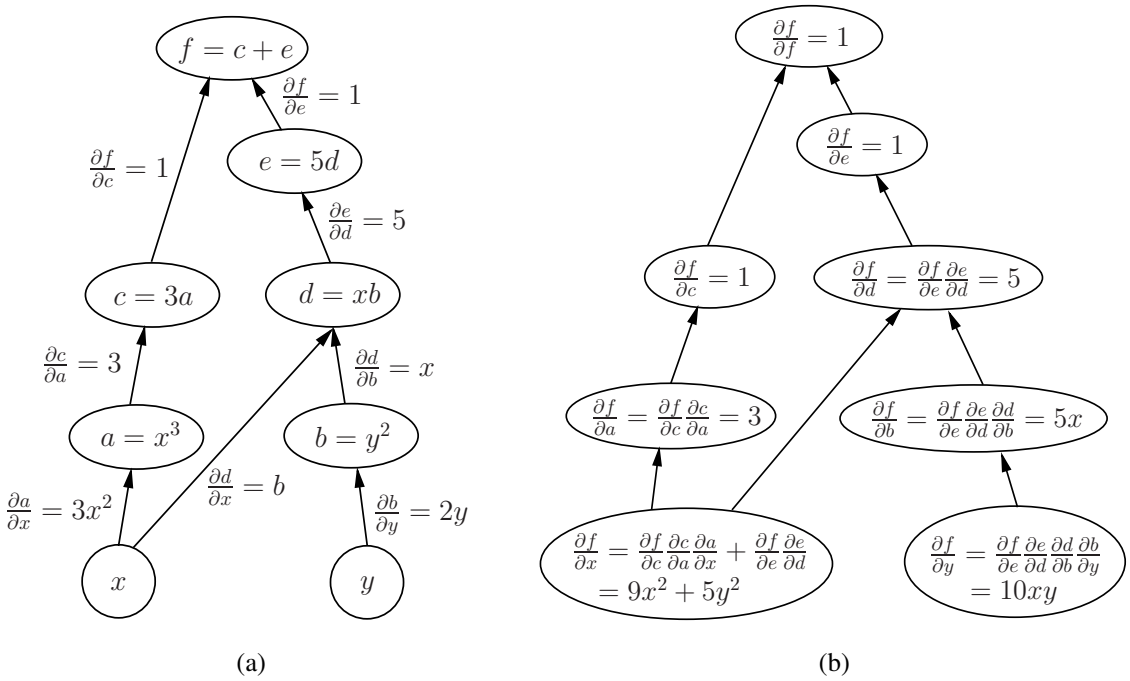


Figure 13. Computation graph of the function $f(x, y) = 3x^3 + 5xy^2$ together with its partial derivatives (a), and algorithm of finding all partial derivatives of the function $f(x, y)$ by backpropagation algorithm (b).

In general, the backpropagation algorithm for computation partial derivatives can be described as follows. Let $G = (V, E)$ denote some computation graph for a function $f(x_1, x_2, \ldots, x_n)$, where V is a set of vertices and E is a set of edges. Functions $g \in V$ are vertices of the graph, some vertices correspond to the input variables $x_1, x_2, \ldots, x_n$ and do not have input edges, and one vertex corresponding to the function f does not have output edges. In order to find partial derivatives of $f \in V$, it is necessary to initialize $\frac{\partial f}{\partial f} = 1$ and then for each vertex $g \in V$ that has 'children' (vertices into which the edges

go from it) already processed by the algorithm, the following equation is calculated

$$\frac{\partial f}{\partial g} = \sum_{\tilde{g} \in \text{Children}(g)} \frac{\partial f}{\partial \tilde{g}} \frac{\partial \tilde{g}}{\partial g}.$$

When the input vertices $x_1, x_2, \ldots, x_n$ are reached, the partial derivatives $\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}$ are computed and thus the gradient of the function f can be obtained

$$\nabla_{\vec{x}} f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix},$$

where $\vec{x} = (x_1, x_2, \ldots, x_n)$.

To sum up, the ANN learning is an iterative process consisting of forward pass and backpropagation [36]. In the forward pass, the input and output at each neuron in each layer are calculated based on the training data. When all the neurons have made their computations, the error (loss function) is estimated in the output layer based on the predicted and correct results. Next, the error is propagated to all the neurons of the hidden layers starting from the output layer. After spreading the information back to all neurons, the gradients are computed through backpropagation and then the weights are updated by the gradient descent method.

3.5 Adaptive gradient descent methods

In the traditional gradient descent method, the update step depends only on the current value of gradient and learning rate η, but does not take into account the update history of every single parameter. Unlike this approach, the adaptive optimization methods allow for automatical adjustment of the learning rate for different parameters.

An adaptive gradient method called *Adagrad* is based on the idea that the update step should be higher for those parameters that vary to great extent in the data, while smaller step accounts for less volatile parameters [46]. If $g_{t,i}$ denotes the gradient of the loss function with respect to the parameter θ_i, i.e.

$$g_{t,i} = \nabla_{\theta_i} L(\theta),$$

then the update rule for the parameter θ_i in Adagrad is as follows

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \cdot g_{t,i}, \tag{27}$$

where parameter ε allows eliminating division by zero and G_t is a diagonal matrix where each element $G_{t,ii}$ is a sum of squared gradients of the corresponding parameters for the previous steps, that is,

$$G_{t,ii} = G_{t-1,ii} + g_{t,i}^2.$$

Adadelta is a modification of Adagrad algorithm [47]. Instead of accumulating a sum of all past gradients squared, Adadelta adjusts learning rates according to a moving window of gradient updates that is set by choosing a decay factor $\rho < 1$ and then the matrix G_t is as follows

$$G_{t,ii} = \rho G_{t-1,ii} + (1 - \rho) g_{t,i}^2.$$

The update step of Adadelta is the same as that of Adagrad (see equation (27)).

Another adaptive optimization algorithm called *Adam* was designed specially for training DNNs [48]. The algorithm uses exponential moving averages of the gradient

$$m_t = \beta_1 m + (1 - \beta_1) g_t,$$

and the squared gradient

$$v_t = \beta_2 m + (1 - \beta_2) g_t^2,$$

where the parameters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of these moving averages. The Adam's update rule is as follows

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{v_t + \varepsilon}} \cdot m_t.$$

3.6 Convolutional neural networks

Convolutional neural networks (CNN) are the special kind of DNNs used for processing the data that have grid-like topology, i.e. data with spatially correlated points. For example, CNNs are successfully applied for processing images that can be considered as two-dimensional grids of pixels.

A black-and-white image can be represented as a two-dimensional matrix of size $M \times$

N, where M and N are width and height (in pixels), respectively. In general, image is considered as a three-dimensional matrix of size $M \times N \times d$, i.e. in each pixel of an image there is a d-dimensional vector with components called channels and thus the image is a set of d matrices of size $M \times N$. For example, one color image is usually represented by three matrices each corresponding to one color channel (RGB).

The name of CNN indicates that the network uses a convolution operation that is linear transformation of the input data [39]. Let X^l denotes the input of the l-th layer, then the result of two-dimensional convolution with a matrix of weights W (called convolution filter or kernel) of size $(2d+1) \times (2d+1)$ is as follows

$$Y^l_{i,j} = \sum_{-d \le a,b \le d} W_{a,b} X^l_{i+a,j+b}.$$

The result $Y^l_{i,j}$ of convolution in the l-th layer is called a feature map.

Let us consider the following example of convolution between the input image represented by (5×5)-matrix X (an element $X_{i,j}$ is a value of the pixel with number (i,j)) and (3×3)-matrix W of weights

$$X = \begin{bmatrix} 1 & 0 & 1 & 5 & 2 \\ 3 & 2 & 1 & 0 & 1 \\ 0 & 5 & 3 & 4 & 0 \\ 4 & 3 & 0 & 1 & 4 \\ 2 & 0 & 1 & 5 & 3 \end{bmatrix}, \; W = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}.$$

The result of convolution $Y = X * W$ depends on two parameters: stride and padding. Stride determines the number of pixels by which the filter W shifts. Concerning padding, if the output is restricted to only positions where the filter lies entirely within the image, then the convolution is said to have *valid padding* [45]. In this case, the result of convolution is always smaller than the size of the input image (Fig. 14). If after applying convolution the spatial dimensionality of an image remains unchanged, *same padding* is applied (Fig. 15). In this case, the zero padding supplements the input image around the border. If k is a size of the convolution kernel and the stride is equal to 1, then the width of zero padding ρ is calculated as

$$\rho = \frac{k-1}{2}. \tag{28}$$

For instance, in Fig. 15, the kernel size is $k = 3$, therefore the zero padding is $\rho = 1$. The

general formula for computing the size y of a feature map is

$$y = \frac{x - k + 2\rho + s}{s},$$

where x is a height (width) of an input image, k is a kernel size, ρ is a padding and s is a stride.

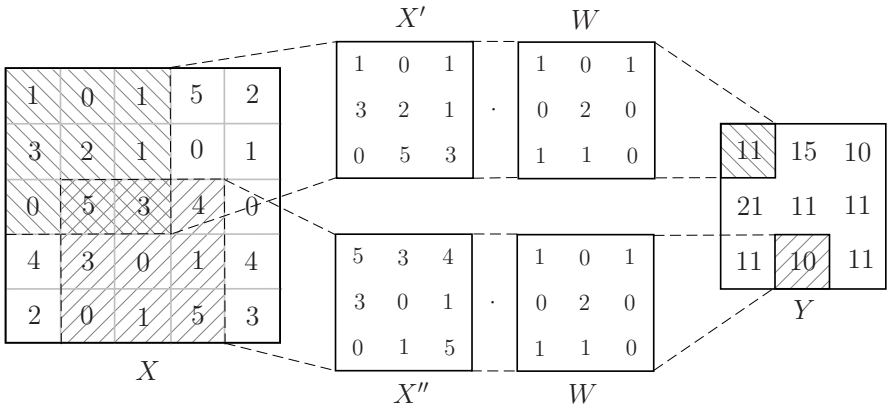


Figure 14. An example of convolution with the stride equal to 1 and *valid padding*. The filter W is convolving around an input image X, performing elementwise multiplication between the filter W and the portion of the image (e.g. X', X''). Next, the multiplications are summed up, producing one element in the feature map Y.

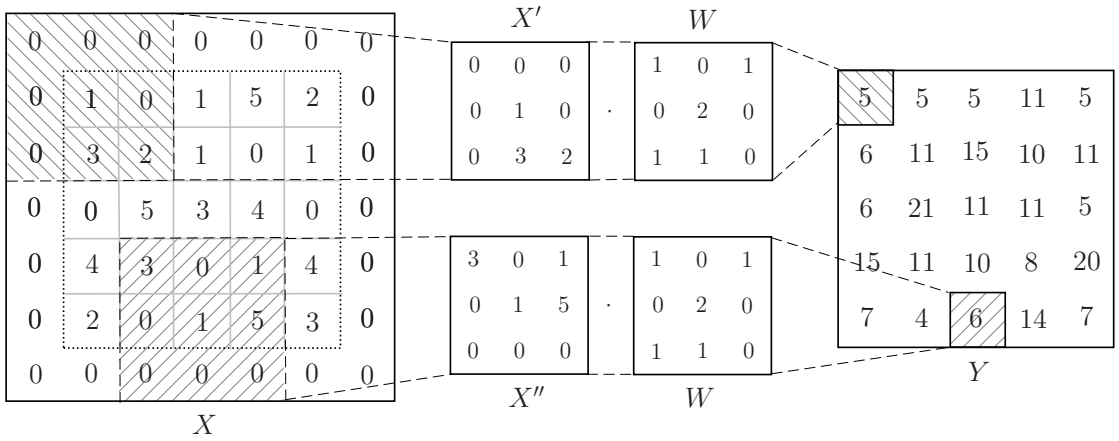


Figure 15. An example of convolution with the stride equal to 1 and *same padding*. The input image X is augmented by a zero padding of width ρ calculated according to equation (28). Next, the kernel W is hovering over the image to produce the feature map Y.

In the context of neural networks, the convolution operation is implemented as multiple convolutions applied in parallel [45], since the CNN input image is a three-dimensional matrix (two indices correspond to the spatial coordinates of a pixel and one index corresponds to the different channels in this pixel). Thus, it is important to specify a number of filters for convolution, a value of this parameter must correspond to the number of channels in the image to which the convolution filter is applied.

A typical layer of a CNN can be divided into three stages [45], as shown in Fig. 16. The first stage (convolution stage) performs convolutions in order to produce a set of linear activations. In the second stage (detector stage), linear activations are processed by some non-linear activation function h (e.g. ReLU) resulting in a feature map

$$Z_{i,j}^{l} = h(Y_{i,j}^{l}). \tag{29}$$

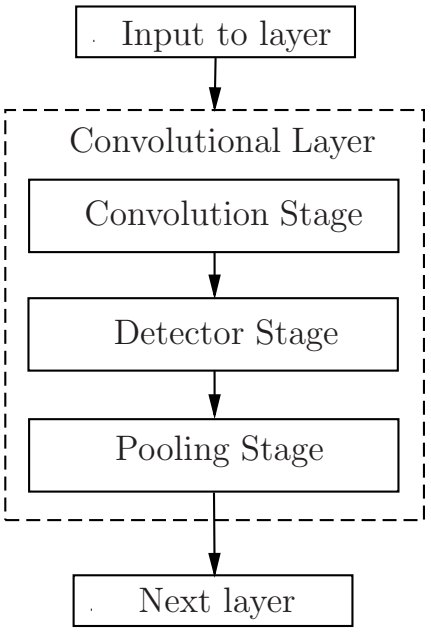


Figure 16. The components of a convolutional layer in a CNN. The convolution stage performs convolution of layer inputs and weights. In the detector stage, the results of the convolution operation are passed trough some non-linear activation function, producing a feature map. The pooling stage performs pooling to reduce the number of parameters (dimensionality).

In the third and last stage (pooling stage), a pooling layer is utilized to downsample (reduce the dimension) a feature map by summarizing its features. The most common pooling technique is max-pooling that slides a window of a given size over the input and

computes the maximum value in the window, i.e.

$$X_{i,j}^{l+1} = \max_{-d \leq a \leq d, -d \leq b \leq d} Z_{i+a,j+b}^{l},$$

where $Z_{i+a,j+b}^{l}$ is an output of the Detector Stage defined by equation (29).

The size of pooling operation is always smaller than the size of the feature map. If the size of pooling is 2×2 pixels with a stride of 2 pixels, this means that the pooling layer halves each dimension of feature map. For instance, the result of such pooling layer applied to feature map of 8×8 pixels is an output pooled feature map of 4×4 pixels.

An example of typical CNN architecture is AlexNet designed by Alex Krizhevsky et al. [49]. AlexNet is composed of 5 convolutional layers (some of them are followed by max-pooling) and 3 fully connected layers. In 2012 AlexNet won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), an annual contest launched by the ImageNet project [50], where various software algorithms compete to accurately detect and classify objects.

3.7 Autoencoders

An autoencoder is an ANN that learns to copy its input to its output [45]. The network consists of an encoder part mapping the input to the code (hidden layer), and a decoder part that produces a reconstruction of the original input, given the code (Fig. 17). The main purpose of the autoencoder is to extract important properties from the data rather than just copy the input. A possible way to obtain useful features from the network is to restrict the code to have a smaller dimension than the input, in this case autoencoder is called undercomplete.

In the overcomplete case (the dimension of the hidden code is greater than that of the input), the regularized autoencoders, such as sparse autoencoders and denoising autoencoders, are applied to prevent the network from directly coping its input to its output. In the sparse autoencoder, a penalty term is added to the loss function such that only a fraction of hidden nodes are active simultaneously [39]. The denoising autoencoder receives the input data corrupted by some random noise and the network must recover the original noise-free data rather than simply copying the input [45].

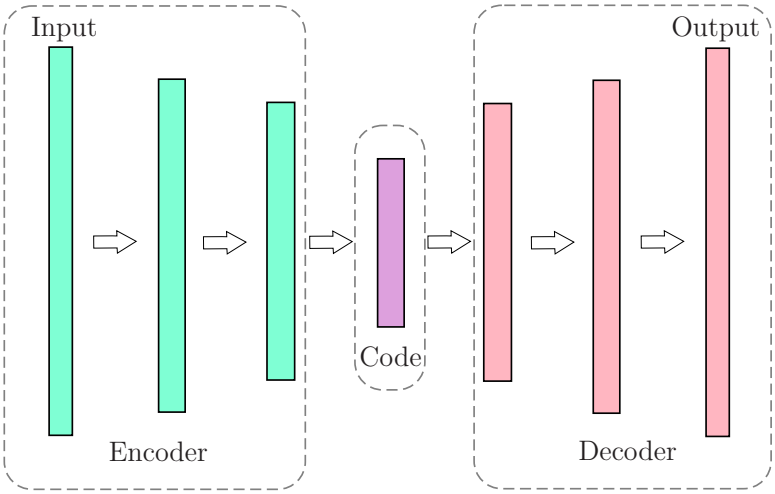


Figure 17. Autoencoder architecture consists of three components: encoder, code and decoder. The encoder maps the input to the code and then the decoder produces the output using this code. This autoencoder is undercomplete since the hidden code layer has a smaller dimension than the input layer.

3.8 U-net

U-net is a fully connected CNN that was introduced for biomedical image segmentation by Ronneberger et al. [51] in order to reduce the number of images required for the learning process since the number of training data is usually limited in biomedical cases. Image segmentation is closely related to classification tasks, where a single class label is output to an image. In segmentation problems, a class label is supposed to be assigned to each pixel of an image. In [51], U-net is shown to yield precise segmentations with very few training data.

As shown in Fig. 18, the network architecture has a 'U' shape and can be divided into two parts called contracting path (left half) and expansive path (right half). The contracting path can be considered as a typical CNN. It comprises the repeated application of 3×3 convolutions with valid padding (see Section 3.6), each accompanied by ReLU and max-pooling layers (see Section 3.6) for image size reduction called downsampling. The goal of the contracting path is to encode the input image into feature representations.

The expansive path consists of upsampling (increase in size) of the feature map implemented by 2×2 transposed convolutions (up-convolutions). After up-convolutions, the image is concatenated with the cropped feature map from the contracting path to incor-

porate information from the previous layers and thus improve the accuracy of prediction. Next, 3×3 convolutions (again accompanied by ReLUs) are applied to halve the number of features. The aim of expansive path is to project the features learnt by the contractive path onto the pixel space to obtain the output segmentation map.

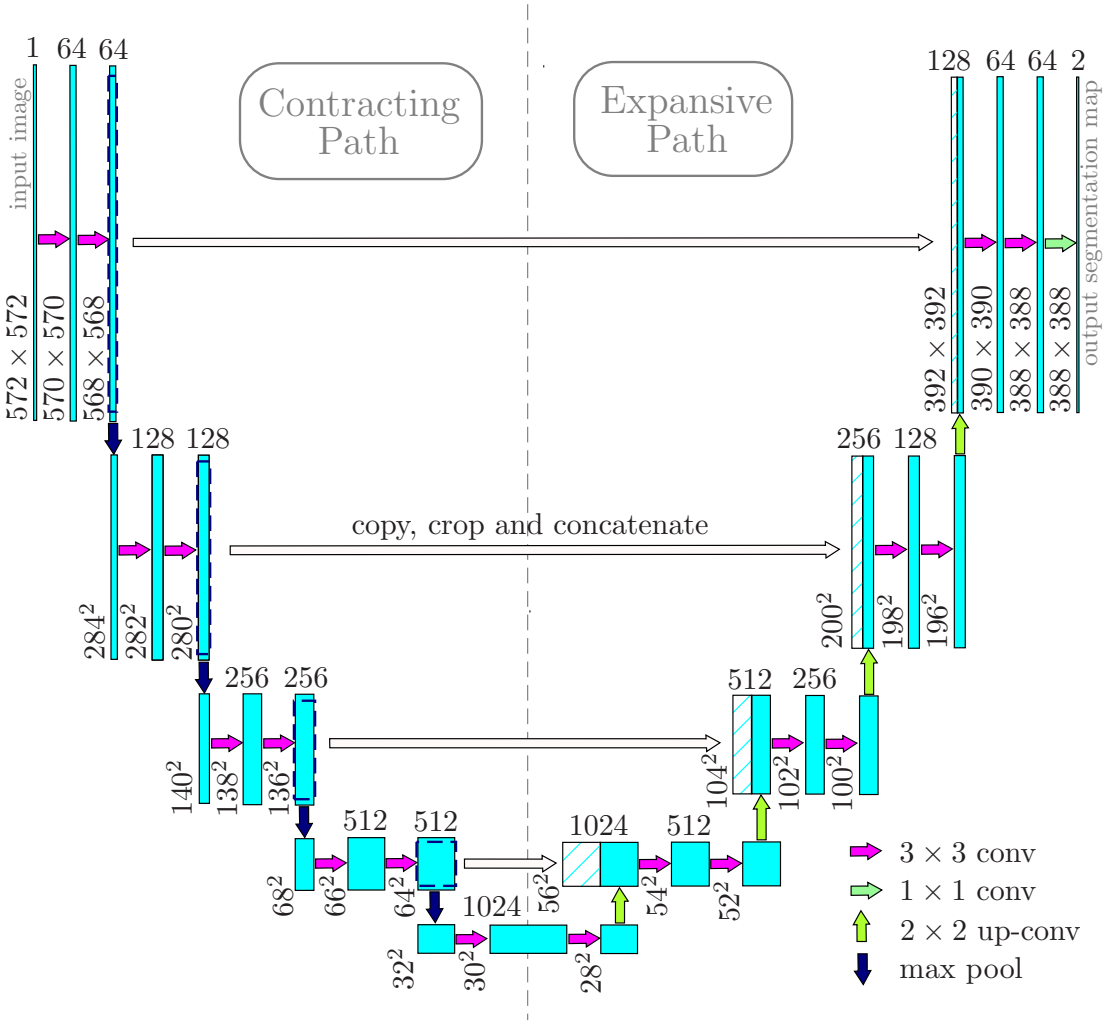


Figure 18. U-net architecture has a 'U' shape and consists of two halves: contracting path and expansive path. Rectangles represent the images, and numbers on the side correspond to their width and height (in pixels), while numbers placed above refer to the number of channels. Pink arrows pointing right represent convolutions with filters of size 3×3. Blue arrows pointing down refer to max-pooling that reduces the image size. Green arrows correspond to 2×2 transposed convolutions (up-convolutions) that expand the image dimensions. A pale green arrow in the expansive path denotes convolution with a filter of size 1×1. White arrows represent the process when the image from contracting path is copied, cropped, and then concatenated with the corresponding image from the expansive path.

3.9 AUTOMAP

Automated transform by manifold approximation (AUTOMAP) is introduced in [9] as an approach to learned image reconstruction which allows for a mapping between the detector domain (input sinogram) and the image domain (output reconstruction). The AUTOMAP implementation is based on a deep neural network consisting of three consecutive fully connected layers followed by a sparse autoencoder (see Section 3.7) composed of convolutional layers.

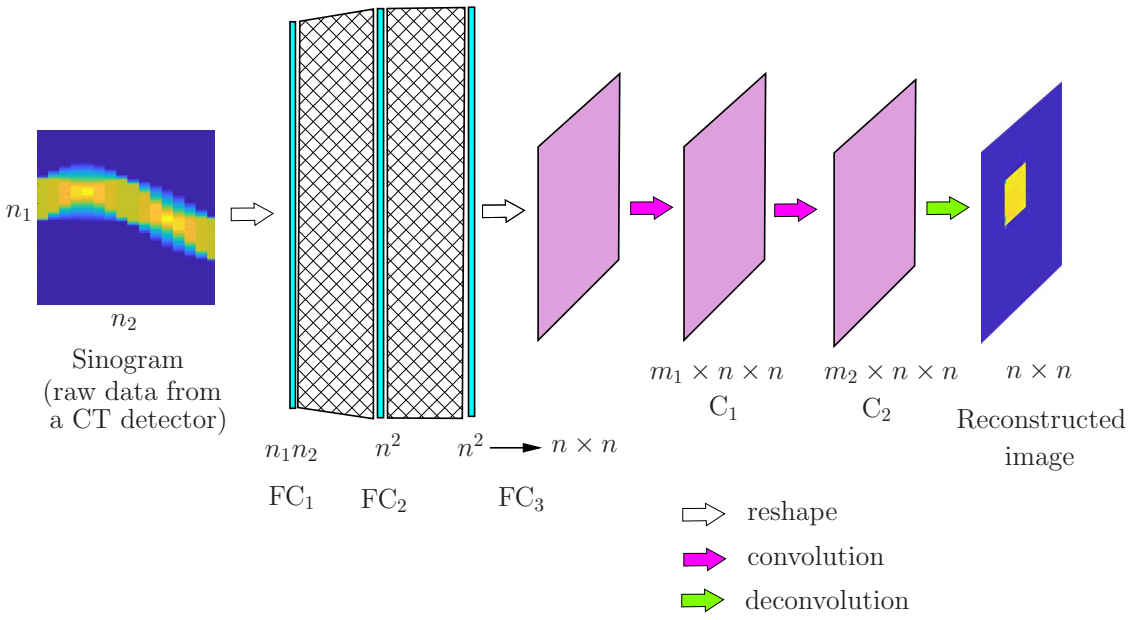


Figure 19. The AUTOMAP architecture. An input sinogram ($n_1 \times n_2$ matrix) is reshaped to the $n_1 n_2 \times 1$ input vector. The input layer FC_1 is fully connected to the hidden layer FC_2 of size $n^2 \times 1$ that is fully connected to another hidden layer FC_3. The $n^2 \times 1$ layer FC_3 is reshaped to the $n \times n$ matrix and followed by two successive convolutional layers C_1 and C_2 each convolving $m_1, m_2 = 64$ filters of 5×5 with stride 1. Finally, deconvolution with 64 filters of 7×7 with stride 1 is applied to the C_2 layer to produce the output reconstruction.

The AUTOMAP architecture used in this thesis is slightly different as compared to that presented in [9]. The modification must be done due to the different types of input data used. In [9], complex-valued CT detector data ($n \times n$ matrix) are employed as an input, they are separated into real and imaginary components and then concatenated in the $2n^2 \times 1$ real-valued input vector. In this thesis, input data ($n_1 \times n_2$ matrix) are real-valued, therefore they are directly reshaped to the $n_1 n_2 \times 1$ input vector. As shown in Fig. 19, the input layer FC_1 is fully connected to the hidden layer FC_2 of size $n^2 \times 1$ with by

the hyperbolic tangent activation function (see Section 3.3), the layer FC_2, in its turn, is fully connected to another hidden layer FC_3 that is also activated by the hyperbolic tangent function. The layer FC_3 is reshaped to the $n \times n$ matrix to prepare for upcoming convolution. The reshaped FC_3 layer is followed by the first convolutional layer C_1 that convolves 64 filters of 5×5 with stride 1 and utilizes ReLU as an activation function. The second convolution layer C_2 performs in the same way. Finally, deconvolution with 64 filters of 7×7 with stride 1 is applied to the C_2 layer that results in the output reconstructed image.

4 EXPERIMENTS

4.1 Experimental setup

The experiments, being discussed in this section, can be divided into two independent parts corresponding to two different approaches in image reconstruction from the raw data collected by a CT detector:

- the two-step approach when a CNN is used to post-process FBP-reconstructions;
- the one-step approach when a reconstruction process is fully automated with a DNN.

The aim of the experiments is to test the generalization capability of both techniques. The general idea of the two-step approach is shown in Fig. 20. In the first step, the filtered backprojection is applied to a sinogram (raw data received from a CT detector) to obtain the FBP-reconstruction. In the second step, the convolutional neural network processes this reconstructed image to improve its quality by eliminating star artefacts and other defects resulting from the limited number of available data. In this approach, the U-net CNN architecture is used (see Section 3.8).

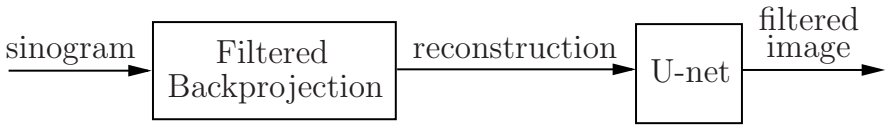


Figure 20. The general idea of a two-step approach in CT image reconstruction. In the first step, the filtered backprojection is applied to a sinogram resulting in reconstruction. In the second step, U-net operates on this reconstruction to reduce the number of artefacts.

In the one-step approach, AUTOMAP deep neural network (see Section 3.9) is used to learn the full reconstruction process, as shown in Fig. 21. That is, the network approximates the projection filtering, backprojection operator and image post-processing at the same time. The input of AUTOMAP DNN is a sinogram, and the output is a backprojected and filtered image.

Figure 21. The general idea of the one-step approach in CT image reconstruction. The technique utilizes DNN to learn the whole reconstruction process, mapping directly raw measured data (sinogram) to the filtered image.

For evaluating the performance of the two discussed approaches synthetic data consisting of various digital phantoms were used (Fig. 22). These phantoms constituted different patterns of fixed size and random location (or combinations of the patterns), depending on the purpose of the experiment conducted. A disk was chosen as the main pattern for network training. In addition to disk patterns, larger disks, ellipses, patterns with right angles (squares of two sizes and diamonds), and stripes (as qualitatively different patterns) were used in the training sets in some experiments. For the purpose of testing all the mentioned patterns were involved.

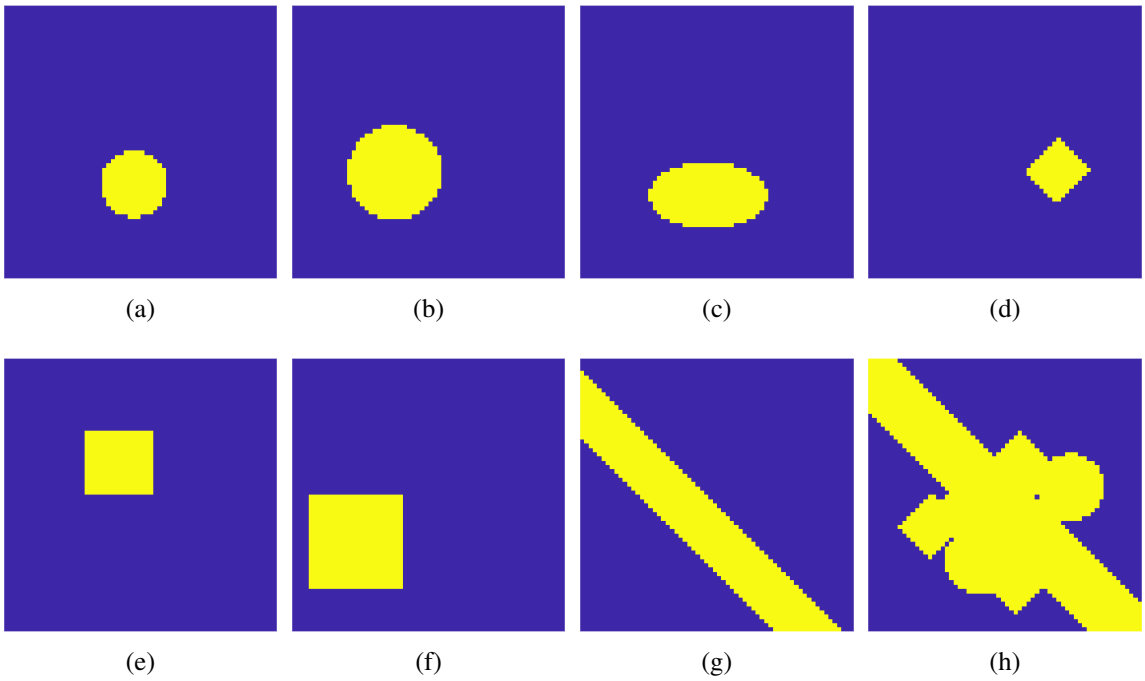


Figure 22. Examples of synthetic data (digital phantoms): (a) disk, (b) larger disk, (c) ellipse, (d) diamond, (e) square, (f) larger square, (g) stripe, (h) combination of disk, diamond, and stripe patterns.

In the case of the one-way approach, the synthetic input data (sinograms) were generated by applying the Radon transform to the digital phantoms with a specified number of angles at which projections were computed according to the parallel scanning geometry (see Section 2.8). With regard to the two-step approach, the synthetic input data for U-net constituted the filtered backprojections that were obtained using inverse Radon transform with Ram-Lak filter applied to sinograms (see Section 2.7).

The size of the testing dataset was equal to 128 elements in all the experiments, while the size of the training dataset varied for different experiments. The training and testing sets were generated in MATLAB, next stored in HDF5 files (Hierarchical Data Format 5 file type) and accessed from Python by using h5py package. Network architectures were implemented in TensorFlow [52]. For ANN training and evaluation a computer with 8GB RAM and NVIDIA GeForce GTX 980 SLI was used. It should be noted that in the last experiment (Section 4.3.4), NVIDIA DGX-1 with four Tesla V100 GPUs was used, because in this case the training set included a big variety of patterns, and therefore the AUTOMAP network required much more time to be trained.

In the implementation of both approaches, the loss function was defined as the Frobenius norm [53] of the difference between the ground truth image y and reconstruction $\hat{y}$, i.e.

$$\|y - \hat{y}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |y_{i,j} - \hat{y}_{i,j}|^2}.$$

In two-step approach with U-net used for post-processing, the loss function was minimized through Adam Optimizer with learning rate $\eta = 0.0001$ (see Section 3.5), while in one-step approach RMSprop Optimizer (identical to Adadelta Optimizer discussed in Section 3.5) with learning rate $\eta = 0.00002$ and decay factor $\rho = 0.9$ was used.

For U-net training the batch size equal to 8 elements was used. In the case of AUTOMAP trained on the dataset of 512 elements, the batch size used was also 8 elements, but in the case of the training set containing 50000 elements, the batch size was increased up to 100 elements. The training time and the number of epochs depended on the network architecture as well as amount and diversity of data in the training set. The network was trained until a constant minimum training error was achieved, in overall it took

- about 15 minutes to train U-net on 256, 512, and 1024 single disk phantoms, numbers of epochs were 1563, 782, and 391, respectively (Section 4.2);
- approximately 4 hours to train AUTOMAP on 512 single disk phantoms, number

of epochs was 782 (Section 4.3.1);

- about 8 hours to train AUTOMAP on 50000 multiple disk phantoms, number of epochs was 200 (Section 4.3.2);
- approximately 12 hours to train AUTOMAP on 50000 phantoms with multiple disks, diamonds and stripes, number of epochs was 400 (Section 4.3.3);
- about 36 hours to train AUTOMAP on 50000 phantoms with multiple disks, diamonds, stripes and squares (the training was done using NVIDIA DGX-1 with four Tesla V100 GPUs), number of epochs was 500 (Section 4.3.4).

4.2 Reconstructed image post-processing with U-net

In the experiments discussed in this section, U-net was applied to FBP-reconstructions in the post-processing stage. The conducted experiments were meant

- to find an optimal size of the training set (with single disk phantoms) that delivers the best generalization capability of U-net (Section 4.2.1),
- to examine how different number of projections in the training set (with single disk phantoms) affects the reconstruction quality (Section 4.2.2),
- to test the effect of multiple disk patterns in the training set on the generalization capability of the network (Section 4.2.3).

4.2.1 Determining the optimal size of the training set

To find the optimal size of the training set, U-net was trained using 3 different datasets, consisting of 256, 512 and 1024 elements (Fig. 23). An element of the training set was a pair of images. The first image in the pair corresponded to a disk phantom (ground truth image). The second image was a filtered backprojection (input data). In all the experiments discussed in this section, projections were computed at 16 different angles evenly spaced between 0 and 180 degrees (see Fig. 4 for details).

Three sets of experiments described below corresponded to three different sizes of training datasets. Each experimental case checked the capability of U-net for generalization by testing it on different phantoms, not included into the training set.

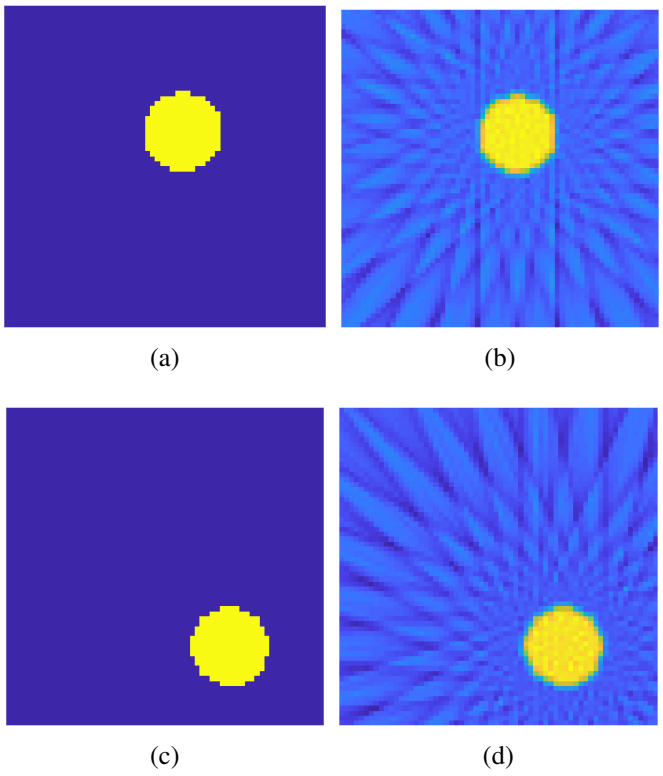
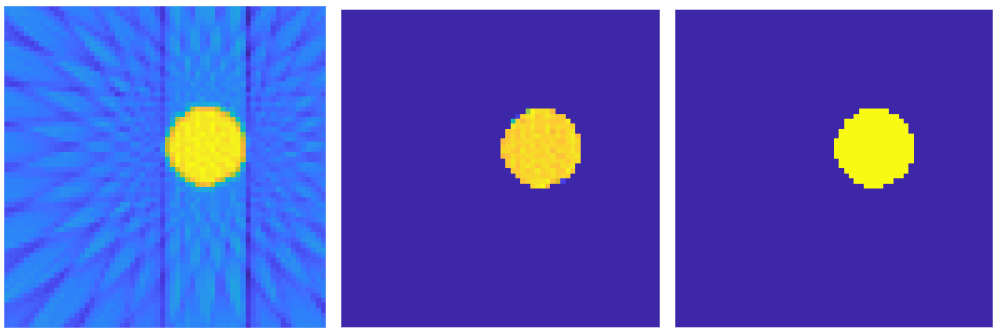


Figure 23. Digital disk phantoms from the training set (a),(c) and corresponding filtered backprojections (b), (d).

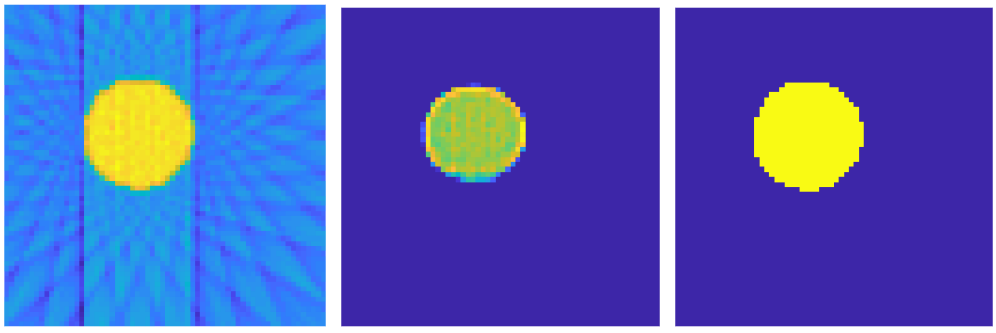
Training set of 256 elements.

This set of experiments corresponded to the smallest number of training data (256 elements). As shown in Fig. 24, when tested on the disk patterns of the same size as those in the training set, U-net performed successfully. However, with increase in the size of the disks in the testing set, the accuracy of U-net decreased, namely, it tended to slightly reduce the radiuses of disk patterns and add some noise into their inner parts (Fig. 25). Likewise, the performance of U-net suffered when it was tested on ellipse patterns: ellipse interior was noisy in output reconstructions (Fig. 26).



(a) (b) (c)

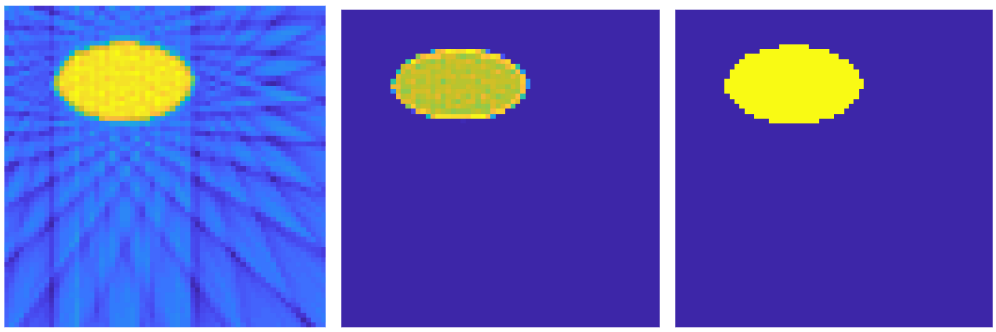
Figure 24. FBP post-processing with U-net (training set: 256 disk phantoms, testing set: 128 disk phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (disk phantom).



(a) (b) (c)

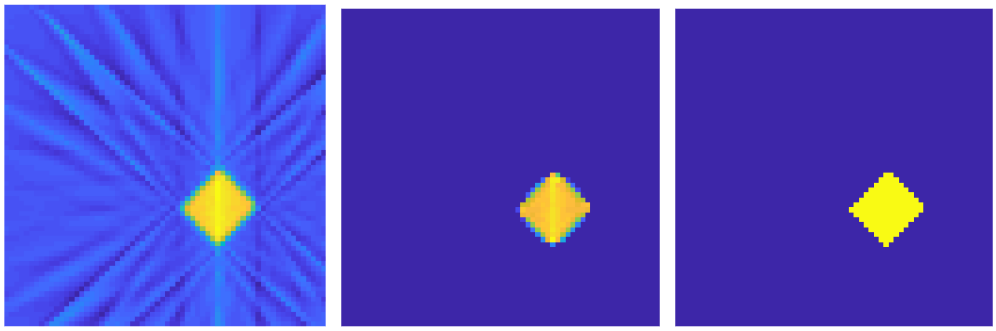
Figure 25. FBP post-processing with U-net (training set: 256 disk phantoms, testing set: 128 phantoms containing larger disks): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (disk phantom). The disk pattern in the output image is noisy and smaller than that in the ground truth image.

Next, the testing sets containing images of squares and diamonds were considered. U-net performed well enough when tested on diamonds (Fig. 27) and squares of about the same size as disks in the training set (Fig. 28): in the output images, the edges of these patterns were relatively sharp. However, when the testing set contained squares of slightly increased sizes, the generalization capability of U-net failed, since the bottom parts of the squares were significantly corrupted, as shown in Fig. 29. It can be noticed that the closer to the boundary a square was situated, the more corrupted it was in the output reconstruction.



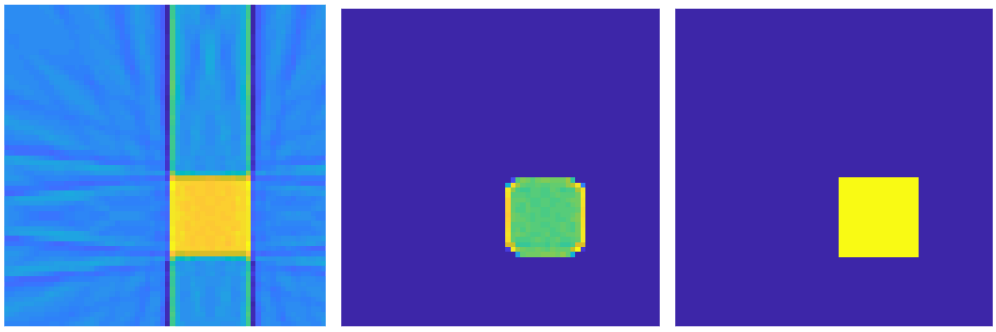
(a) (b) (c)

Figure 26. FBP post-processing with U-net (training set: 256 disk phantoms, testing set: 128 ellipse phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (ellipse phantom). The ellipse in the output reconstruction is noisy.



(a) (b) (c)

Figure 27. FBP post-processing with U-net (training set: 256 disk phantoms, testing set: 128 diamond phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (diamond phantom).



(a) (b) (c)

Figure 28. FBP post-processing with U-net (training set: 256 disk phantoms, testing set: 128 square phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (square phantom).

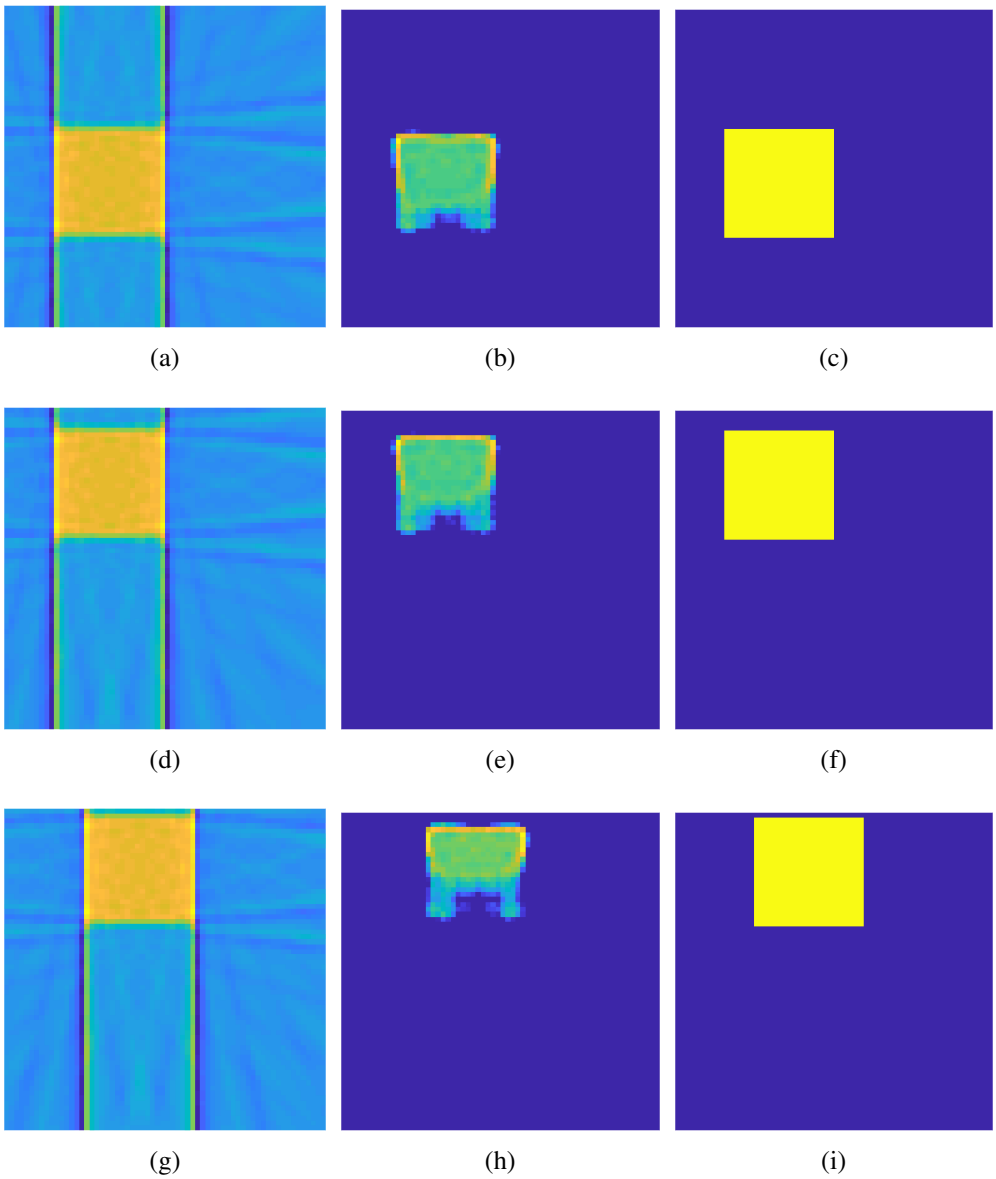
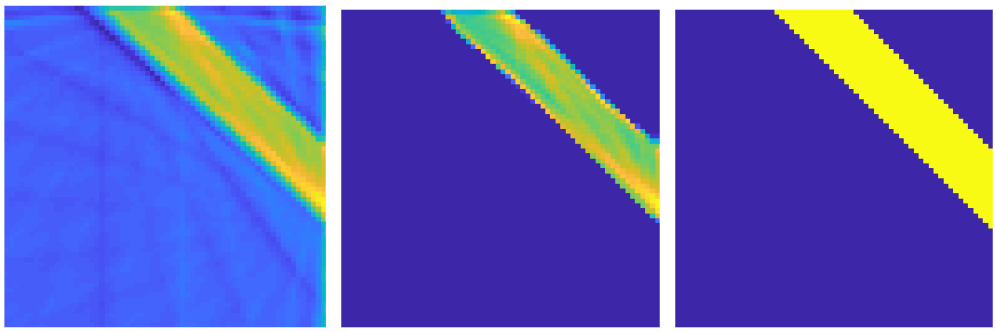


Figure 29. FBP post-processing with U-net (training set: 256 disk phantoms, testing set: 128 phantoms containing larger squares): (a), (d), (g) filtered backprojections (network input); (b), (e), (h) images after post-processing (network output); (c), (f), (i) ground truth data (square phantom). The closer to the image boundary a square is located, the more it is corrupted in the output images.

Turning to the case where the images in the testing set contained stripe patterns, it can be noticed that U-net performed successfully regardless the position of a stripe in the image (Fig. 30).



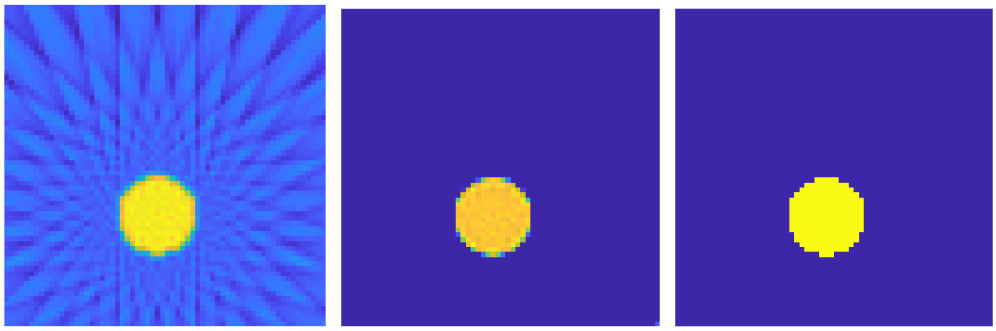
(a) (b) (c)

Figure 30. FBP post-processing with U-net (training set: 256 disk phantoms, testing set: 128 stripe phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (stripe phantom).

Summary. The results indicated a high generalization capability of U-net trained on 256 disk phantoms when it was tested on patterns of different shapes, but the same sizes as the disk patterns in the training set. However, the accuracy of the network decreased significantly when the testing sets contained larger patterns.

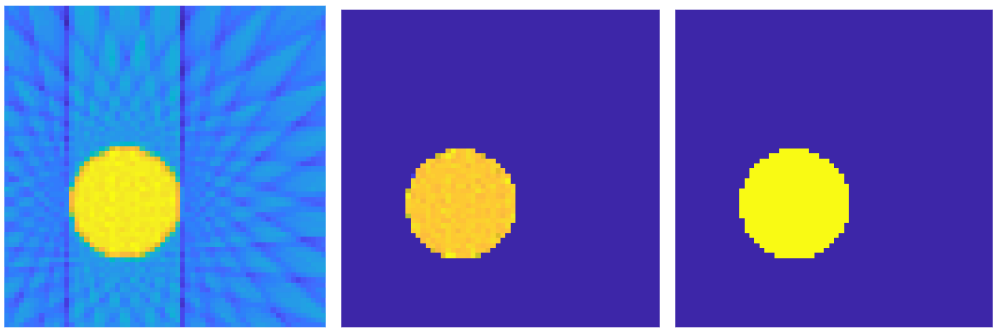
Training set of 512 elements.

In this set of experiments, the number of data in the training set was increased up to 512. U-net accurately processed disk patterns of the same size as those in the training set (Fig. 31) as well as larger disks (Fig. 32) and ellipses (Fig. 33).



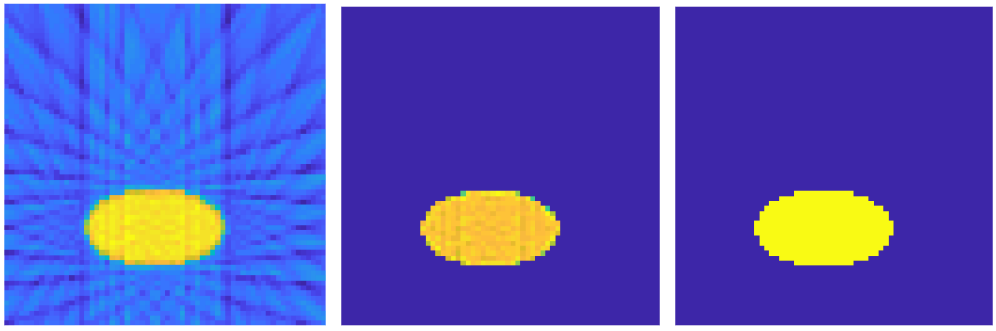
(a) (b) (c)

Figure 31. FBP post-processing with U-net (training set: 512 disk phantoms, testing set: 128 disk phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (disk phantom). Output image is good enough as compared to the ground truth image.



(a) (b) (c)

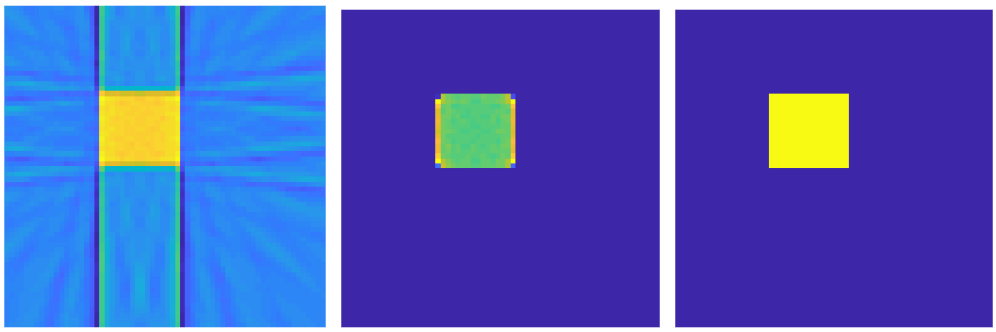
Figure 32. FBP post-processing with U-net (training set: 512 disk phantoms, testing set: 128 phantoms containing larger disks): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (disk phantom).



(a) (b) (c)

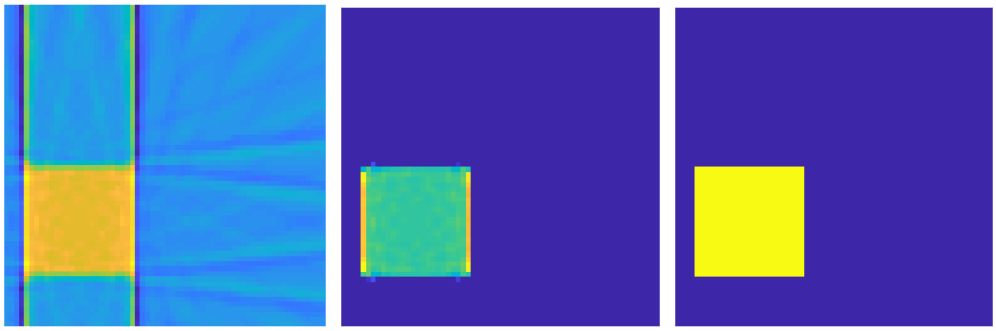
Figure 33. FBP post-processing with U-net (training set: 512 disk phantoms, testing set: 128 ellipse phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (ellipse phantom).

With regard to the testing set containing square patterns of about the same size as disk patterns in the training set (Fig. 34) and the testing set with slightly increased square patters (Fig. 35), the following results were obtained. In both cases U-net performed adequately, processing square patterns as successfully as disk and ellipse patterns. In addition, no significant difference in accuracy of the network was found while testing on the diamond patterns (Fig. 36).



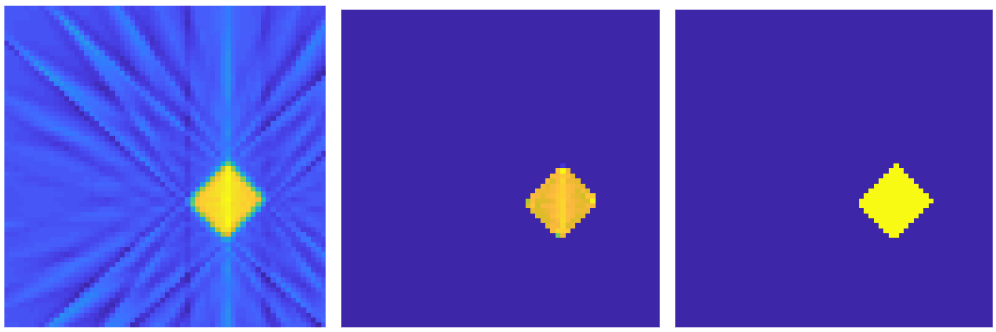
(a) (b) (c)

Figure 34. FBP post-processing with U-net (training set: 512 disk phantoms, testing set: 128 square phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (square phantom).



(a) (b) (c)

Figure 35. FBP post-processing with U-net (training set: 512 disk phantoms, testing set: 128 phantoms containing larger squares): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (square phantom).



(a) (b) (c)

Figure 36. FBP post-processing with U-net (training set: 512 disk phantoms, testing set: 128 diamond phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (diamond phantom).

Another result to emerge from the experiments was that the generalization capability of U-net significantly decreased when the testing set contained stripe patterns (Fig. 37). It can be noticed that when the stripe coincided with the main diagonal, the output image contained no artefacts (Fig. 37(e)). But when the stripe was displaced from the main diagonal, the output reconstruction contained artefacts near the edges (Fig. 37(b), 37(h)).

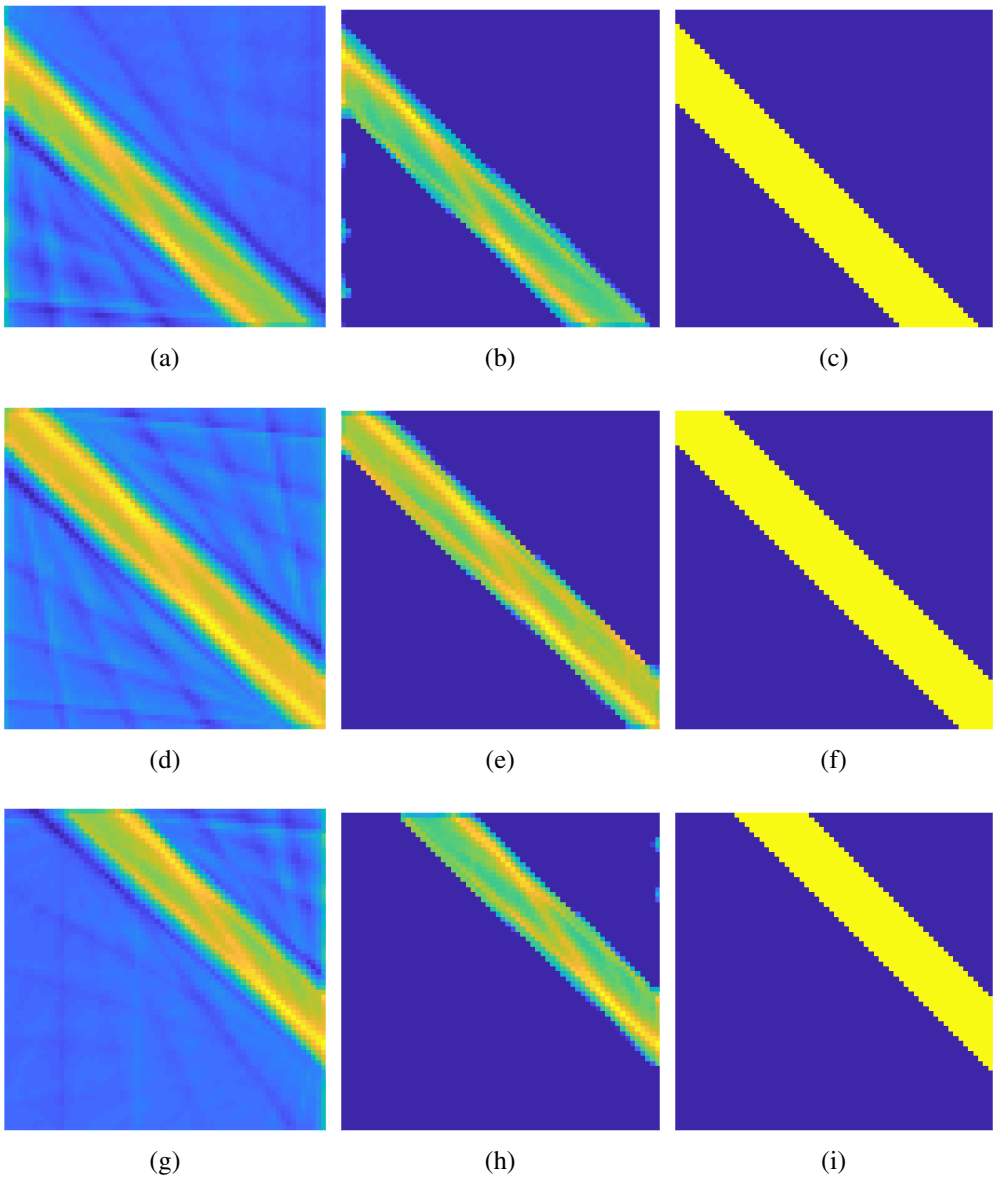


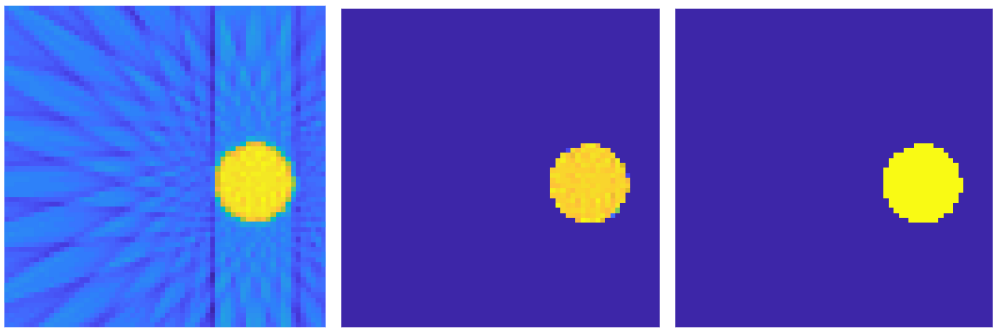
Figure 37. FBP post-processing with U-net (training set: 512 disk phantoms, testing set: 128 stripe phantoms): (a), (d), (g) filtered backprojections (network input); (b), (e), (h) images after post-processing (network output); (c), (f), (i) ground truth data (stripe phantom). Some artefacts appear in the output images if the stripe is displaced from the main diagonal, e.g. (b), (h).

Summary. The results showed that U-net trained on 512 disk phantoms performed well enough when tested on disks and squares (of two different sizes), ellipses, and diamonds. However, poor performance was observed when the training set contained stripes.

Training set of 1024 elements.

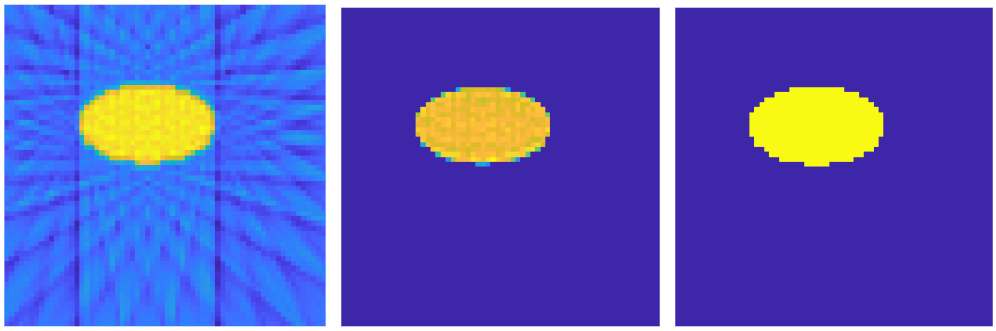
This set of experiments corresponded to the highest amount of training data (1024 ele-

ments). When tested on the disk patterns of the same size as those in the training set (Fig. 38) and on ellipse patterns (Fig. 39), U-net performed adequately. However, while testing on larger disks, some noise in the center of disks appeared (Fig. 40).



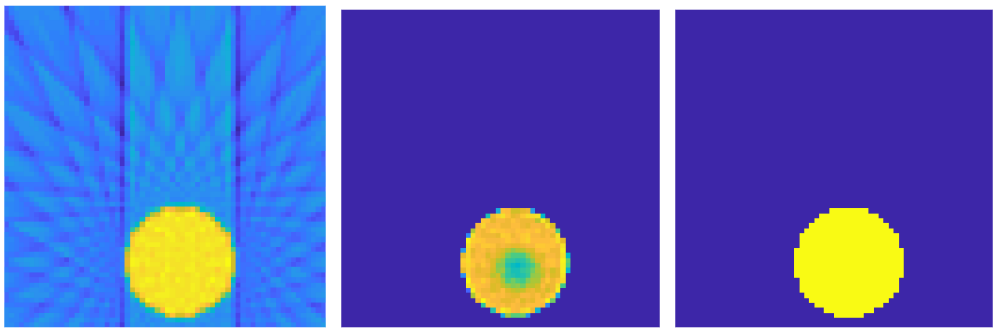
(a) (b) (c)

Figure 38. FBP post-processing with U-net (training set: 1024 disk phantoms, testing set: 128 disk phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (disk phantom).



(a) (b) (c)

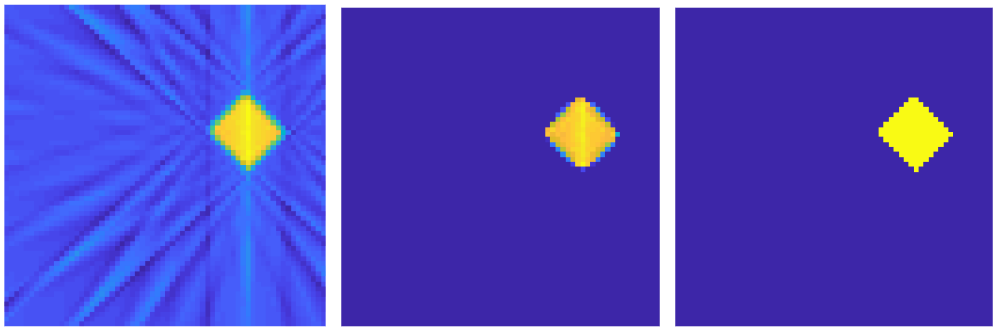
Figure 39. FBP post-processing with U-net (training set: 1024 disk phantoms, testing set: 128 ellipse phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (ellipse phantom).



(a) (b) (c)

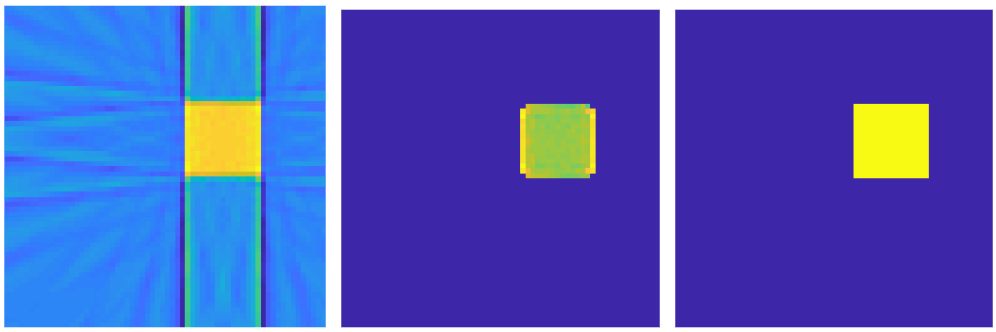
Figure 40. FBP post-processing with U-net (training set: 1024 disk phantoms, testing set: 128 phantoms containing larger disks): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (disk phantom). The disk in the output image is noisy in the central part, but has the same size as that in the ground truth image.

With regard to the testing sets containing images of squares and diamonds, U-net performed adequately when tested on diamond (Fig. 41) and square patterns of about the same size as patterns in the training set (Fig. 42), delivering relatively sharp pattern edges. However, when the testing set contained larger square patterns, the generalization capability of U-net decreased, since it tended to slightly 'erase' the central parts of the squares, as shown in Fig. 43.



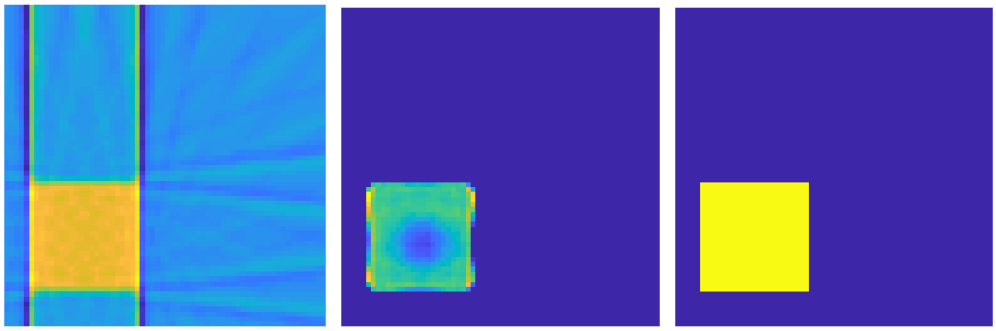
(a) (b) (c)

Figure 41. FBP post-processing with U-net (training set: 1024 disk phantoms, testing set: 128 diamond phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (diamond phantom).



(a) (b) (c)

Figure 42. FBP post-processing with U-net (training set: 1024 disk phantoms, testing set: 128 square phantoms): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (square phantom).



(a) (b) (c)

Figure 43. FBP post-processing with U-net (training set: 1024 disk phantoms, testing set: 128 phantoms containing larger squares): (a) filtered backprojection (network input), (b) image after post-processing (network output), (c) ground truth data (square phantom). The central part of the square in the output image is slightly 'erased'.

In the case of the testing set containing stripe patterns, the generalization capability of U-net decreased and depended on the position of a stripe (Fig. 44). It can be noted that when the stripes were displaced from the main diagonal, the output reconstruction contained artefacts near the edges (Fig. 44(b), 44(h)).

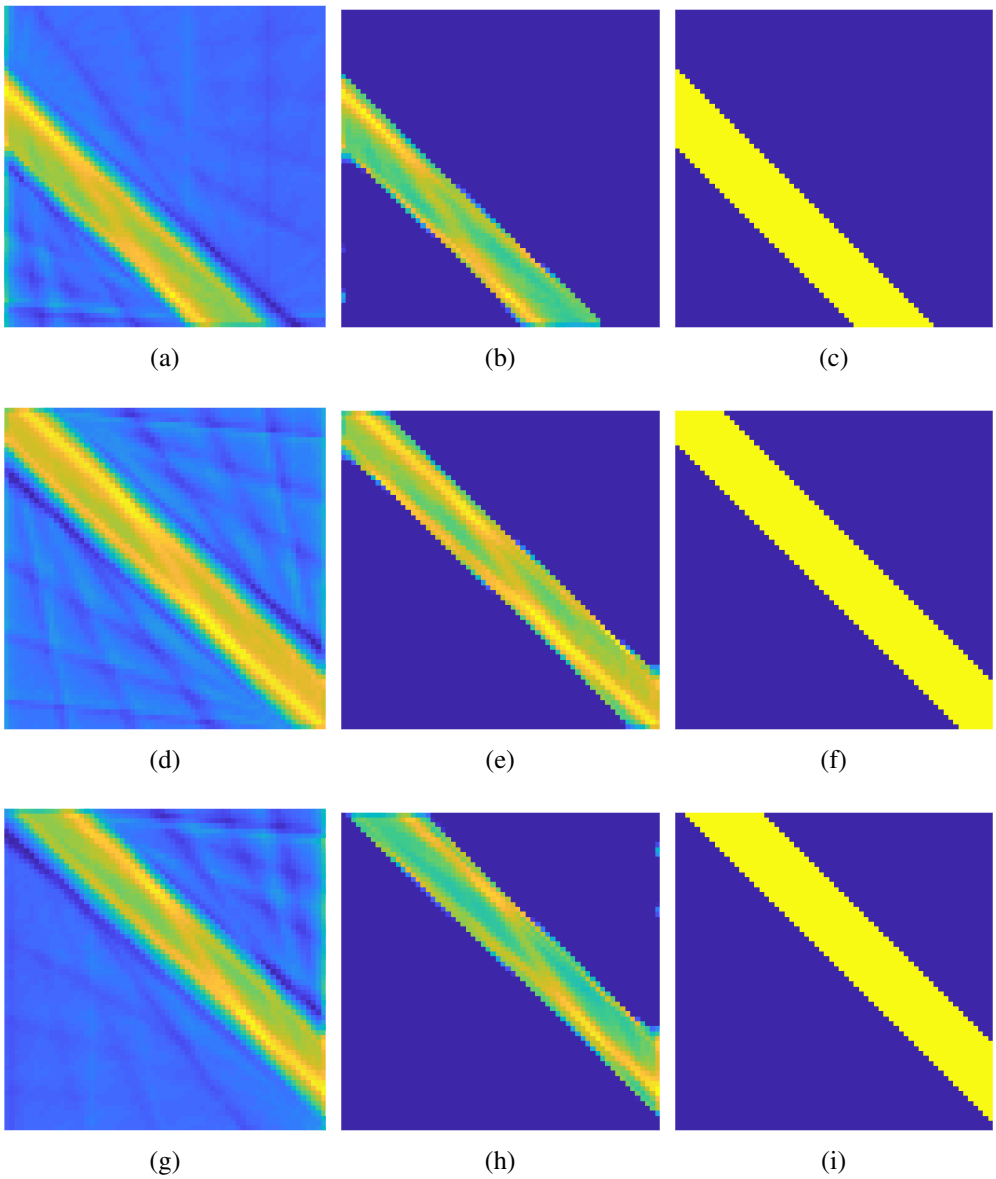
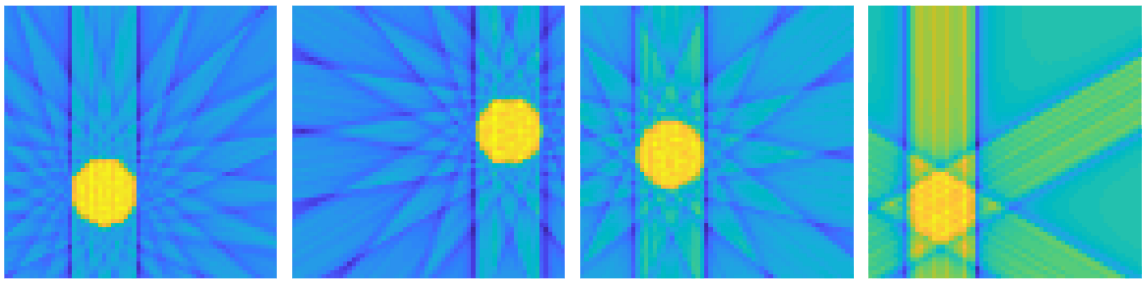


Figure 44. FBP post-processing with U-net (training set: 1024 disk phantoms, testing set: 128 stripe phantoms): (a), (d), (g) filtered backprojections (network input); (b), (e), (h) images after post-processing (network output); (c), (f), (i) ground truth data (stripe phantom). Some artefacts appear near the edges in the output images if the stripe is displaced from the main diagonal, e.g. (b), (h).

Summary. The results showed that U-net trained on 1024 disk phantoms accurately processed diamond, ellipse, disk and square patterns of about the same size as those in the training set. However, the generalization capability of the network decreased when it was applied to larger disk and square patterns as well as stripe patterns.

4.2.2 Studying the influence of the projection number on the generalization capability of U-net

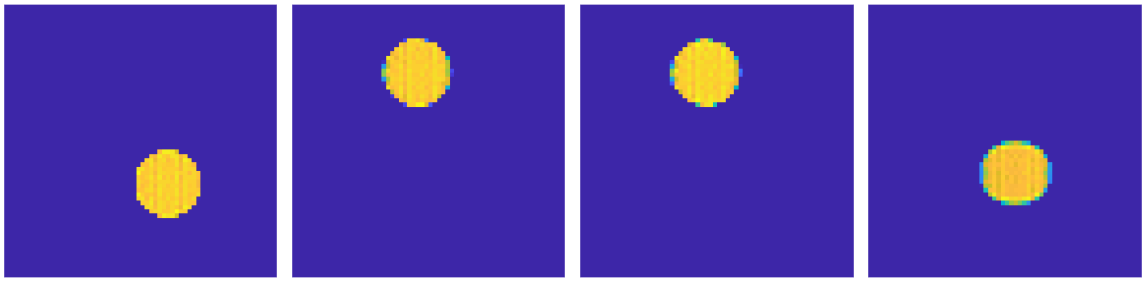
The experiments described in this section were aimed to test the effect of the number of projections on the ability of U-net to generalize. In a manner similar to that described in Section 4.2.1, U-net was trained using 512 digital disk phantoms and corresponding filtered backprojections, while the number of projections was varied (see Fig. 45). In total four training sets were used, they corresponded to 12, 10, 8 and 4 angles at which the projections were computed (see Fig. 4 for a detailed description of projections and the scanning geometry).



(a) (b) (c) (d)

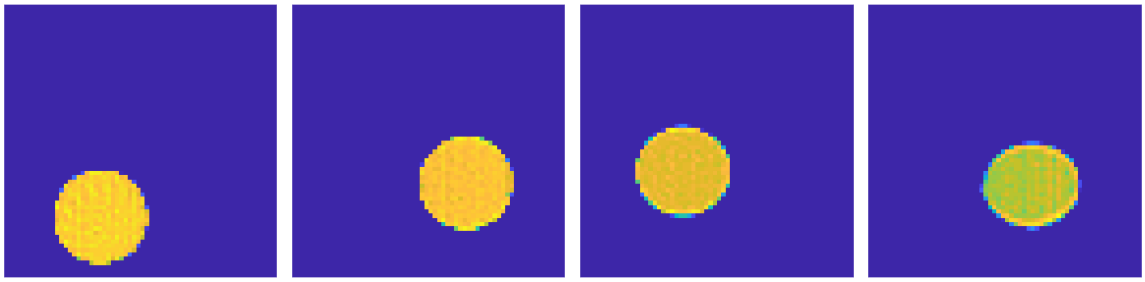
Figure 45. Filtered backprojections in the different training sets. The number of angles at which projections are computed: (a) 12, (b) 10, (c) 8, (d) 4.

After training on each of the four datasets, U-net was tested on the same patterns as in Section 4.2.1 (the projections were computed at 16 different angles evenly spaced between $0°$ and $180°$). As shown in Fig. 46, in the case of the testing set containing disk patterns of the same size as those in the training set, more artefacts appeared on the pattern boundaries as the number of projections decreased. The similar results were observed when U-net was tested on ellipses (Fig. 48). With regard to the testing set containing larger disk patterns, less number of projections caused the significant alteration in the shape of the patterns, as shown in Fig. 47.



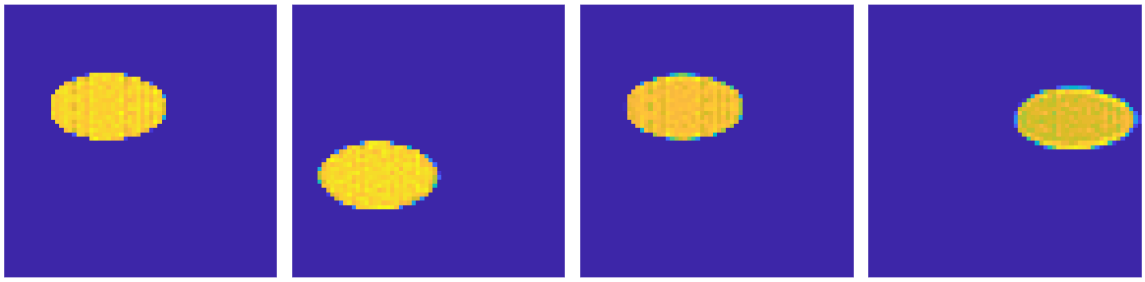
(a) (b) (c) (d)

Figure 46. FBP post-processing with U-net (training set: 512 disk phantoms with the varied number of projection angles, testing set: 128 disk phantoms). Filtered reconstructions (network output), the number of angles at which projections are computed in the training set: (a) 12, (b) 10, (c) 8, (d) 4. The fewer projections are used in the training set, the more noise appears in the disk boundaries in the reconstructions.



(a) (b) (c) (d)

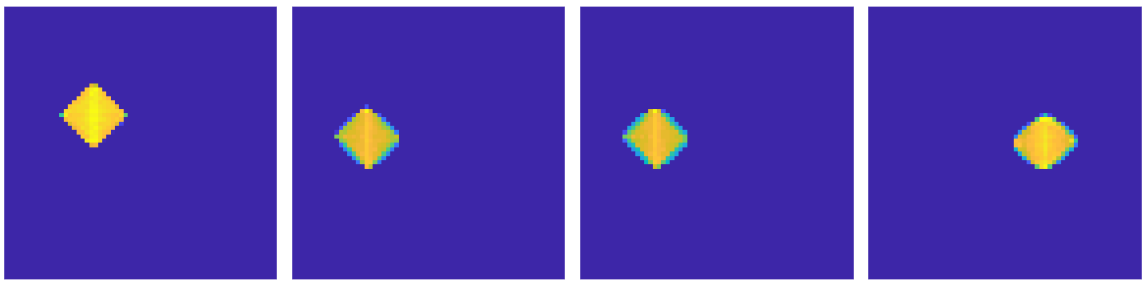
Figure 47. FBP post-processing with U-net (training set: 512 disk phantoms with the varied number of projection angles, testing set: 128 phantoms containing larger disks). Filtered reconstructions (network output), the number of projections at which projections are computed in the training set: (a) 12, (b) 10, (c) 8, (d) 4. A lower number of projections in the training set causes disks to shrink from the top down.



(a) (b) (c) (d)

Figure 48. FBP post-processing with U-net (training set: 512 disk phantoms with the varied number of projection angles, testing set: 128 ellipse phantoms). Filtered reconstructions (network output), the number of angles at which projections are computed in the training set: (a) 12, (b) 10, (c) 8, (d) 4. The fewer projections are used in the training set, the more noise on the boundary of ellipses is observed.

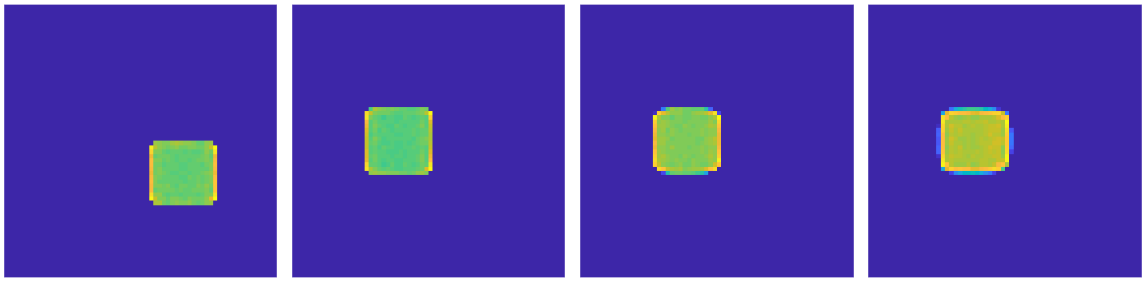
In the case of diamond patterns in the testing set, the ability of U-net to generalize deteriorated: the output reconstructions contained inscribed quadrilaterals rather than diamonds as the number of projections decreased (Fig. 49).



(a) (b) (c) (d)

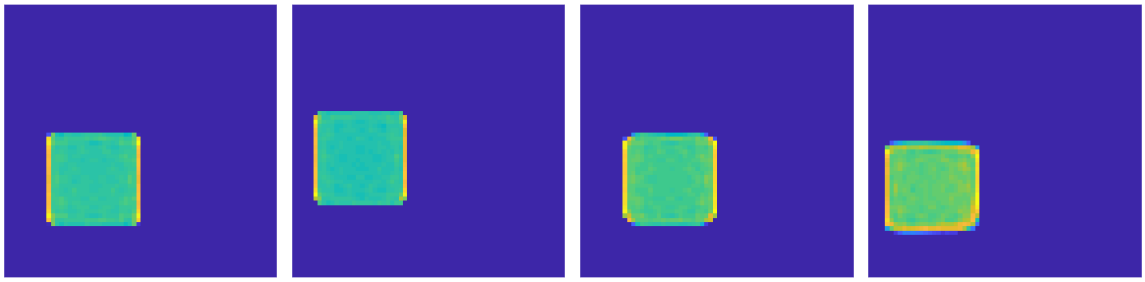
Figure 49. FBP post-processing with U-net (training set: 512 disk phantoms with the varied number of projection angles, testing set: 128 diamond phantoms). Filtered reconstructions (network output), the number of angles at which projections are computed in the training set: (a) 12, (b) 10, (c) 8, (d) 4. U-net tends to inscribe the diamonds into the circles as the number of projections in the training set decreases.

Turning to the case of the testing set containing square patterns, it can be noticed that the less number of projections was used, the more U-net rounded the corners of squares (Fig. 50, 51). However, as shown in Fig. 52, while testing on stripes U-net performed equally good regardless of the number of projections in the training set.



(a) (b) (c) (d)

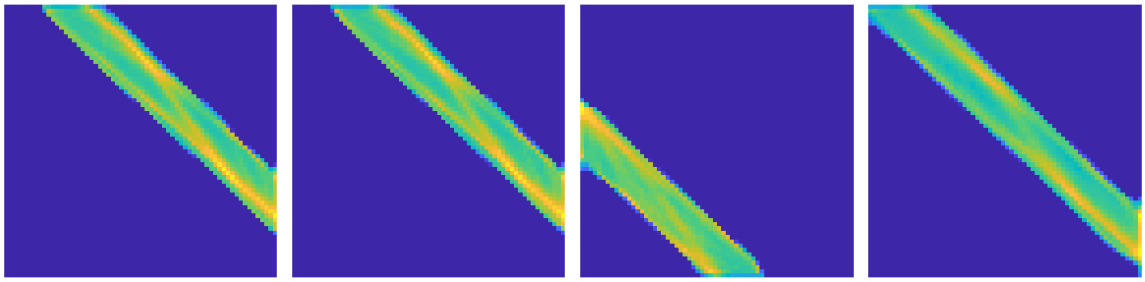
Figure 50. FBP post-processing with U-net (training set: 512 disk phantoms with the varied number of projection angles, testing set: 128 square phantoms). Filtered reconstructions (network output), the number of angles at which projections are computed in the training set: (a) 12, (b) 10, (c) 8, (d) 4. The decrease in the number of projections in the training set causes U-net to round the corners of squares.



(a) (b) (c) (d)

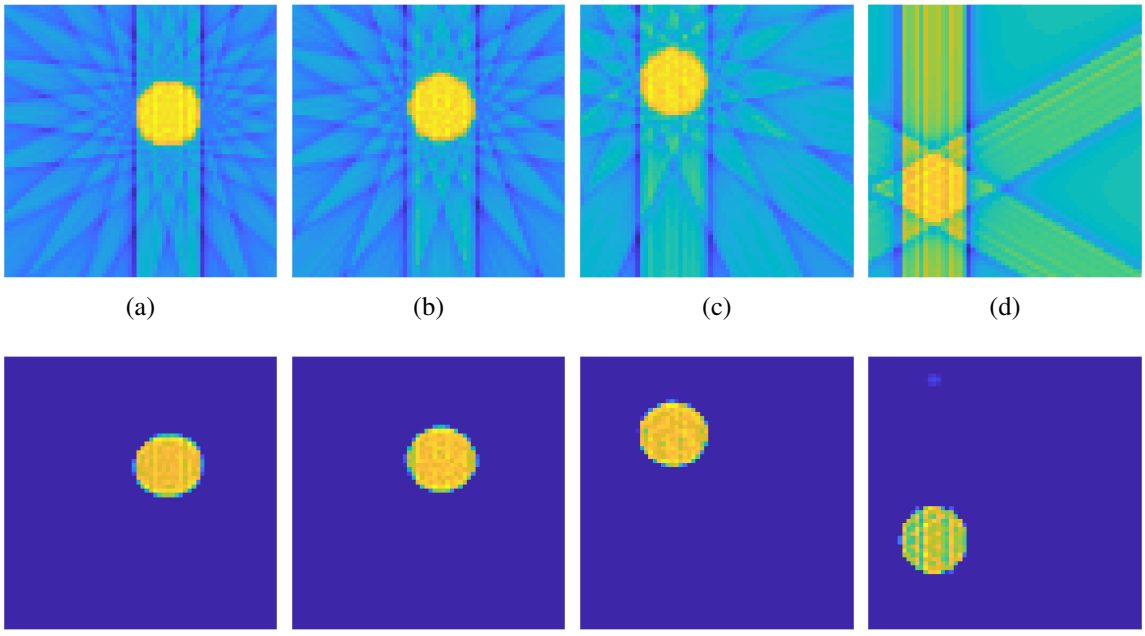
Figure 51. FBP post-processing with U-net (training set: 512 disk phantoms with the varied number of projection angles, testing set: 128 phantoms containing larger squares). Filtered reconstructions (network output), the number of angles at which projections are computed in the training set: (a) 12, (b) 10, (c) 8, (d) 4. The decrease in the number of projections in the training set causes U-net to round the corners of larger squares.

Additionally, U-net was trained on 512 disk phantoms with projections computed at 4 angles evenly spaced between $0°$ and $180°$ (Fig. 45(d)), and then tested on disk phantoms with the various numbers of projection angles (12, 10, 8 and 4 angles). As shown in Fig. 53, the less number of projections the testing set contained, the worse the quality of the reconstruction was.



(a) (b) (c) (d)

Figure 52. FBP post-processing with U-net (training set: 512 disk phantoms with the varied number of projection angles, testing set: 128 stripe phantoms). Filtered reconstructions (network output), the number of angles at which projections are computed in the training set: (a) 12, (b) 10, (c) 8, (d) 4. No significant discrepancy is observed when the number of projections in the training set declines.



(a) (b) (c) (d)

(e) (f) (g) (h)

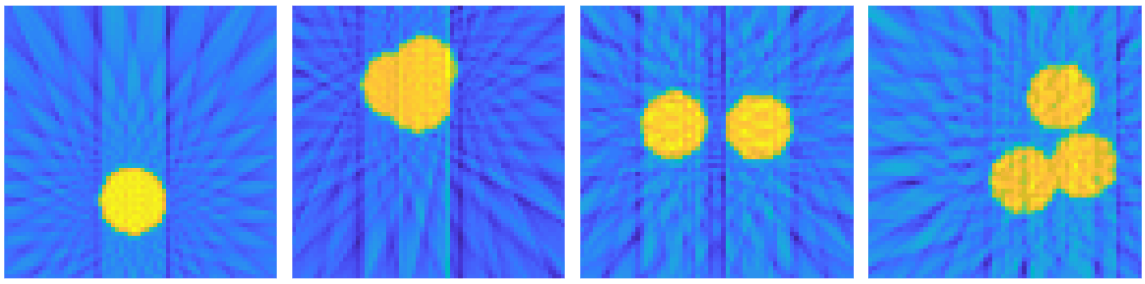
Figure 53. FBP post-processing with U-net (training set: 512 disk phantoms with 4 projection angles, testing set: 128 disk phantoms with the varied number of projection angles): filtered backprojections (network input), number of angles at which projections are computed in the testing set is (a) 12, (b) 10, (c) 8, (d) 4; corresponding images after post-processing (network output), the number of angles at which projections are computed in the testing set is (e) 12, (f) 10, (g) 8, (h) 4.

Summary. The experimental results demonstrated that a small number of projections in the training set resulted in the deteriorating accuracy of the reconstructions for all the patterns except for the stripes (stripe patterns were well-processed regardless of the projection number in the training set). In the case of the training set containing a small fixed number

of projections (projections were computed at 4 angles evenly spaced between $0°$ and $180°$) the results are the same: the fewer projections were used, the worse reconstruction quality was observed.

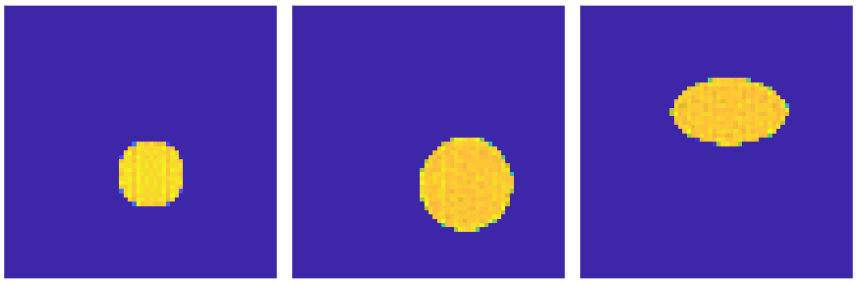
4.2.3 Testing the effect of multiple disk patterns on the generalization capability of U-net

This section provides the experiments, where the training set of 512 images contained phantoms with 1 to 3 disks which were allowed to intersect and form more sophisticated patterns (Fig. 54). As shown in Fig. 55 and Fig. 56, in the case of the testing sets containing disks and squares of different sizes, ellipses, diamonds and stripes, U-net performed equally well as compared to the experiments with single disk patterns in the training set (Section 4.2.1).



(a) (b) (c) (d)

Figure 54. Examples of FBFs in the training set. Some of the possible options: a single disk (a), three intersecting disks (b), two disjoint disks (c), one disjoint and two intersecting disks (d).



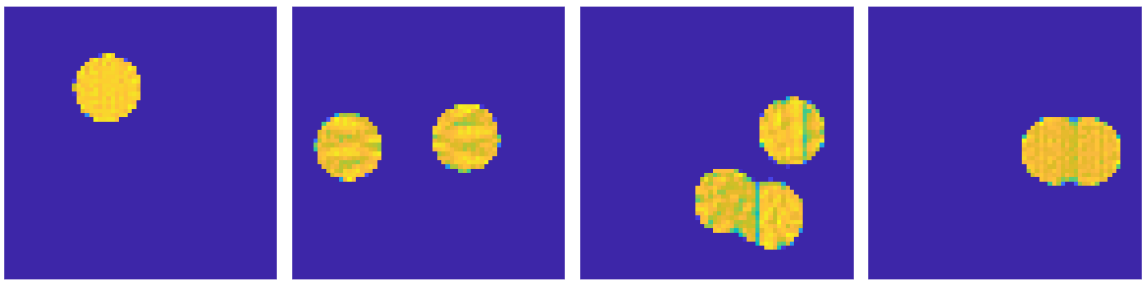
(a) (b) (c)

Figure 55. FBP post-processing with U-net (training set: 512 phantoms containing multiple disks). Images after post-processing (network output): (a) testing set of 128 disk phantoms, (b) testing set of 128 phantoms containing larger disks, (c) testing set of 128 ellipse phantoms.

(a) (b) (c) (d)

Figure 56. FBP post-processing with U-net (training set: 512 phantoms containing multiple disks). Images after post-processing (network output): (a) testing set of 128 square phantoms, (b) testing set of 128 phantoms containing larger squares, (c) testing set of 128 diamond phantoms, (d) testing set of 128 stripe phantoms.

In the case when the training set contained single disk patterns and the testing set consisted of multiple disk patterns, the generalization capability of U-net was also high, as shown in Fig. 57.



(a) (b) (c) (d)

Figure 57. FBP post-processing with U-net (training set: 512 disk phantoms, testing set: 128 phantoms containing multiple disks that may intersect): images after post-processing (network output).

Summary. The experiments demonstrated that an increased number of disks in the training set did not affect significantly the generalization capability of U-net tested on different patterns. In the case of the training set containing single disk patterns and the testing set consisting of multiple disk patterns, the generalization capability of U-net did not change significantly as well.

4.3 Fully learned image reconstruction

This section describes the experiments with the AUTOMAP deep neural network applied to learn the full reconstruction process in CT. The following experimental cases were considered to test the generalization capability of the AUTOMAP network:

- the 512 element training set with sinograms of single disk phantoms was taken as an input (Section 4.3.1),
- the 50000 element training set with sinograms of phantoms containing multiple disks was used (Section 4.3.2),
- the 50000 element training set with sinograms of phantoms containing multiple disks, diamonds, and stripes was taken (Section 4.3.3),
- the 50000 element training set with sinograms of phantoms containing multiple disks, diamonds, stripes, and squares was used (Section 4.3.4).

In all four experiments, the datasets consisting of disks, ellipses, squares of two sizes, diamonds, or stripes were used for testing.

4.3.1 Training set of 512 single disk patterns

The following experiments were aimed to test the generalization capability of the one-step approach. The AUTOMAP DNN was trained on the dataset of 512 elements, since this size of a training dataset was shown to be optimal in the experiments concerning the two-step approach, when U-net was used to post-process the image reconstructions (see Section 4.2.1). The sinograms of single disk phantoms were used as input data for the AUTOMAP DNN.

As shown in Fig. 58, the AUTOMAP trained on single disk phantoms failed to reconstruct from the sinograms containing 1 to 3 disks: disjoined disks were seriously corrupted (Fig. 58(h)), while intersected disks were treated as a single disk (Fig. 58(e)).

When applied to reconstructing from the images containing ellipses (Fig. 59), squares of two sizes (Fig. 60, Fig. 61) and diamonds (Fig. 62) the AUTOMAP network tended to treat these patterns like a single disk. Moreover, the position of the pattern close to the edges resulted in artefacts scattered throughout the reconstructions. In the case when the

AUTOMAP was tested on sinograms of stripes, the reconstructions contained shapeless patterns dispersed throughout the images, as shown in Fig. 63.

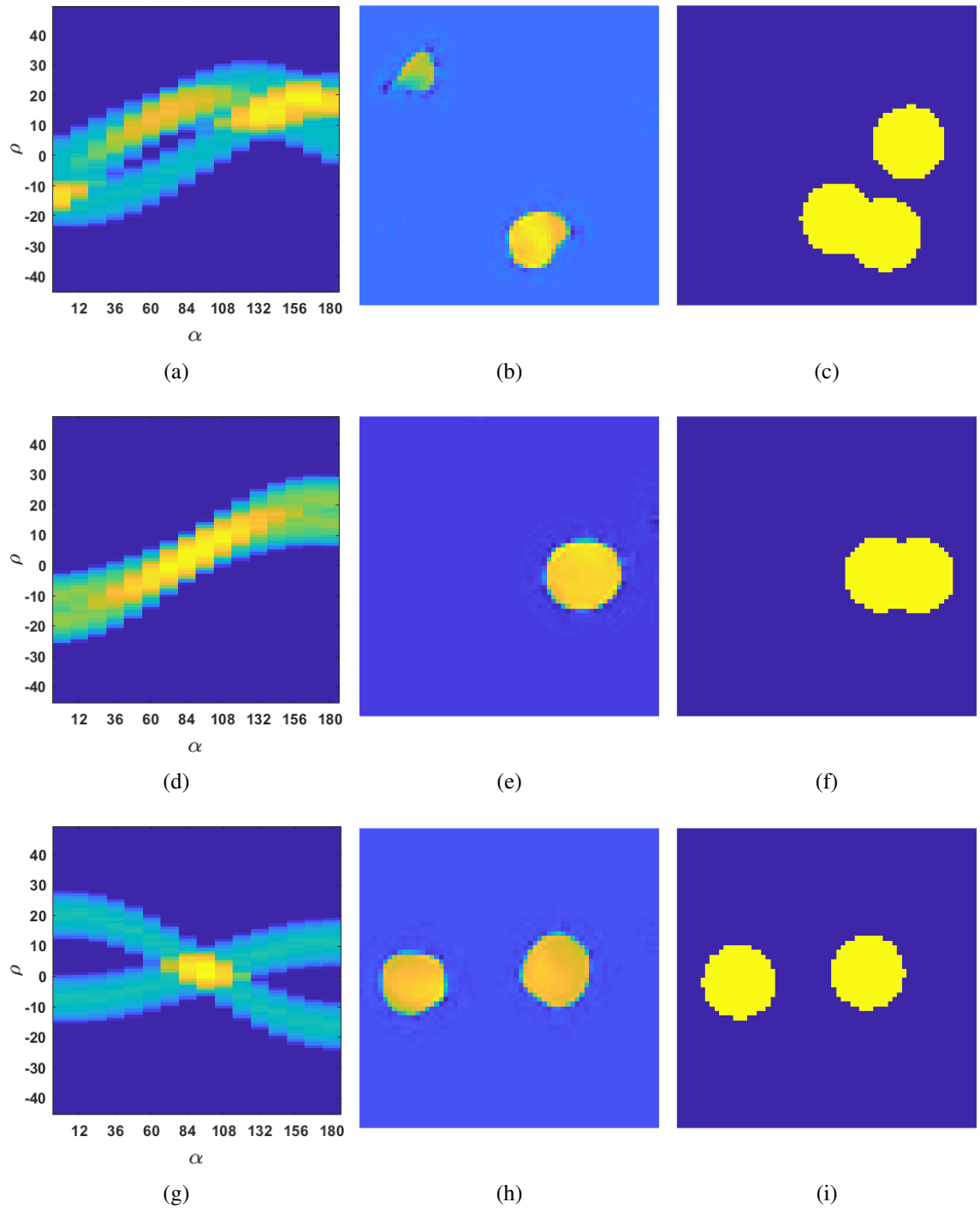


Figure 58. Fully learned image reconstruction with AUTOMAP (training set: 512 disk phantoms, testing set: 128 phantoms with multiple disks): (a), (d), (g) sinograms of digital phantoms (network input); (b), (e), (h) reconstructions (network output); (c), (f), (i) ground truth data (multiple disk phantoms). The generalization capability of the network is weak: two intersected disks are reconstructed like a single disk, and the shapes of the patterns are significantly corrupted.

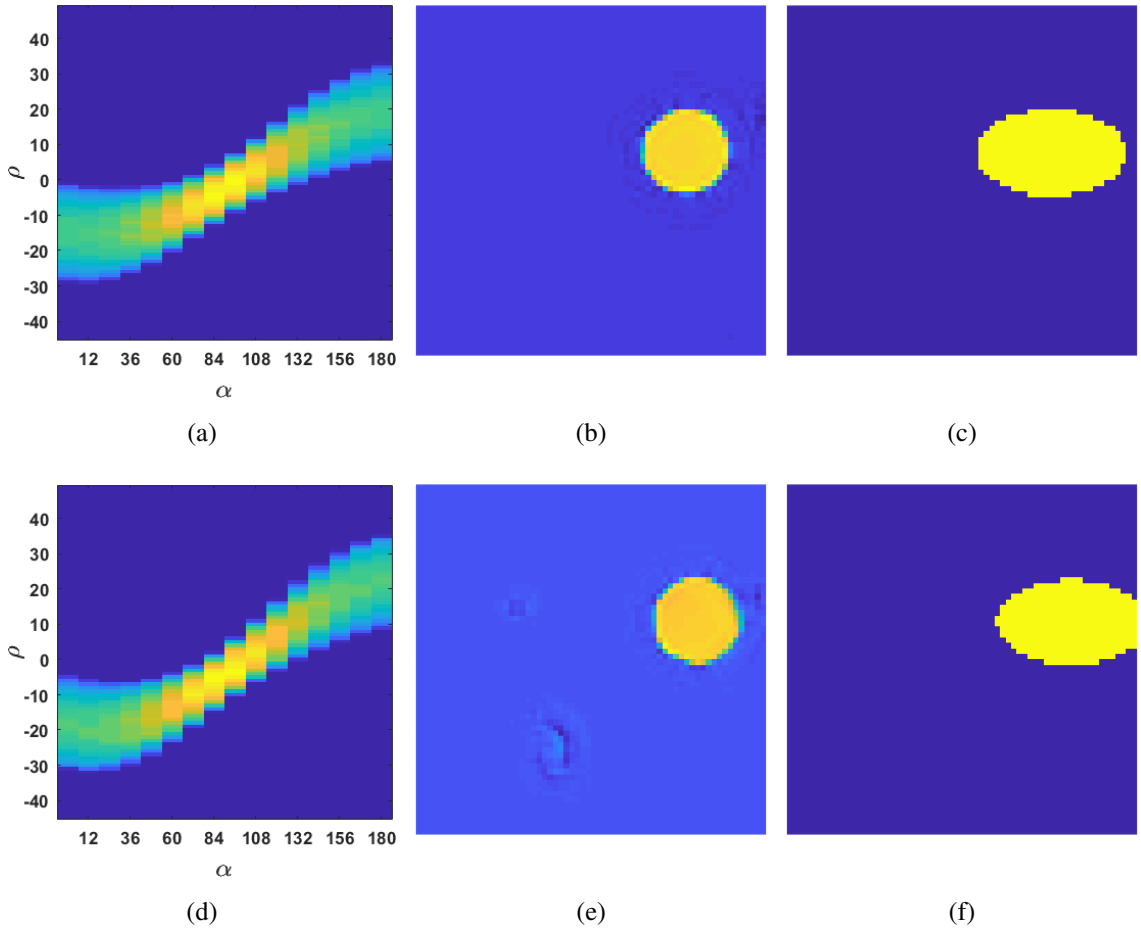


Figure 59. Fully learned image reconstruction with AUTOMAP (training set: 512 disk phantoms, testing set: 128 ellipse phantoms): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (ellipse phantoms). The ellipses are processed like a single disk, and some artefacts appear, if the ellipse extends beyond the image border (e).

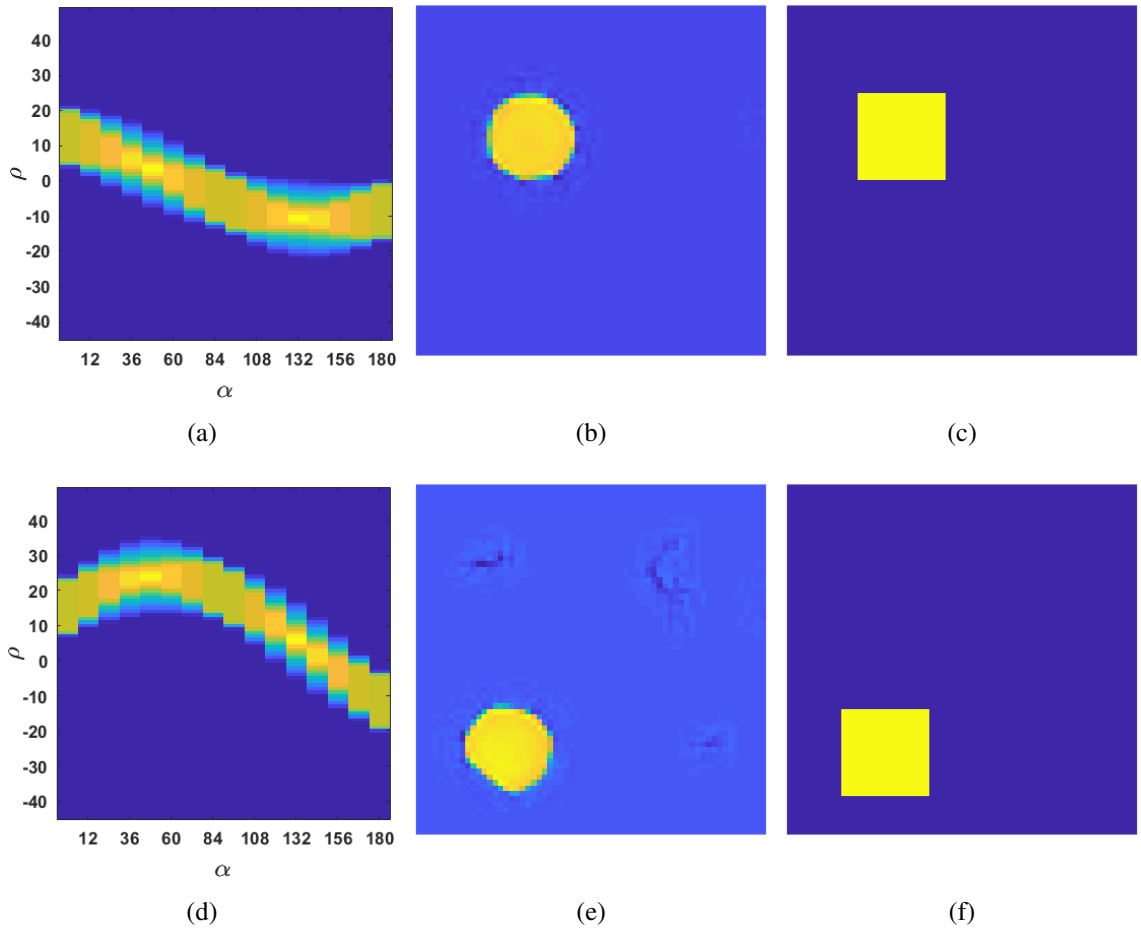


Figure 60. Fully learned image reconstruction with AUTOMAP (training set: 512 disk phantoms, testing set: 128 square phantoms): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (square phantoms). A square is treated like a single disk, and if the square is closed to the image edges, the artefacts are scattered throughout the reconstruction (e).

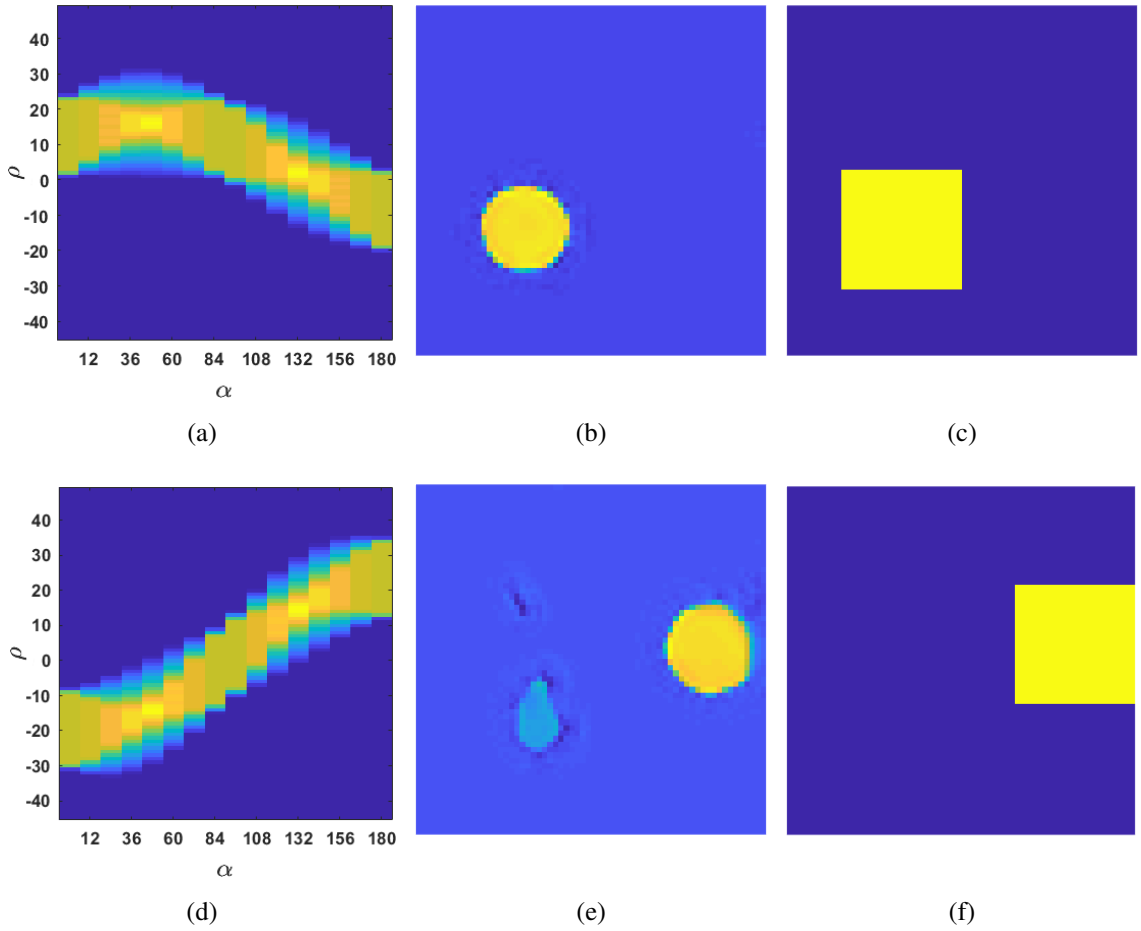
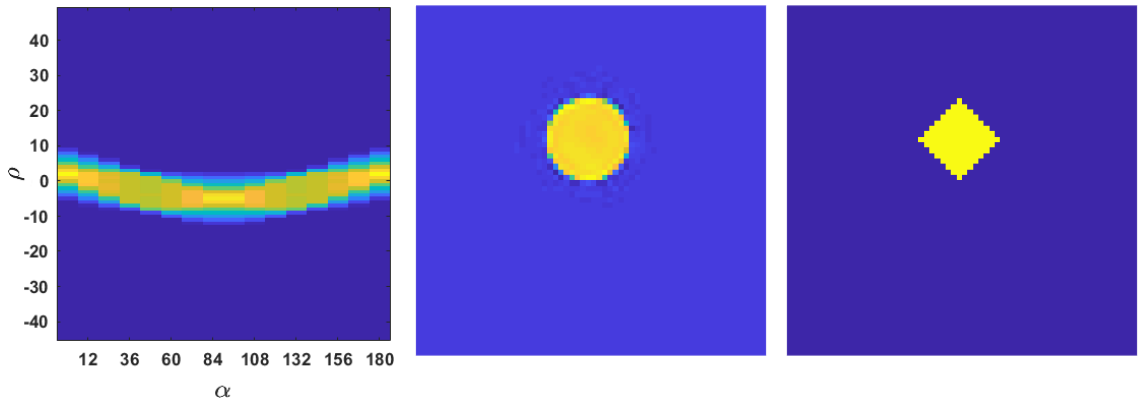


Figure 61. Fully learned image reconstruction with AUTOMAP (training set: 512 disk phantoms, testing set: 128 phantoms with larger squares): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (square phantoms). In the reconstructions there are disks of the smaller size rather than squares, and also some artefacts appear if the square is close to the image boundary (e).



(a) (b) (c)

Figure 62. Fully learned image reconstruction with AUTOMAP (training set: 512 disk phantoms, testing set: 128 diamond phantoms): (a) sinogram of digital phantom (network input), (b) reconstruction (network output), (c) ground truth data (diamond phantom). A diamond is treated like a single disk.

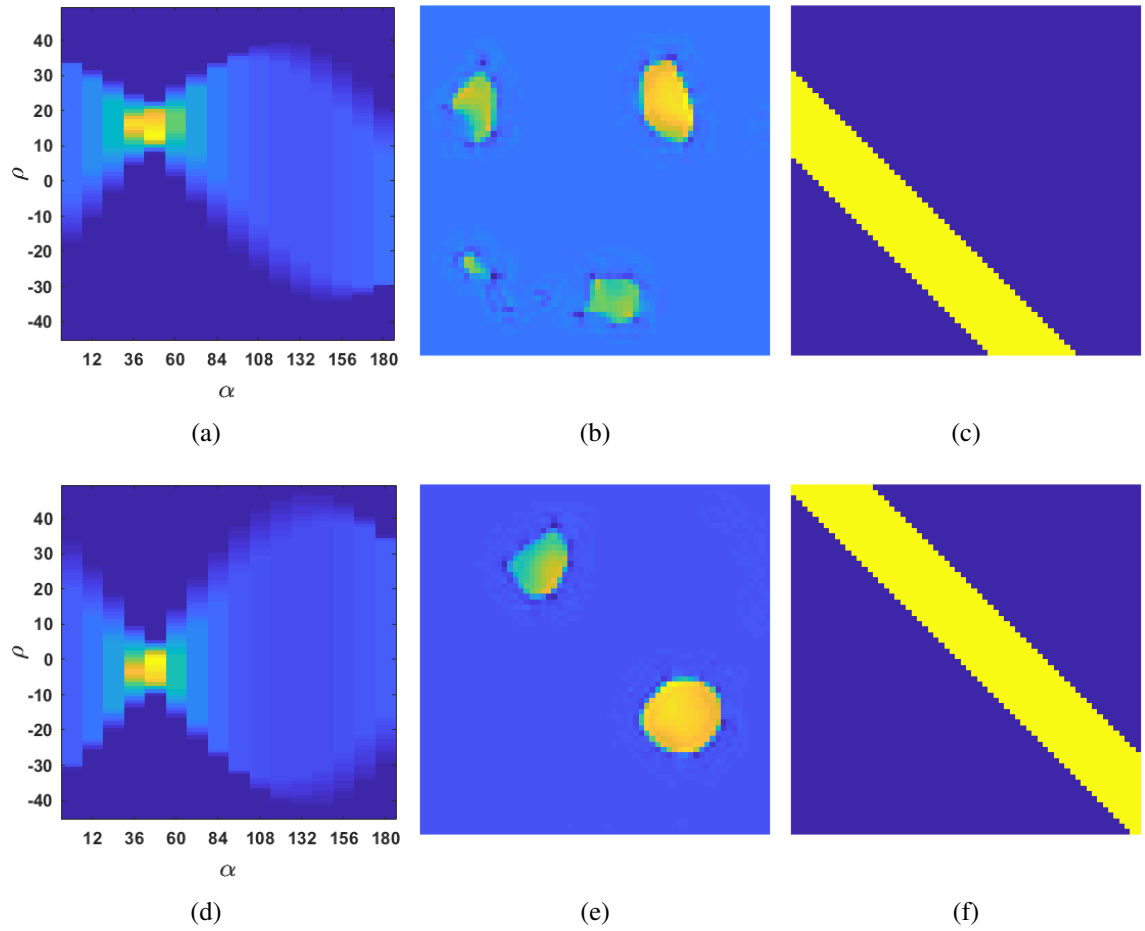


Figure 63. Fully learned image reconstruction with AUTOMAP (training set: 512 disk phantoms, testing set: 128 stripe phantoms): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (stripe phantoms). The displacement of a stripe from the main diagonal results in shapeless patterns scattered throughout the reconstruction (b).

Summary. Stripes in the testing set were shown to result in shapeless patterns scattered throughout the reconstructions. Multiple disks, ellipses, squares, and diamonds in phantoms were treated by the network as disks, and the reconstructions contained a lot of artefacts in most testing cases. Therefore it can be concluded that AUTOMAP was not able to generalize when the training dataset contained only 512 disk phantoms due to an insufficient amount of training data.

4.3.2 Training set of 50000 elements, 1 to 10 disks in sinograms

In the set of experiments described in this section, according to recommendations given in [9], the size of the training dataset was increased up to 50000 elements. In addition, for a better performance this dataset included the sinograms of phantoms with 1 to 10 disks that were allowed to intersect. It should be noted that the dataset of 50000 elements required much more training epochs to deliver the satisfactory accuracy.

The AUTOMAP performed well while testing on ellipses (Fig. 64), reconstructing the shape accurately, except for the case of an ellipse intersected with an edge (Fig 64(e)).

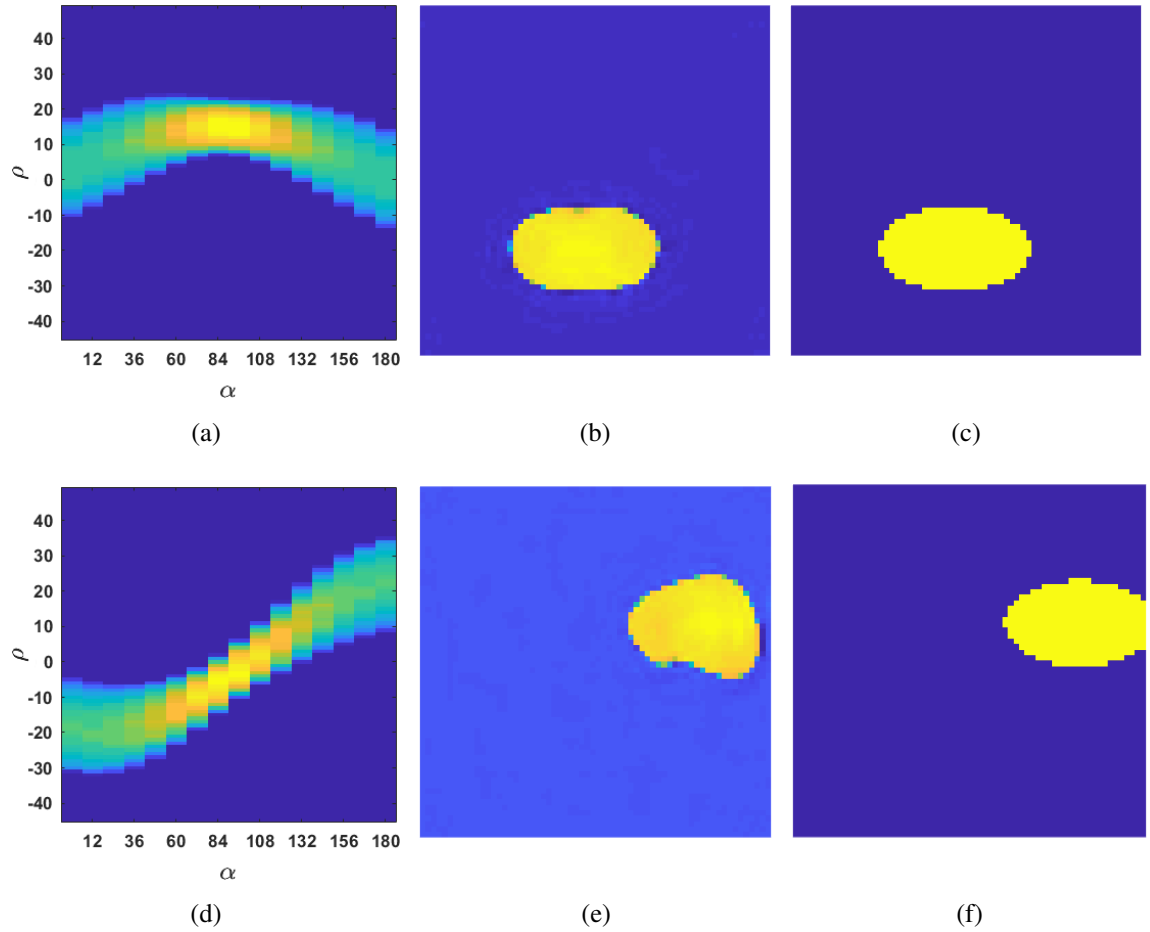
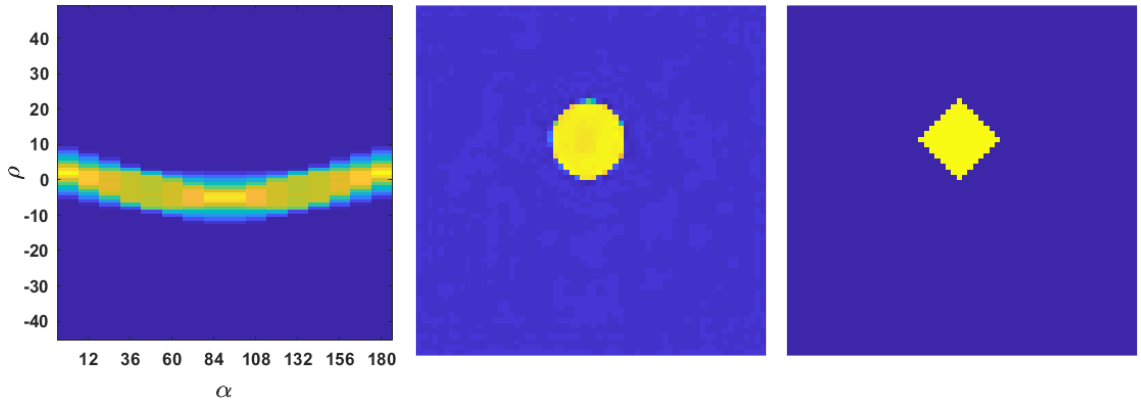


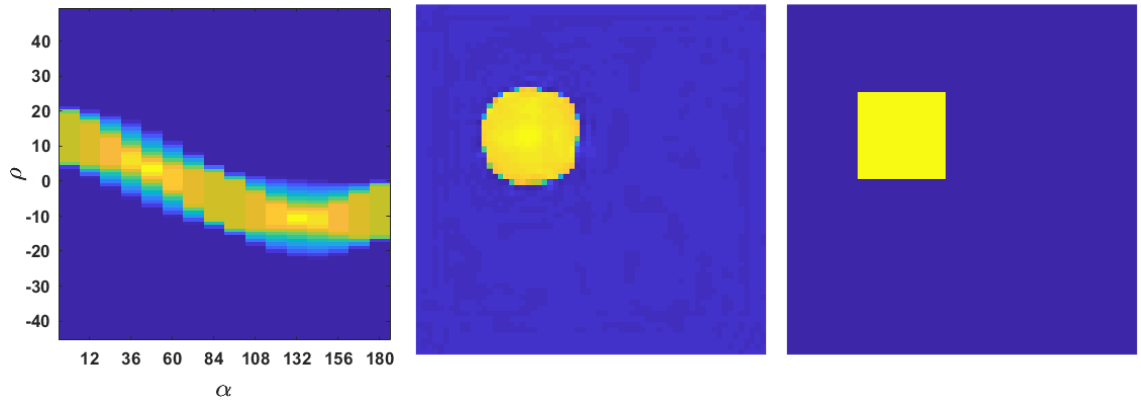
Figure 64. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, testing set: 128 ellipse phantoms): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (ellipse phantoms). The shape of an ellipse is reconstructed accurately, except the case of the ellipse extended beyond the image edges (e).

Concerning tests on patterns with right angles, the AUTOMAP still treated diamonds (Fig. 65) and squares (Fig. 66) as disks, but the shape of larger squares was slightly improved and looked like an approximation by four intersected disks (Fig. 67). As shown in Fig. 68, the generalization to stripes failed, since their shape was severely corrupted.



(a) (b) (c)

Figure 65. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, testing set: 128 diamond phantoms): (a) sinogram of digital phantom (network input); (b) reconstruction (network output); (c) ground truth data (diamond phantom). The diamond is treated like a disk.



(a) (b) (c)

Figure 66. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, testing set: 128 square phantoms): (a) sinogram of digital phantom (network input); (b) reconstruction (network output); (c) ground truth data (square phantom). The pattern in the reconstructed image is similar to a disk rather than a square.

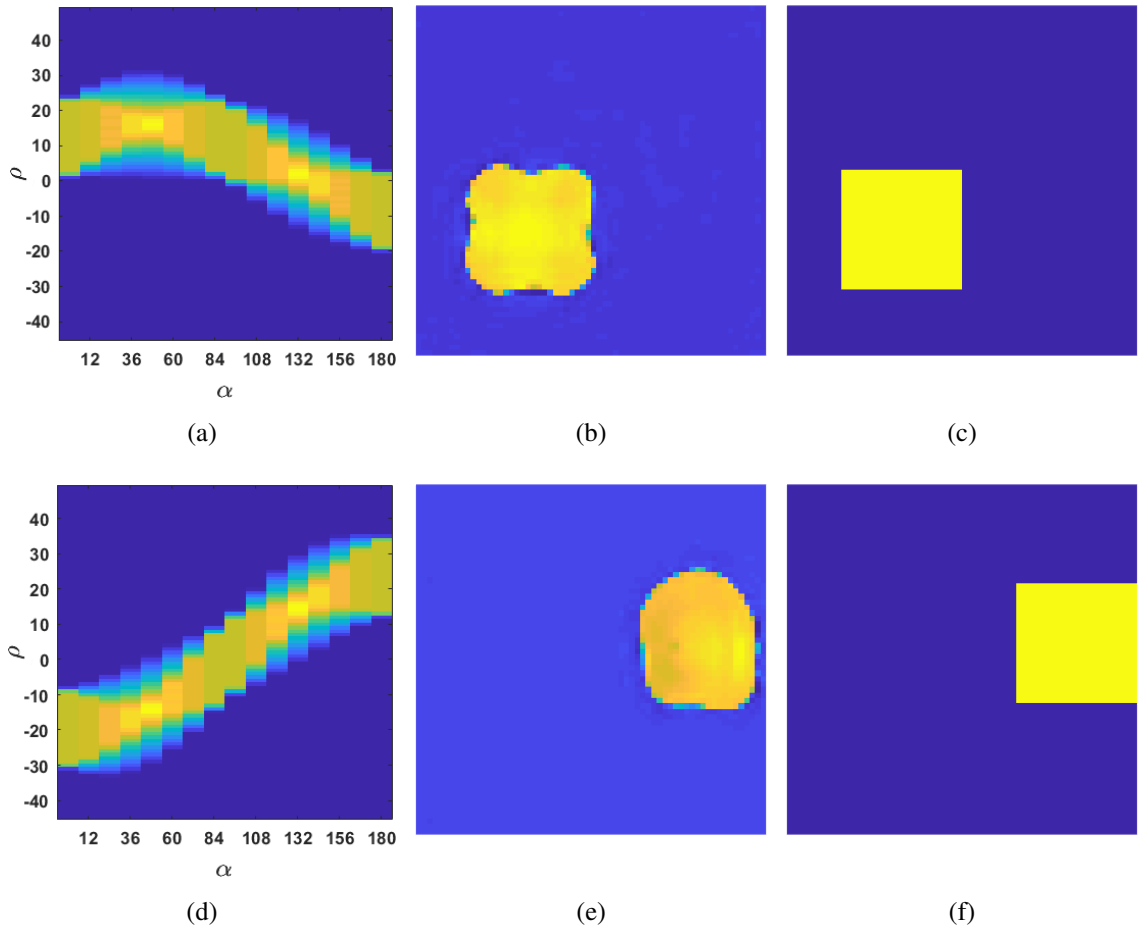


Figure 67. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, testing set: 128 phantoms with larger squares): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (square phantoms). If a square is close to the edges, its shape is more corrupted (e).

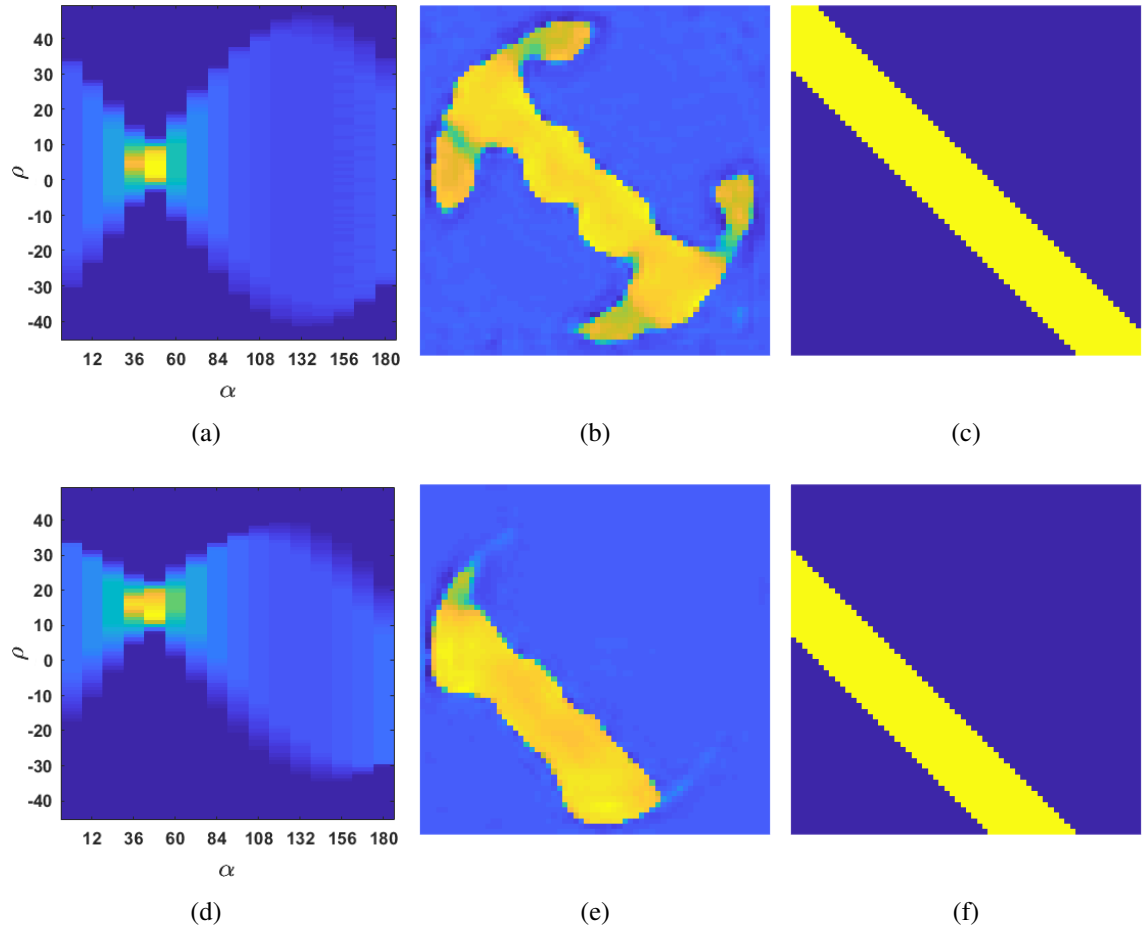
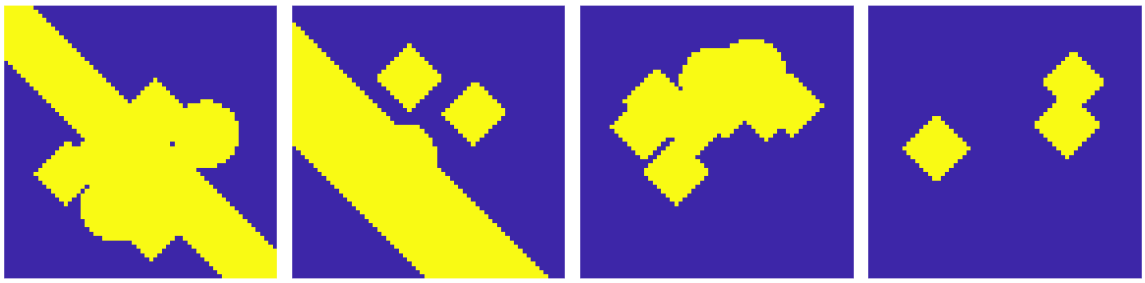


Figure 68. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, testing set: 128 stripe phantoms): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (stripe phantoms). The shape of stripes is significantly corrupted in the reconstructions.

Summary. The AUTOMAP DNN trained on 50000 elements containing phantoms with 1 to 10 disk patterns that were allowed to intersect, showed better generalization capability than the same network trained on the dataset of 512 elements (Section 4.3.1): it processed ellipses and larger squares (but the shape of the square pattern was approximated by four joint disks) more accurately. However, the generalization capability failed significantly when the testing sets contained squares, diamonds (these patters were still treated as disks) or stripe patterns.

4.3.3 Training set of 50000 elements, 1 to 10 disks, diamonds and stripes in sinograms

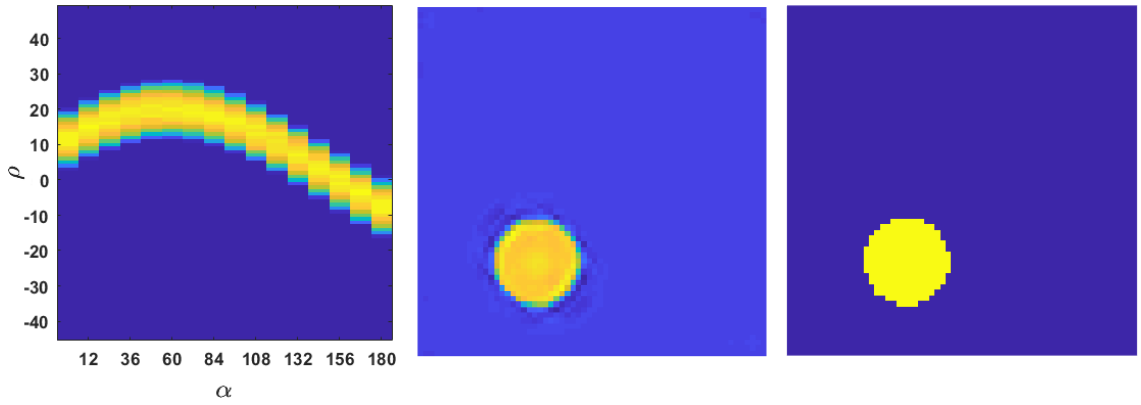
As discussed in Section 4.3.2, an increased size of the training set and a greater number of disks in sinograms improved the generalization capability of the network tested on ellipses and bigger squares. However, the reconstructions containing squares, diamonds and stripes still suffered from poor quality. Therefore, in this set of experiments, diamonds and stripes were included in phantoms. Thus, the training set of 50000 elements contained sinograms of phantoms with 1 to 10 disks, diamonds and stripes that were allowed to intersect and form sophisticated patterns, as shown in Fig. 69.



(a) (b) (c) (d)

Figure 69. Examples of ground truth images in the training set: digital phantoms with 1 to 10 disks, diamonds and stripes that may intersect.

The AUTOMAP network performed well when tested on disks (Fig. 70) and ellipses (Fig. 71). As expected, reconstructions of diamonds (Fig. 72) and stripes (Fig. 73) looked more accurate as compared to experiments described in Section 4.3.2. However, there was no significant improvement in the shape of reconstructed squares (Fig. 74) and larger squares (Fig. 75).



(a) (b) (c)

Figure 70. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds and stripes; testing set: 128 disk phantoms): (a) sinogram of digital phantom (network input); (b) reconstruction (network output); (c) ground truth data (disk phantom).

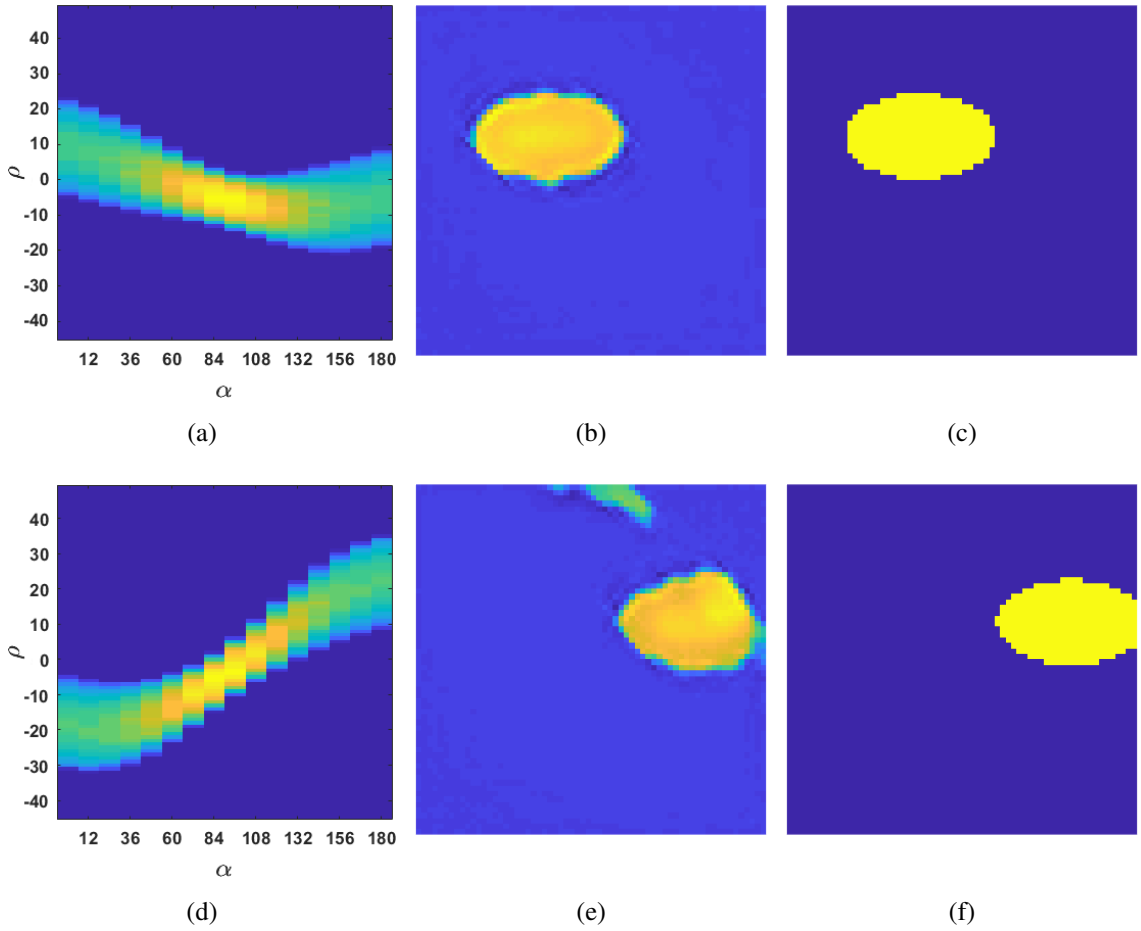
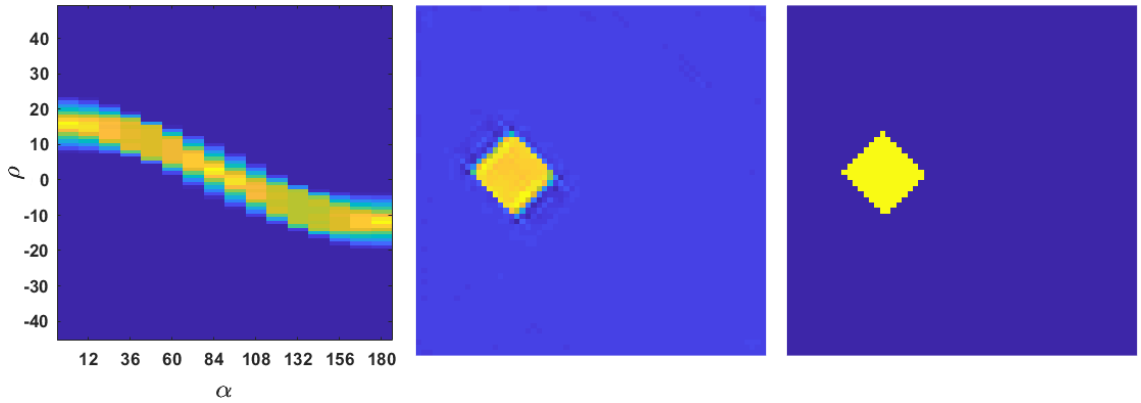
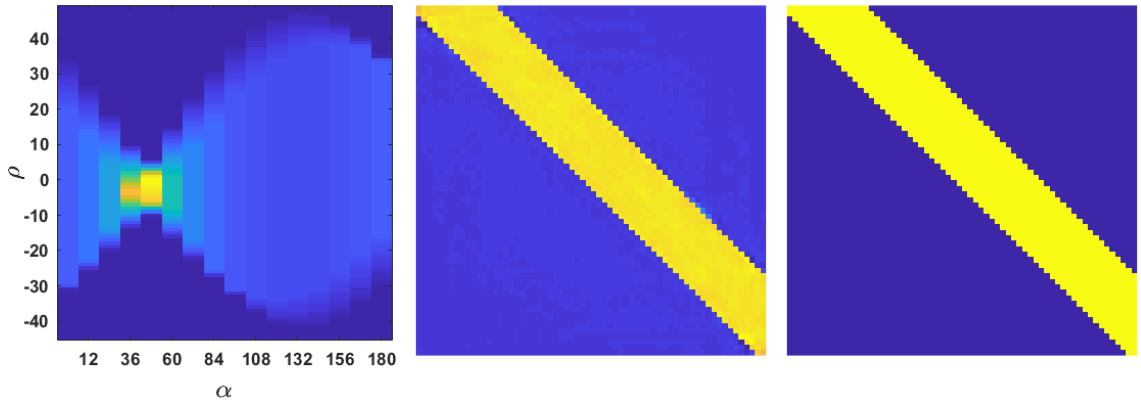


Figure 71. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds and stripes; testing set: 128 ellipse phantoms): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (ellipse phantoms). If an ellipse is close to the edges its shape is severely corrupted in the reconstruction (e), otherwise the reconstructed shape is good enough.



(a) (b) (c)

Figure 72. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds and stripes; testing set: 128 diamond phantoms): (a) sinogram of digital phantom (network input); (b) reconstruction (network output); (c) ground truth data (diamond phantom).



(a) (b) (c)

Figure 73. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds and stripes; testing set: 128 stripe phantoms): (a) sinogram of digital phantom (network input); (b) reconstruction (network output); (c) ground truth data (stripe phantom).

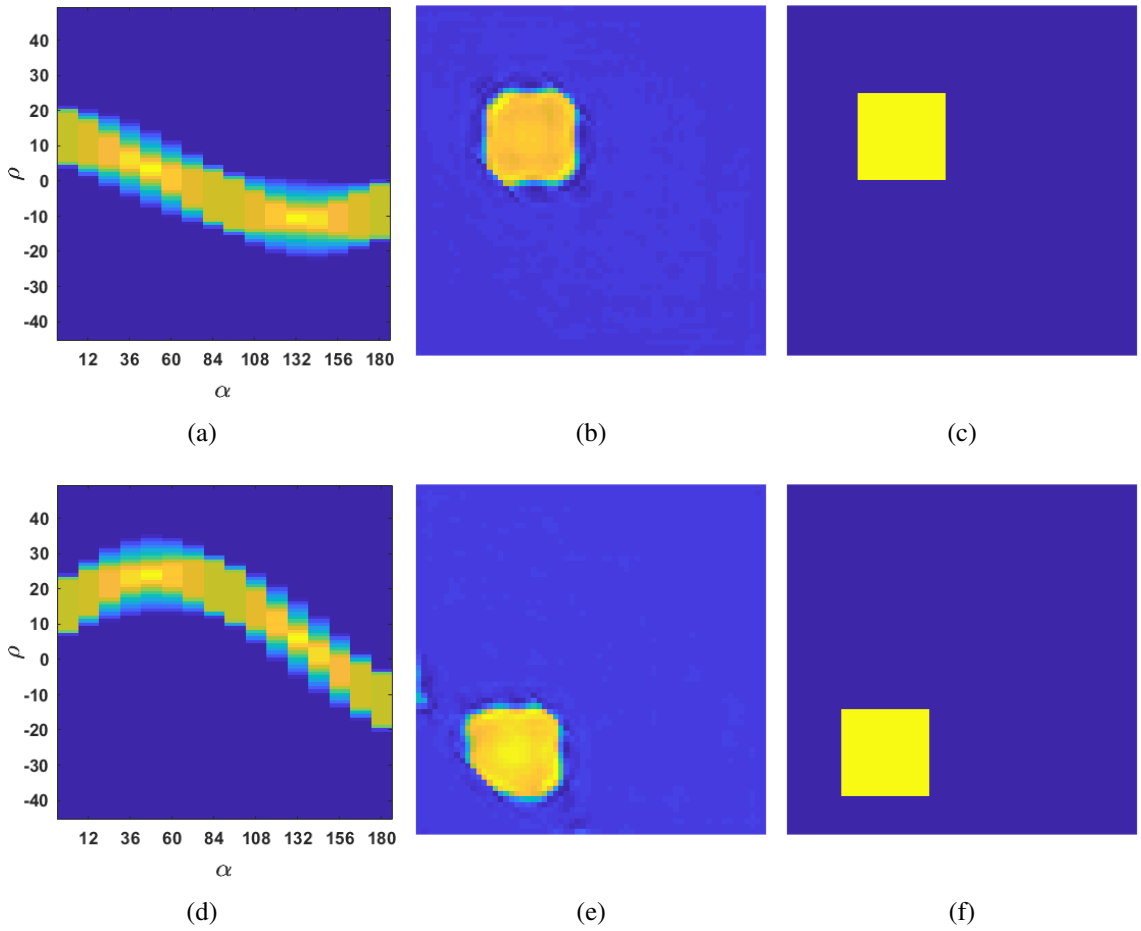


Figure 74. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds and stripes; testing set: 128 square phantoms): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (square phantoms). If a square is close to the edges its shape is more corrupted (e).

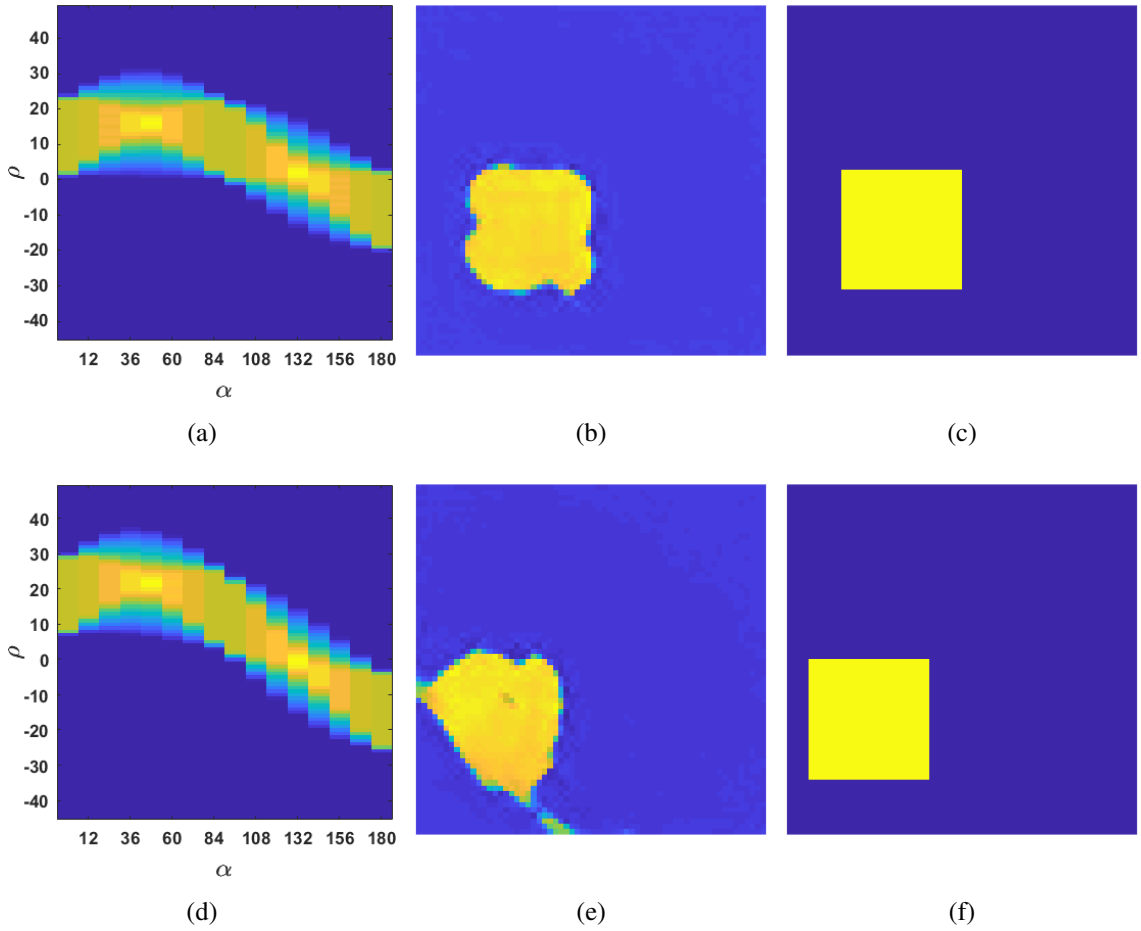


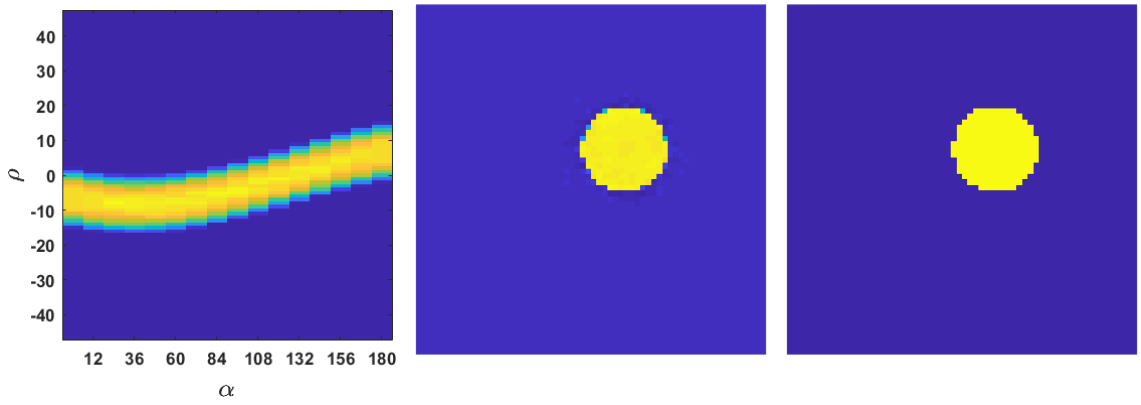
Figure 75. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds and stripes; testing set: 128 phantoms with larger squares): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (square phantoms). In the case when the square is close to the edges (e), it looks like a shapeless pattern, and some artefacts appear.

Summary. When the training set of 50000 elements, containing phantoms with multiple disks, diamonds, and stripes, was used, the generalization capability of the AUTOMAP network slightly improved. It accurately processed disk patterns, ellipses (except for the case where the patterns are close to the edges), diamond and stripe patterns. However, the shapes of squares and larger squares were still corrupted and treated like disks (or their union) owing to a lack of such patterns in the training set.

4.3.4 Training set of 50000 elements, 1 to 10 disks, diamonds, stripes and squares in sinograms

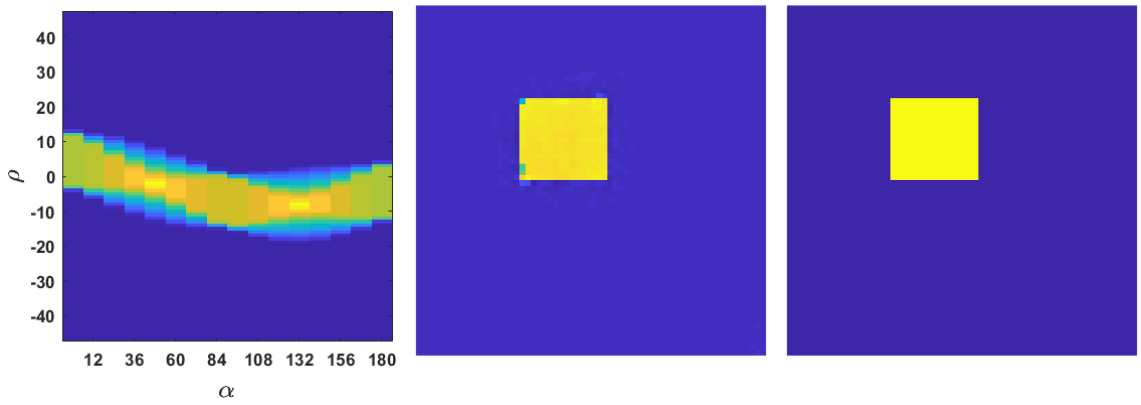
As discussed in Section 4.3.3, in the case when the AUTOMAP was trained on 50000 phantoms with 1 to 10 disks, diamonds and stripes, the network failed to process square and larger square patterns. Consequently, in the set of experiments described in this section, square patterns were added to the training phantoms.

As a result, AUTOMAP successfully processed disk phantoms (Fig. 76) as well as squares (Fig. 77), diamonds (Fig. 78), and stripes (Fig. 79), regardless of their location relative to the edges. It should be noted that ellipse phantoms (Fig. 80) and larger squares (Fig. 81) were also accurately reconstructed, even though these patterns were not included in the training set. However, the generalization capability of the network failed significantly, if ellipses and larger squares were located close to the edges: the patterns were severely corrupted and the reconstructions contained some artefacts, as shown in Fig. 80(e), 81(e).



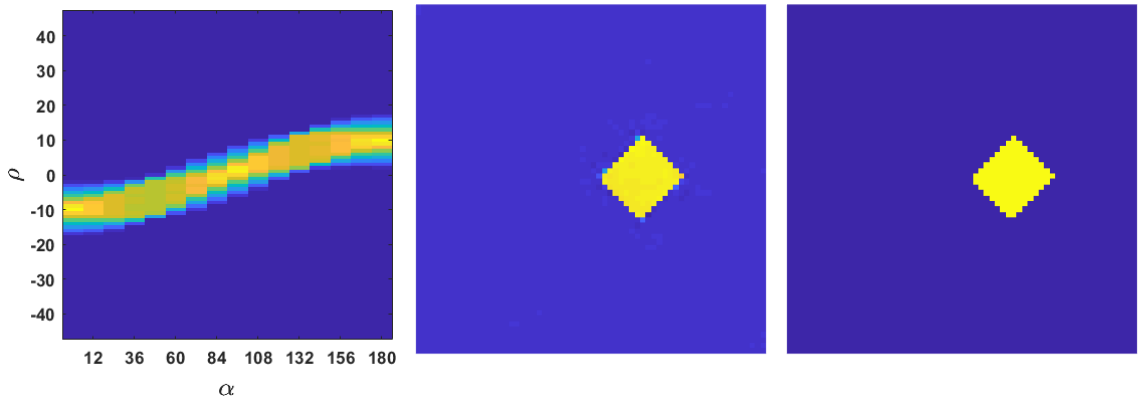
(a) (b) (c)

Figure 76. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds, stripes and squares; testing set: 128 disk phantoms): (a) sinogram of digital phantom (network input); (b) reconstruction (network output); (c) ground truth data (disk phantom).



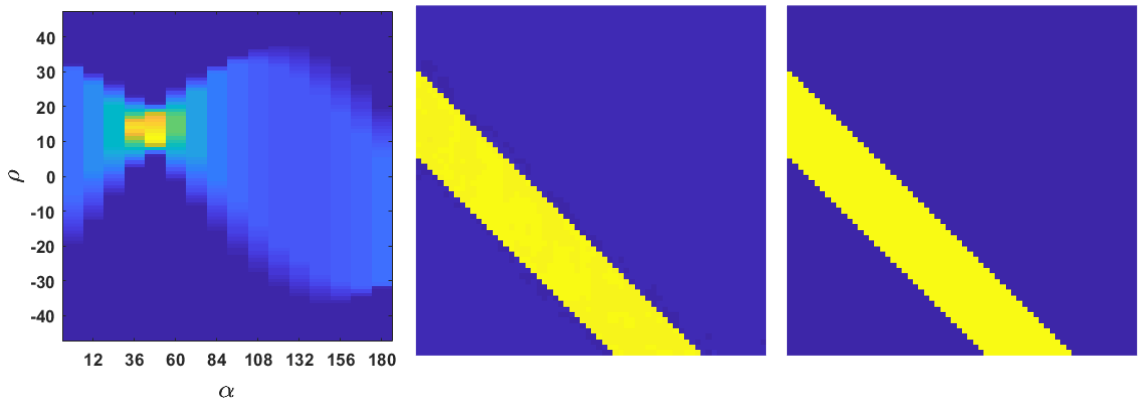
(a) (b) (c)

Figure 77. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds, stripes and squares; testing set: 128 square phantoms): (a) sinogram of digital phantom (network input); (b) reconstruction (network output); (c) ground truth data (square phantom).



(a) (b) (c)

Figure 78. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds, stripes and squares; testing set: 128 diamond phantoms): (a) sinogram of digital phantom (network input); (b) reconstruction (network output); (c) ground truth data (diamond phantom).



(a) (b) (c)

Figure 79. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds, stripes and squares; testing set: 128 stripe phantoms): (a) sinogram of digital phantom (network input); (b) reconstruction (network output); (c) ground truth data (stripe phantom).

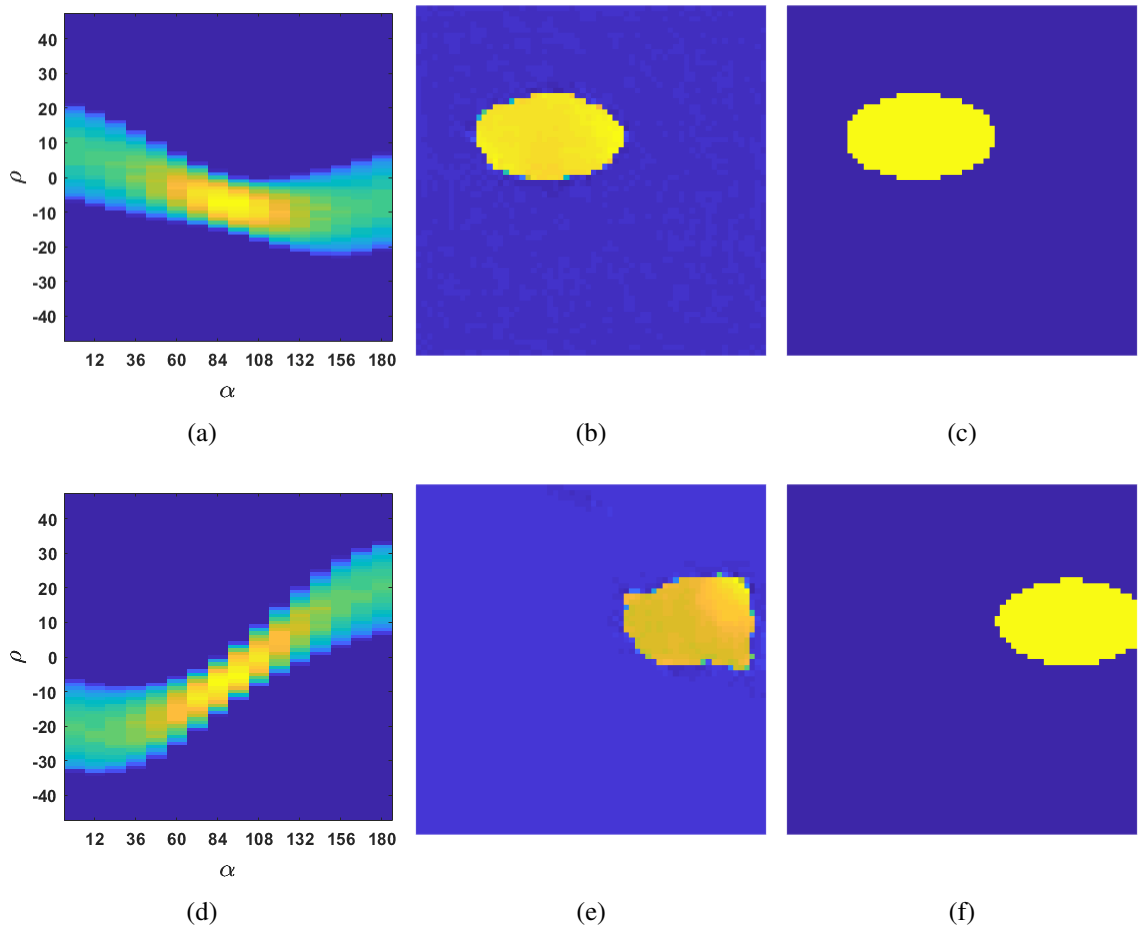


Figure 80. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds, stripes and squares; testing set: 128 ellipse phantoms): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (ellipse phantoms). If an ellipse is close to the edges its shape is severely corrupted in the reconstruction (e), otherwise the reconstructed shape is good enough, even though ellipse patterns were not included in the training set.

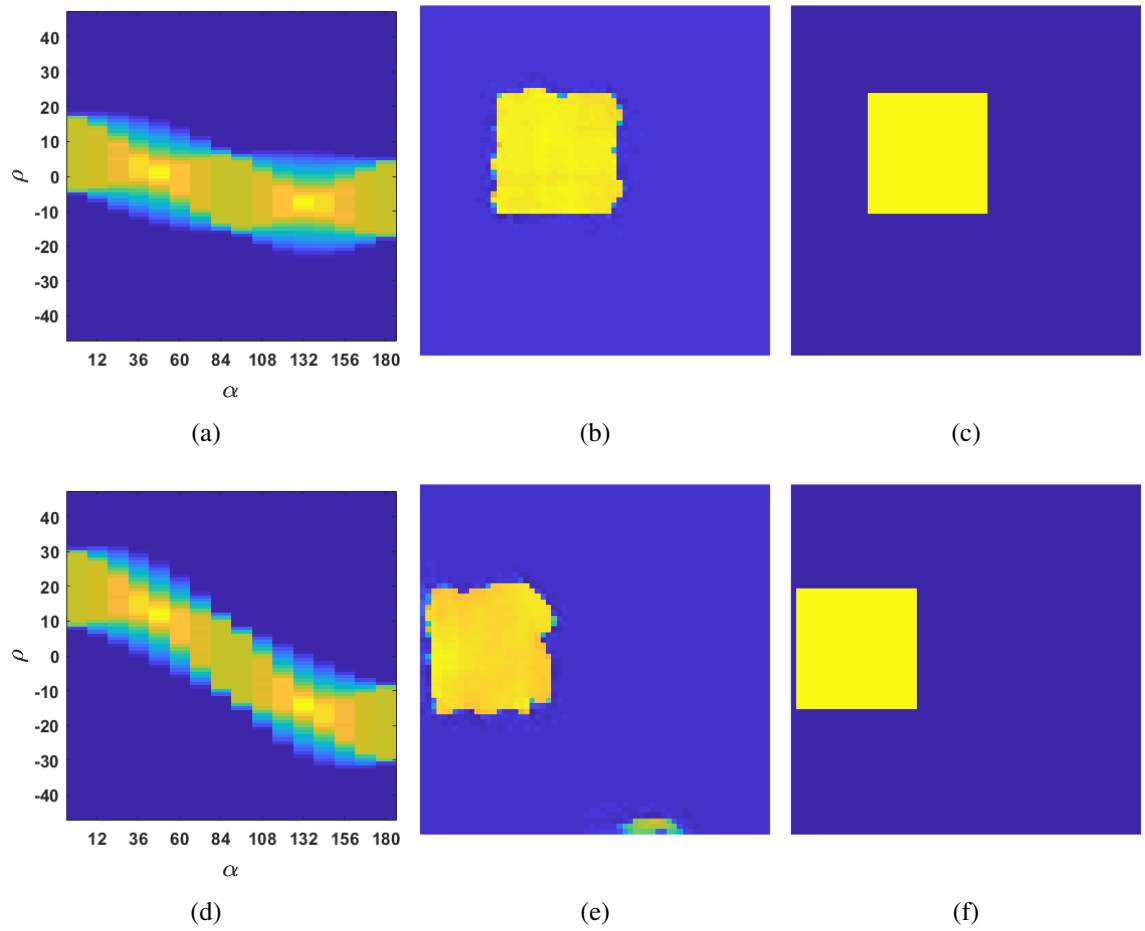


Figure 81. Fully learned image reconstruction with AUTOMAP (training set: 50000 phantoms with 1 to 10 disks, diamonds, stripes and squares; testing set: 128 phantoms with larger squares): (a), (d) sinograms of digital phantoms (network input); (b), (e) reconstructions (network output); (c), (f) ground truth data (square phantoms). In the case when the square is close to the edges (e), its shape is more corrupted, and some artefacts appear.

Summary. When the set of 50000 phantoms with multiple disks, diamonds, stripes and squares, was used for training, the generalization capability of the AUTOMAP network improved significantly. It accurately processed not only patterns included in the training set (disks, squares, diamonds and stripes), but also ellipse patterns and larger squares that were not included in the training set.

Comparison between one- and two-step approaches. U-net (two-step approach) outperformed AUTOMAP (one-step approach) in terms of generalization capability, since U-net trained on disk phantoms performed successfully while testing on sets containing multiple disks, ellipses, squares, diamonds or stripes. Unlike U-net, AUTOMAP was able to reconstruct different patterns, only if these patterns were included in the training set.

5 DISCUSSION

5.1 U-net post-processing of reconstructed images

In the first part of experiments (Section 4.2) concerning the two-step approach in CT image reconstruction, the generalization capability of U-net applied to post-process FBP-reconstructions was tested. The experiments were aimed to find an optimal size of the training set, to examine the effect of different projection numbers in the training set on the reconstruction quality, and to check whether the phantoms containing multiple disks affect the generalization capability of the network.

To find an optimal size of the training set delivering the best generalization capability of U-net, the network was trained on three different datasets consisting of 256, 512 and 1024 single disk phantoms. Various datasets consisting of 128 single disk, ellipse, square, diamond or stripe phantoms each, were employed for the purpose of testing. U-net was applied to the filtered backprojections in the testing sets, and next the obtained reconstructions were compared to the ground truth images. The results indicated that the generalization capability of U-net trained on 256 disk phantoms was high when the network was tested on the patterns of the same size as those in the training set. However, in the case of larger patterns, U-net post-processing resulted in the reconstructions with corrupted patterns (Fig. 25, 29). The U-net networks trained on 512 and 1024 phantoms processed the larger patterns adequately, but while testing on stripe patterns they resulted in artefacts near the edges (Fig. 37, 44). In general, the experimental results demonstrated that U-net trained on 512 disk phantoms was more accurate than U-net trained on 1024 elements since the latter network resulted in the noisy interiors of the larger disk and square patterns in the reconstructions (Fig. 40, 43). The possible reason for the poor generalization capability of the U-net trained on 1024 phantoms is overfitting (overtraining). It can be concluded that in terms of the generalization capability, the dataset containing 512 disk phantoms was shown to be optimal for U-net training.

To examine the effect of the different number of projections in the training set on the ability of U-net to generalize, the network was trained on four different datasets containing 512 disks and the various number of projections. The first training set contained projections computed at 12 different angles evenly spaced between 0 and 180 degrees. The second, third and fourth datasets contained projections computed (in the same manner) at 10, 8 and 4 angles, respectively. Experimentation with different testing sets (containing 128 disk, ellipse, square, diamond or stripe patterns, projections computed at 16 angles)

suggested that a decreasing number of projections in the training set resulted in the deteriorating accuracy of the reconstructions for all the patterns (Fig. 46 – 51), except for the stripes that were equally well-processed regardless of the projection number in the training set (Fig. 52). In the case of the training set containing a small fixed number of projections (computed at 4 angles) and four different testing sets containing disk phantoms with projections computed at 12, 10, 8 or 4 angles, similar results were obtained: the fewer projections were used in the testing sets, the worse reconstruction quality was observed (Fig. 53). The practical motivation for testing a fewer number of projections is that in medical CT imaging it is important to reduce the number of X-ray beams used for scanning and thus mitigate the radiation exposure on the target object.

To test the effect of phantoms containing multiple patterns in the training set on the reconstruction quality, U-net was trained on 512 phantoms containing 1 to 3 disks, and then tested on the datasets each containing 128 phantoms with single disk, square, ellipse, diamond or stripe patterns. The experimental results demonstrated that an increased number of disks in the training phantoms did not provide significant improvement in the generalization capability of U-net (Fig. 55, 56). An additional experiment showed that in the case of the training set containing a single disk and the testing set containing multiple disks, the generalization capability of U-net did not change dramatically as well (Fig. 57).

5.2 Fully-learned image reconstruction with AUTOMAP

In the second part of experiments (Section 4.3) concerning the one-step approach, the generalization capability of the AUTOMAP deep neural network (applied to learn the full reconstruction process in CT) was tested. Unlike the prior experiments where the FBP-reconstructions were used as a U-net input, AUTOMAP received sinograms (synthetic data generated by applying the Radon transform to the digital phantoms) as its input. The output reconstructions were compared to the ground truth images. The experiments were meant to test the generalization capability of the network trained on four datasets of different sizes and contents.

In the first case, AUTOMAP was trained on the dataset of 512 elements, containing sinograms of single disk phantoms. This size of the training set was chosen, because it resulted in the best generalization capability of U-net used for reconstruction post-filtering. However, the AUTOMAP trained on 512 elements was not able to generalize: multiple disk patterns, single ellipses, squares, and diamonds in phantoms were treated by the network as disks, and the reconstructions contained a lot of artefacts (Fig. 58 – 62). With regard to

the testing set consisting of stripe patterns, the output reconstructions contained scattered shapeless patterns (Fig. 63).

The poor generalization capability of the AUTOMAP trained on 512 elements prompted an increase in the size of the training dataset up to 50000 elements in the second case. Moreover, this dataset included phantoms containing 1 to 10 disks. It should be noted that the number of training epochs, as well as batch size, were also increased (see Section 4.1). As a result, the AUTOMAP network processed ellipse patterns accurately, except for the patterns located close to the edges (Fig. 64). Larger square patterns were approximated by four joint disks in the reconstructions (Fig. 67), but in general, they looked more accurate than those reconstructed by the AUTOMAP trained on 512 disk phantoms. However, the generalization capability failed significantly when the training sets contained squares, diamonds (these patters were still treated as disks) or stripe patterns (Fig. 66, 65, 68).

Since an increase in the training set size and the number of disks in phantoms positively affected the ability of AUTOMAP to process ellipse patterns in the testing set, but did not significantly improve the quality of reconstructions containing patterns with right angles and stripe patterns, it was decided to add diamond and stripe patterns to the training set in the third set of experimens. Thus, phantoms in the third training set included 1 to 10 disk, diamond and stripe patterns that were allowed to intersect and form more sophisticated patterns. As expected, in this case AUTOMAP adequately processed single diamond and stripe patterns (Fig. 72, 73). Disk and ellipse patterns were also accurately reconstructed (Fig. 70, 71), while the shapes of squares and larger squares were still badly corrupted (Fig. 74, 75).

In the final case, the training set contained 50000 phantoms with 1 to 10 disk, diamond, stripe, and square patterns together. When the square patterns were added to the training phantoms, the AUTOMAP performed successfully on the testing sets containing disk, square, diamond and stripe phantoms (Fig. 76 – 79). The network also showed better generalization capability on larger squares, if those were not located close to the edges, otherwise larger squares were more corrupted, and artefacts appeared in the reconstructions (Fig. 81). Moreover, the AUTOMAP accurately processed ellipse phantoms (except for the ellipses located close to the edges), even though they were not included in the training set (Fig. 80). As a result, it was concluded that the network adequately processed the patterns in the testing sets only if similar patterns were included in the dataset used for training, otherwise the network showed a lack of generalization capability.

6 CONCLUSION

The aim of this study was to compare two different learned image reconstruction approaches in terms of their generalization capability. In the first (two-step) approach, U-net was used only for post-processing of image reconstructions, whereas in the second (one-step) approach, the AUTOMAP deep neural network performed the full reconstruction process. For training and testing of the networks the synthetic datasets were created. For the one-step approach the datasets consisted of sinograms generated in MATLAB by applying the Radon transform to the digital phantoms containing different patterns (e.g. disks). For the two-step approach, the synthetic input data constituted the filtered back-projections that were obtained by applying the inverse Radon transform with Ram-Lak filter to the sinograms.

The experimental results imply that the two-step approach performs better, and requires much smaller and less diverse training dataset to deliver good reconstruction quality: U-net trained on 512 disk phantoms shows high generalization capability when applied to different testing sets containing phantoms with multiple disks, ellipses, squares, larger disks and squares, diamonds and stripes. As for the one-step approach, AUTOMAP requires highly diverse training sets to show accurate generalization capability: unlike U-net, the AUTOMAP network is able to reconstruct different patterns in testing sets accurately, only when these patterns are included in the training set. In addition, a large variety of patterns in the training set results in a considerable amount of time needed for the network to be trained. The issue of time-consuming training can be solved, for example, by using GPUs that fasten the training process. However, the matter of the training set diversity can be crucial in the context of computed tomography, since usually the amount of real data available for training is quite limited.

REFERENCES

[1] Frank Natterer. *The Mathematics of Computerized Tomography*. Society for Industrial and Applied Mathematics, 2001.

[2] Timothy G. Feeman. *The Mathematics of Medical Imaging*. Springer Undergraduate Texts in Mathematics and Technology, 2010.

[3] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 3d edition, 2008.

[4] Ge Wang. A perspective on deep imaging. *IEEE Access*, 4:8914–8924, 2016.

[5] Sarah Jane Hamilton and Andreas Hauptmann. Deep D-bar: Real-time electrical impedance tomography imaging with deep neural networks. *IEEE Transactions on Medical Imaging*, 37(10):2367–2377, 2018.

[6] Sarah Hamilton, Asko Hänninen, Andreas Hauptmann, and Ville Kolehmainen. Beltrami-net: domain independent deep D-bar learning for absolute imaging with electrical impedance tomography (a-EIT). *Physiological Measurement*, 40(7):074002, 2019.

[7] Yoeri Boink, Srirang Manohar, and Christoph Brune. A partially-learned algorithm for joint photo-acoustic reconstruction and segmentation. *IEEE Transactions on Medical Imaging*, 39(1):129–139, 2020.

[8] Jonas Adler and Ozan Öktem. Learned primal-dual reconstruction. *IEEE Transactions on Medical Imaging*, 37:1322–1332, 2018.

[9] Bo Zhu, Jeremiah Z. Liu, Stephen F. Cauley, Bruce R. Rosen, and Matthew S. Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555:487–492, 2018.

[10] Edward A. Bender. *An Introduction to Mathematical Modeling*. Dover Publications, 1st edition, 1987.

[11] Aleksandr O. Vatul'yan. *Obratnye Zadachi v Mekhanike Deformiruemogo Tverdogo Tela [Inverse Problems of Solid Mechanics]*. FIZMATLIT [Publishing House of Physical, Mathematical and Technical Literature], 2007.

[12] Albert Tarantola. *Inverse Problem Theory*. Elsevier, 1st edition, 1987.

[13] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005.

[14] Alun Jones and Christopher Taylor. Solving inverse problems in computer vision by scale space reconstruction. In *IAPR Workshop on Machine Vision Application*, 1994.

[15] Marco Prato and Luca Zanni. Inverse problems in machine learning: an application to brain activity interpretation. *Journal of Physics: Conference Series*, 135:012085, 2008.

[16] Benyuan Sun, Shihong Yue, Zhenhua Hao, Ziqiang Cui, and Huaxiang Wang. An improved Tikhonov regularization method for lung cancer monitoring using electrical impedance tomography. *IEEE Sensors Journal*, 19(8):3049–3057, 2019.

[17] Yuqing Wan, Andrea Borsic, John Heaney, John Seigne, Alan Schned, Michael Baker, Shaun Wason, Alex Hartov, and Ryan Halter. Transrectal electrical impedance tomography of the prostate: spatially coregistered pathological findings for prostate cancer detection. *Medical physics*, 40(6):063102, 2013.

[18] David S. Holder. Electrical impedance tomography (EIT) of brain function. *Brain Topography*, 5(2):87–93, 2019.

[19] Jen Beatty. *The Radon Transform and the Mathematics of Medical Imaging*. Honors Theses. Paper 646, 2012. `https://digitalcommons.colby.edu/honorstheses/646`.

[20] Johann Radon. Über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten. *Berichte Sächsische Akademie der Wissenschaften*, 29:262–277, 1917.

[21] Johann Radon. On the determination of functions from their integral values along certain manifolds (P.C. Parks, Trans.). *IEEE Transactions on Medical Imaging*, MI-5(4):170–176, 1986.

[22] Stanley R. Deans. *The Radon Transform and Some of Its Applications*. New York: John Wiley & Sons, 1983.

[23] Jonathan M. Blackledge. *Digital Image Processing*. Woodhead Publishing, 2006.

[24] Guillaume Bal and Philippe Moireau. Fast numerical inversion of the attenuated radon transform with full and partial measurements. *Inverse Problems, Institute of Physics Publishing*, 20:1137–1164, 2004.

[25] Charles L. Epstein. *Introduction to the Mathematics of Medical Imaging*. Society for Industrial and Applied Mathematics, 2d edition, 2007.

[26] Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

[27] Maria Lyra and Agapi Ploussi. Filtering in SPECT image reconstruction. *International Journal of Biomedical Imaging*, 2011:693795, 2011.

[28] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Pub., 1997.

[29] Avinash C. Kak and Malcolm Slaney. *Principles of Computerized Tomographic Imaging*. Society of Industrial and Applied Mathematics, 2001.

[30] Richard W. Hamming. *Digital Filters*. Dover Publications, 2013.

[31] Larry A. Shepp and Benjamin F. Logan. The fourier reconstruction of a head section. *IEEE Transactions on Nuclear Science*, 21(3):21–43, 1974.

[32] Richard E. Thomson and William J. Emery. *Data Analysis Methods in Physical Oceanography*. Elsevier Science, 3d edition, 2001.

[33] Michael A. King, Stephen J. Glick, et al. Interactive visual optimization of SPECT prereconstruction filtering. *Journal of Nuclear Medicine*, 28(7):1192–1198, 1987.

[34] Jonathan M. Links, Richmond W. Jeremy, et al. Wiener filtering improves quantification of regional myocardial perfusion with thallium–201 SPECT. *Journal of Nuclear Medicine*, 31(7):1230–1236, 1990.

[35] Pascal Wallisch, Mike Lusignan, Marc Benayoun, Tanya I. Baker, Adam S. Dickey, and Nicho G. Hatsopoulos. *MATLAB for Neuroscientists: An Introduction to Scientific Computing in MATLAB*. Academic Press, 2d edition, 2014.

[36] Santanu Pattanayak. *Pro Deep Learning with TensorFlow: A Mathematical Approach to Advanced Artificial Intelligence in Python*. Apress, 2017.

[37] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[38] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[39] Sergei Nikolenko, Artur Kadurin, and Ekaterina Arkhangel'skaya. *Glubokoe Obuchenie [Deep Learning]*. Piter [Saint-Petersburg], 2018.

[40] Donald O. Hebb. *The Organization of Behavior*. Wiley, 1949.

[41] Marvin Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[42] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the ICML*, 2013.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[44] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). https://arxiv.org/abs/1511.07289, 2015.

[45] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[46] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[47] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. http://arxiv.org/abs/1212.5701, 2012.

[48] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. http://arxiv.org/abs/1412.6980, 2014.

[49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems Conference*, volume 25, pages 1097–1105. MIT Press, 2012.

[50] Olga Russakovsky, Jia Deng, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science*, 9351:234–241, 2015.

[52] Martín Abadi, Ashish Agarwal, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. http://download.tensorflow.org/paper/whitepaper2015.pdf, 2015. [Online; accessed April, 24, 2020].

[53] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, 2000.